Optimisation d'un réseau de production et de distribution

Clarisse Flipo-Dhaenens

A mes parents,

A Emma, ma filleule,

A Jean-Etienne, mon mari,

Avec Tendresse,
Avec Amour.

Remerciements

Le travail présenté dans cette thèse a été initié au sein du laboratoire ARTEMIS. Il s'est poursuivi au laboratoire LEIBNIZ-IMAG dont je tiens à remercier le directeur, M. Philippe Jorrand, pour son accueil.

J'exprime toute ma reconnaissance à M. Gerd Finke, co-directeur de thèse, responsable de la formation doctorale et responsable de l'équipe dans laquelle j'ai effectué mes recherches, pour la confiance qu'il m'a témoignée dès l'année de DEA, pour le goût pour la recherche qu'il m'a transmis et pour son soutien à chaque étape.

Je remercie vivement M. Lionel Dupont, co-directeur de thèse, pour l'attention qu'il m'a accordé et pour l'ouverture industrielle qu'il m'a apportée, en illustrant nos discussions de problématiques réelles qu'il a eu l'occasion de découvrir lors de ses multiples contacts industriels.

Mes remerciements vont ensuite aux membres du jury :

A M. Christian Proust, Directeur de l'E3i à Tours, qui, en tant que coordonnateur du groupe de travail Bermudes, m'a permis de rencontrer la communauté Française d'ordonnancement, et qui m'a fait la joie et l'honneur d'accepter de présider ce jury de thèse.

A M. Pierre Dejax, Professeur à l'École Centrale de Paris et directeur adjoint du laboratoire productique et logistique, qui a accepté d'être rapporteur de mon travail de doctorat. Je le remercie de ses remarques m'ayant permis d'améliorer le document.

A M. Alain Guinet, Professeur à l'INSA de Lyon, pour avoir accepté d'être rapporteur de cette thèse, pour l'interêt qu'il a porté au sujet et pour ses précieux conseils. Je souhaite très sincérement que l'avenir rende possible de nouvelles col-

laborations de recherche.

A M. Jean-Yves Talon, Ingénieur de recherche au Centre de Recherches de Voreppe (C.R.V.) de la société Pechiney, qui m'a donné l'opportunité de m'appuyer sur une problématique industrielle réelle pour effectuer mes recherches de doctorat.

Je tiens également à remercier toutes les personnes qui ont rendu ces trois années de thèse à Grenoble très agréables. Je remercie en particulier :

Les membres du laboratoire GILCO, qui m'ont accepté dans leur communauté, et en particulier M. Bernard Penz, pour sa relecture avisée, pour ses conseils et pour le soutien qu'il m'a témoigné depuis mon DEA.

Les membres du Laboratoire d'Automatique de Grenoble (L.A.G.), avec lesquels j'ai eu l'occasion de collaborer, et plus particulièrement, M. Zdenek Binder, son équipe, et tous les membres du groupe Systèmes à Événement Discrets.

Je remercie également chaleureusement M. Cyrille Gueguen, pour son soutien moral, pour sa patience lors des différentes relectures, pour ses conseils avisés et pour son amitié.

Je remercie ensuite mes collègues du laboratoire LEIBNIZ et plus particulièrement Abdelghani, Ammar, Imed, Jiang, Mouna, Nadia, Samia, Sylvain D., Sylvain G., Vladimir, Yann et les membres du département de Mathématiques Discrètes, pour leur soutien tout au long de ces années. Qu'ils reçoivent ici le témoignage de ma sincère amitié. Je remercie tout spécialement Melle Marie-Laure Espinouse, compagne de ces "années-thèse", qui en a partagé tous les moments forts.

Mes derniers mots seront pour ma famille : mes parents et mes frères et sœurs; Camille, Hugues-Antoine, Charlotte et Jean-Baptiste qui ont toujours eu confiance en moi. Pour ma belle famille : mes beaux parents, Marie-Hélène, Guillaume et Emma. Et bien sûr pour Jean-Etienne, mon mari, qui en dépis des contraintes liées à la thèse et de ses incertitudes, m'a soutenue avec patience tout au long de ces trois années. A tous, je dédie ce mémoire, qui n'aurait pas vu le jour sans leur appui.

Table des matières

Table des figures

Liste des tableaux

Introduction

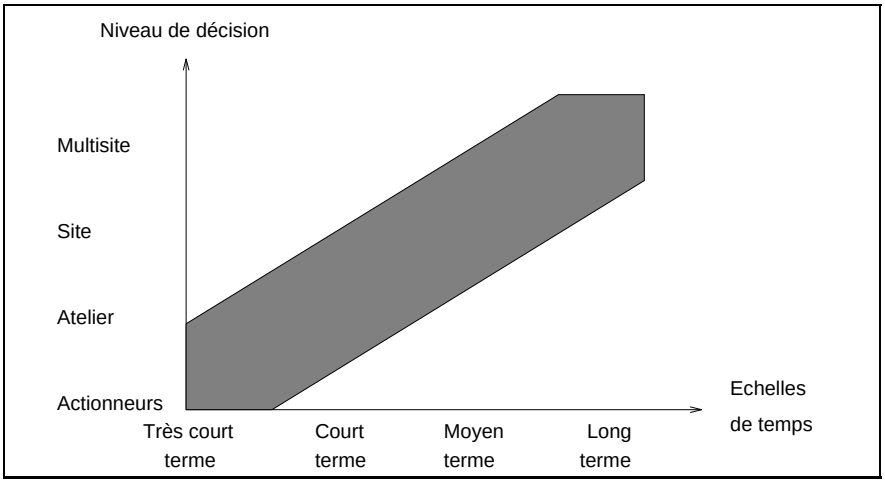
Le travail présenté dans ce mémoire a été initié par une problématique industrielle présentée par des membres du Centre de Recherches de Voreppe (C.R.V) de la société Pechiney.

Cette problématique, détaillée au chapitre 0, concerne l'optimisation globale sur un ensemble de périodes de toute une division industrielle fabriquant plusieurs produits, comprenant plusieurs sites de production et plusieurs clients répartis sur un territoire. La particularité de ce problème réside dans le lien existant entre la production et la distribution. En effet, le choix d'une affectation d'une production à une ligne de production, et donc à une usine, a des répercussions, en terme de coûts, sur la distribution.

Ce souci d'étude globale de l'ensemble du système de production traduit les préoccupations actuelles des entreprises multinationales, ou simplement multisites. En effet, beaucoup d'efforts se sont portés sur l'optimisation des systèmes de production au sein des usines et des ateliers, et encore peu, sur la coordination des différents sites de production d'une même entreprise. Pourtant, les gains réalisables, par de telles politiques globales sont certains, mais la difficulté réside dans la recherche et l'application de méthodes adéquates.

Ainsi, comme nous le verrons dans le document, la plupart des études concernant les problèmes multisites se placent à un niveau de décision à long ou moyen terme en travaillant sur des données agrégées, tandis que les décisions à court terme concernent essentiellement la planification et l'ordonnancement d'un site de production ou d'un atelier.

Sur la figure suivante, la bande grisée représente, en fonction des différentes échelles de temps, le type de problèmes couramment étudiés, et illustre le fait que plus la décision porte sur un horizon à long terme, et moins les données sont précises. Lors de la présentation de la problématique industrielle, il est très vite apparu que les temps de changement entre les produits constituent un aspect fondamental à ne pas négliger, même à un plus haut niveau. En effet, ces temps de changement pouvant être très longs, ils réduisent de façon non négligeable les capacités de production. Nous avons alors eu une démarche un peu



différente de celles couramment utilisées, en nous intéressant, au niveau multisite, non seulement à l'affectation des productions aux lignes mais également à l'ordonnancement (ou tout au moins au séquencement), afin de pouvoir prendre en compte la succession des productions et comptabiliser les temps de changement.

L'étude de la problématique multisite, multiproduit et multipériode exposée par Pechiney s'est fait en plusieurs étapes : tout d'abord, nous avons extrait un sous problème, apparaissant au niveau de l'organe de production, et qui concerne l'ordonnancement de machines parallèles non liées avec temps de changement entre les produits. Ce problème peut être vu comme un problème monosite, multiproduit et monopériode, même si il est possible d'intégrer très facilement les coûts de distribution qui sont une caractéristique importante des problématiques multisites. Puis nous avons étudié la division industrielle de façon plus globale, en considérant les aspects multisites puis finalement les aspects multipériodes.

Le mémoire est donc divisé en deux parties.

Dans la première partie nous nous consacrons à l'étude des problèmes d'ordonnancement sur machines parallèles (ou lignes de production parallèles) en privilégiant le cas de machines parallèles différentes (non liées) avec temps de changement nécessaires lors du passage d'une production à une autre sur une même ligne de production.
Pour pouvoir exposer nos recherches dans ce domaine, il nous semble important de commencer par introduire les problèmes d'ordonnancement ainsi que les notations utilisées dans la suite de ce mémoire. Il est à noter que le domaine d'application des problèmes d'ordonnancement ne concerne pas exclusivement les problèmes relatifs à la gestion de production, mais est beaucoup plus large. En effet, la théorie de l'ordonnancement s'applique dès qu'un ensemble de ressources

est à utiliser pour réaliser un ensemble de tâches, ce qui est notamment rencontré en informatique, où un ensemble de processeurs sert à l'exécution de programmes. Ainsi, pour parler de ressources, nous utiliserons indifféremment (ou suivant le contexte) les termes processeurs, machines ou lignes de production. De même, pour les tâches que nous appellerons également productions. Le chapitre 1 a pour objet la présentation des problèmes d'ordonnancement, leur classification, leur complexité et quelques méthodes classiques de résolution.

Puis, nous proposerons, au chapitre 2, un état de l'art sur l'ordonnancement de machines parallèles en insistant sur le cas des machines parallèles différentes (non liées). Cet état de l'art sera consacré à trois types de problèmes : les problèmes visant la minimisation de la durée totale de production, ceux visant la minimisation de la somme des dates de fin d'exécution et les problèmes considérant des temps de changement entre les produits. Nous ne parlerons pas des problèmes faisant intervenir des dates d'arrivée ou des dates d'échéances, car nous n'avons pas considéré cet aspect dans notre approche de la problématique.

Le dernier chapitre de cette partie (chapitre 3), sera consacré au sous-problème d'ordonnancement que nous avons étudié. Dans un premier temps, nous décrivons ce sous-problème, exposons une formulation ainsi que des bornes et des méthodes exactes. Ce problème comporte deux aspects importants : le respect des capacités de production et la minimisation des coûts. Nous nous posons alors la question du critère de recherche à utiliser dans le cas d'une approche heuristique. Ceci nous amène à exposer brièvement les notions fondamentales de l'optimisation bicritère, et à proposer une procédure itérative de résolution. Nous exposons, en fin de chapitre, des heuristiques de construction ainsi que des méthodes amélioratrices et terminons par une discussion des différents résultats.

Dans la deuxième partie de cette thèse, nous nous intéressons à une prise en compte plus globale de l'ensemble de la division industrielle et en particulier aux aspects multisites. Cette configuration de système de production comportant plusieurs sites répartis sur un territoire, n'est pas spécifique à la société Pechiney, mais devient une réalité pour un grand nombre de sociétés. En effet, la globalisation du marché et les processus de fusion poussent les entreprises à développer une politique multi-site. Les bénéfices d'une telle politique sont évidents. Tout d'abord, les sites de production peuvent être spécialisés dans certains types de produits de façon à réduire les temps et coûts de production. De plus, la dispersion des sites de production sur une région géographique permet aux sociétés d'être proches de leurs clients et réduit les temps et coûts de distribution.
Tout semble donc favorable au développement de politiques multi-sites. Pourtant, une importante difficulté de ces politiques, est la gestion coordonnée des différents sites de production et en particulier, la coordination entre la production et la distribution des produits. Souvent les décisions concernant la production

et la distribution sont prises séparément, avec peu ou pas de coordination. Mais cette approche ne peut s'appliquer pour des problématiques où la production et la distribution ont une importance équivalente en terme de coûts, ce qui est le cas de la problématique Pechiney.

Dans ce cadre, le chapitre 4 commence par présenter une revue de travaux récents portant soit sur la coordination entre les aspects de production et les aspects de distribution, soit sur des problèmes multisites. Puis nous positionnons notre problématique globale et exposons un modèle général. Nous proposons alors, pour traiter les aspects multisites et multiproduits, une décomposition spatiale qui est comparée à une approche centralisée. Cette structure a deux avantages. Premièrement, elle décompose un problème de grande taille en plusieurs problèmes de tailles plus petites, ce qui permet l'utilisation de méthodes plus performantes. Deuxièmement, elle permet, en environnement dynamique, de gérer les éventuelles perturbations à un niveau local.

Nous terminons cette deuxième partie par l'étude du cas très particulier de la division européenne de fabrication de boîtes de boisson. Pour cet exemple, il est possible de traiter la production et la distribution de plusieurs produits en même temps, sur plusieurs périodes. En effet, la taille du problème et les choix stratégiques permettent de modéliser l'ensemble des flux des produits sous forme de réseau, avec des contraintes d'intégralité, que nous traduisons en programme linéaire en nombre entiers, avec peu de variables 0-1. En utilisant cette modélisation, il est alors possible de résoudre des problèmes multiproduits, multisites et multipériodes de tailles réelles.

Chapitre 0

Un problème industriel

Au cours de ce chapitre, nous exposons la problématique industrielle à l'origine du travail de thèse présenté dans ce mémoire. L'intérêt du problème posé par la société Pechiney réside dans la recherche d'une optimisation globale de toute une division industrielle. Dans ce contexte, un point important et difficile, que nous développons plus loin, est l'interaction entre les choix d'affectation des productions aux lignes et la distribution des produits aux clients.

Nous présentons, dans un premier temps, le contexte manufacturier relatif à cette étude puis définissons quels sont les besoins en optimisation exposés par les personnes de Pechiney.

0.1 Le contexte manufacturier

La problématique posée par la société Pechiney concerne à la fois la division industrielle (ou département) s'occupant de la production de boîtes de boisson en aluminium (canettes de bières, de soda, ...) et celle s'occupant de la fabrication de tubes metallo-plastiques (tubes de dentifrice, par exemple). Nous détaillons ici, essentiellement le cas des boîtes de boisson en donnant différentes illustrations numériques sur le problème. La société Pechiney est une multinationale qui a des clients dans divers pays. Aussi, pour répondre à une demande internationale très large, la société dispose d'une gamme de produits très variée [45].

0.1.1 Les produits

Les boîtes peuvent être fabriquées à partir de deux métaux (acier ou aluminium), avoir plusieurs formats (25cl, 33cl, 44cl, 50cl, ...) et avoir différents cols de fermeture. Ce qui porte à une vingtaine le nombre de familles de produits. Le nombre de décorations des boîtes est très élevé, car les décors diffèrent en fonction des marques des clients et de leur pays d'appartenance. Ainsi, chaque famille peut

rassembler plusieurs milliers de références, spécifiques à chaque client. Les temps de changement relatifs à une modification de décor sont très faibles et peuvent être négligés par rapport aux autres changements. Il n'est donc pas nécessaire, pour notre étude, de considérer ce niveau de détails car cela complexifierait fortement le problème et n'apporterait rien de plus. Ainsi, dans la suite du document, nous ferons bien la différence entre références (comprenant la distinction de décor) et produits (équivalents à une famille de produits, sans distinction de décor). Un produit sera alors caractérisé par un métal, une forme et une taille.

0.1.2 L'organisation géographique

La division industrielle dispose sur une zone géographique donnée (le continent nord-américain, ou l'Europe, par exemple) d'un ensemble d'usines. Sur ce même territoire, sont également répartis un ensemble de stocks et un ensemble de clients (voir figure 0.1).

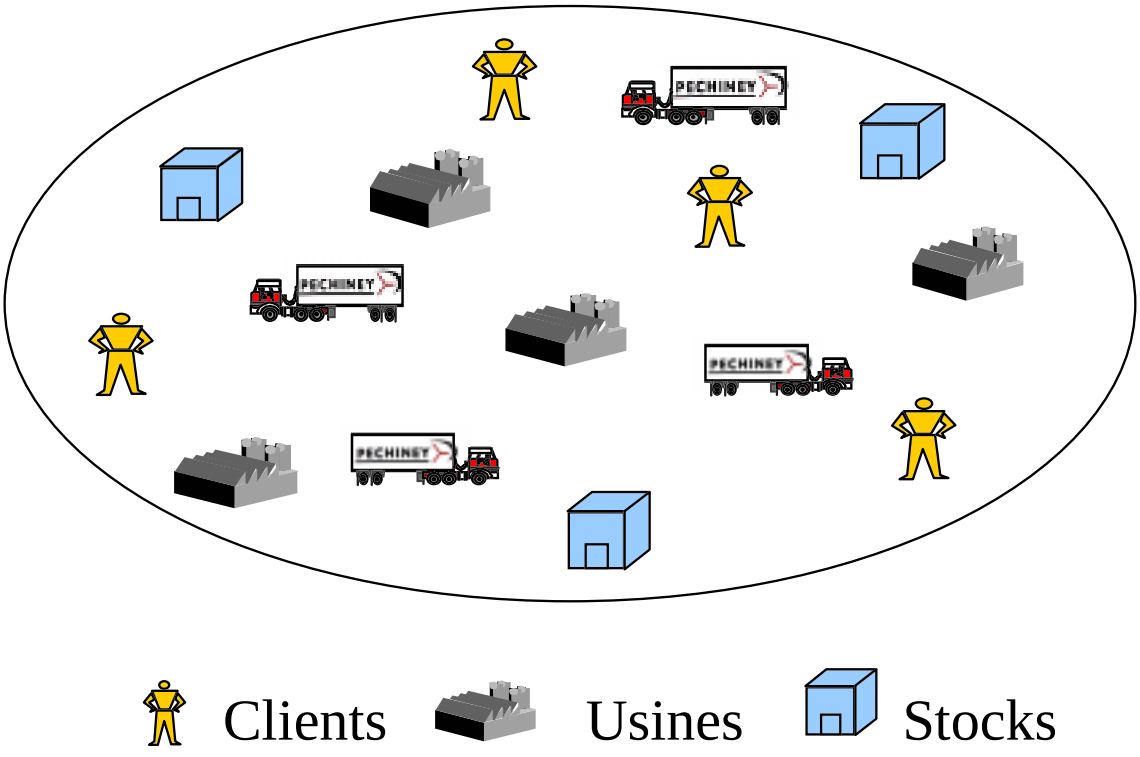


FIG. 0.1 - *La division industrielle*

Un client correspond à une société et un lieu géographique de commande et de livraison. Les clients commandent une certaine quantité de différents produits pour des dates données. Un client peut recevoir sa marchandise depuis une ou plusieurs

usines, un ou plusieurs stocks. Chaque usine a son propre stock, et il existe également des zones de stockage en dehors des usines, réparties sur l'ensemble du territoire, et servant de stocks intermédiaires pour absorber les variations d'une demande saisonnière, où les clients commandent plus à l'approche de l'été qu'en plein hiver. Il faut donc considérer des transports directs (des usines aux clients) et des transports indirects (des usines aux stocks, puis des stocks aux clients). Les coûts de ces transports ne sont pas négligeables par rapport aux coûts de production pour ces produits de faible valeur ajoutée, mais où, lors des tranports, beaucoup de volume est transporté pour peu de produits. Il s'avère que les coûts de distribution sont du même ordre de grandeur que les coûts de production.

0.1.3 Les lignes de production

Les différentes usines de la division industrielle (une dizaine pour le cas de la division européenne de boîtes de boisson) comportent plusieurs lignes de production qui ont chacune leurs propres caractéristiques (voir figure 0.2).

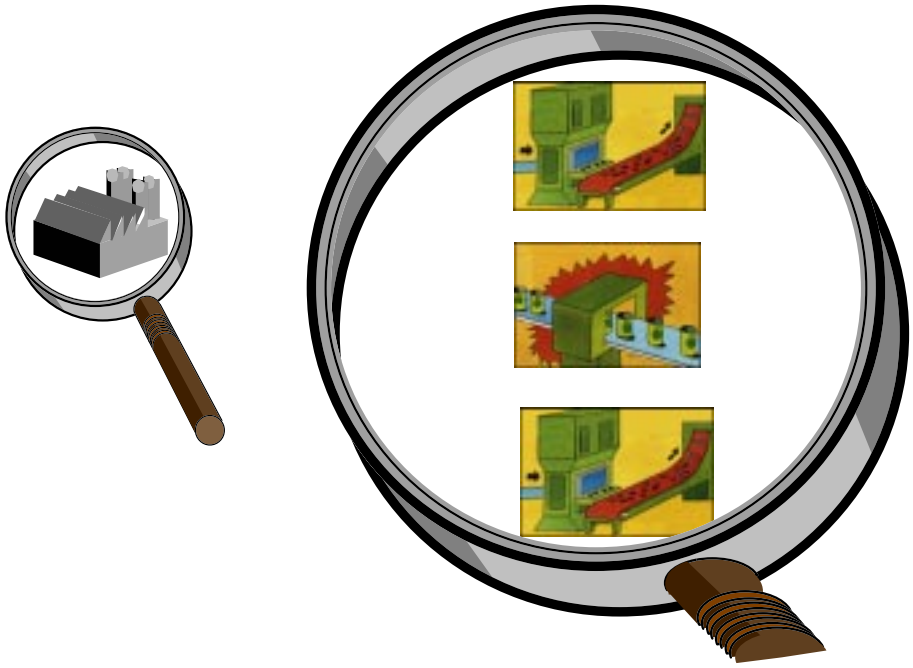


FIG. 0.2 - *Composition d'une usine*

Les lignes de production se différencient sur plusieurs points.

- Tout d'abord, elles sont affectées à un métal et peuvent donc produire des boîtes de boisson, soit en aluminium, soit en acier (mais pas les deux).
- Ensuite, elles permettent la fabrication de plusieurs produits (boîtes de formats, de cols de fermeture, de décors différents), mais chacune avec son

rendement propre en fonction de la génération technologique des machines de la ligne et des outils qu'elles comportent. Ainsi, les taux de production varient sensiblement d'une ligne à une autre en fonction du produit fabriqué.

- De plus, certaines lignes sont techniquement incapables de fabriquer tel ou tel produit (par manque d'outillage, par exemple); par contre elle peuvent avoir un très bon rendement pour d'autres produits.

- Enfin, point très important, les usines sont dispersées sur toute une région géographique (l'Amérique du nord, ou l'Europe, par exemple). Ceci implique que les coûts associés à la production et en particulier les coûts relatifs à la main d'œuvre ou à la matière première, diffèrent d'une usine à l'autre, en fonction du cours des monnaies, des législations et des lois en vigueur dans les différents pays.

Ainsi toutes ces constatations montrent qu'il est nécessaire de considérer chaque ligne de production différemment, en associant à chacune ses caractéristiques propres.

0.1.4 Le procédé de fabrication

Le procédé de fabrication est décomposé en plusieurs étapes. Différentes machines de façonnage se succèdent pour traiter la matière première (acier ou aluminium) fournie en entrée de la ligne.
Dans le cadre de cette étude, nous nous intéressons à la ligne de production en tant que telle, et non pas à chacune des machines séparément. Nous nous plaçons donc à un niveau de macro opérations. En fait la ligne fabrique les produits en continu et il est possible, sans perdre trop de précision, d'assimiler la ligne à une seule machine. Nous parlerons donc, dans la suite de ce document, indifféremment de ligne (de production), de machine ou de processeur. Le plus souvent, les lignes fonctionnent 24 heures par jour, toute l'année, sauf en cas de panne, de maintenance, de surcapacité (ce qui peut arriver exceptionnellement lors des périodes de faibles demandes).

0.1.5 Les changements de fabrication

Nous avons vu précédemment que le nombre de références différentes était assez élevé. Or, lors du passage de la production d'une référence à une autre sur une même ligne de production, il peut y avoir un changement de format, de col de fermeture ou de décor. Ces changements sont plus ou moins longs en fonction des références enchaînées et du reconditionnement de la ligne à effectuer. Ils varient entre quelques minutes pour un changement de décor, quelques heures pour un

changement de fermeture et quelques jours (jusqu'à six jours) pour un changement de format.
D'une part, ces changements de production pénalisent fortement la production car ils introduisent des temps non productifs (perte de temps machine) et peuvent, d'autre part, coûter très cher à cause du personnel spécialisé supplémentaire nécessaire et des produits rebutés suite aux différents réglages.
Dans notre étude, nous ne considérons pas les temps de changement induits par une modification de décor car ils sont peu pénalisants par rapport aux changements de type de fermeture ou de format. Nous considérons donc les différents produits (caractérisés par un métal, un format et un type de fermeture) et les temps de changement entre produits. Ces changements sont dépendant de la séquence des produits enchaînés et sont donnés par une matrice (produit, produit).

0.1.6 Des contraintes supplémentaires

Dans ce contexte, il existe des contraintes supplémentaires à respecter. Celles-ci peuvent être d'ordre technologique ou stratégique.

Les contraintes technologiques sont par exemple :

- Le respect des capacités de production de chaque ligne. Ces capacités de production, en terme du nombre de boîtes fabriquées pendant une durée donnée, dépendent des séquences de produits. En effet, les cadences de production dépendent de la ligne, mais également des produits fabriqués, et les temps de non productivité, induits par les changements de produits, sont fonction des enchaînements des produits. De plus, les rythmes de travail sont spécifiques aux lignes car le temps de travail peut varier, d'une ligne à l'autre, en fonction de la législation du pays dans lequel elles se trouvent, par exemple. La capacité horaire des lignes est alors exprimée en nombre d'heures disponibles dans le mois.

- Le respect des capacités des stocks, qui peuvent être situés au sein même des usines ou en dehors pour servir de zones d'attente entre la production et la livraison des produits.

- La prise en compte des périodes de maintenance préventive déterminées pour chacune des lignes.

Les choix stratégiques peuvent être :

- La volonté de dédier en priorité des lignes à certains produits. Par exemple une usine de la société Coca-Cola se trouve voisine d'une usine de fabrication de boîtes. Les deux usines ont été conçues de telle façon que le remplissage

des boîtes se fasse immédiatement après leur fabrication, en utilisant des tapis roulants pour le transfert d'une usine à l'autre. Une ou plusieurs lignes de cette usine sont alors utilisées en priorité pour la fabrication de boîtes de boisson de Coca-Cola et permettent l'approvisionnement de l'usine voisine. Ces lignes, peuvent toutefois, faire d'autres produits, si cela est nécessaire.

- La volonté de conserver des liens existants entre des clients et des usines. Certains clients sont fidèles à une usine. Ils ont leur interlocuteur privilégié et connaissent ainsi la qualité des produits qu'ils reçoivent. De l'autre côté, les usines connaissent les habitudes de commande de ces clients ce qui permet une meilleure anticipation de la demande et une fabrication pour les stocks en prévision des périodes de fortes demandes.

0.1.7 Les demandes

Une particularité importante, liée au type de produit fabriqué, est la saisonalité des demandes. En effet, en préparation de l'été, les clients passent de fortes commandes à partir des mois de mars-avril, et pour éviter les stocks, diminuent fortement leur demande après septembre. Ainsi, Pechiney doit anticiper ces variations de façon à équilibrer la charge de travail sur l'année.

Les demandes sont fermes à court terme, puis prévisionnelles à plus long terme. Elles peuvent s'exprimer sous forme de contrat sur une période assez longue (6 mois, par exemple), en précisant la quantité minimale à livrer, mais sans préciser le cadencement exact des livraisons.

En général, la demande peut être exprimée au mois, à partir des commandes fermes complétées par les commandes prévisionnelles. Ceci détermine les périodes de découpage de l'horizon qui sont d'une longueur d'un mois.

0.1.8 Ordre de grandeur des données

Le tableau suivant donne des indications sur les ordres de grandeur des différents éléments à considérer lors de la recherche de la planification.

0.1.9 Conclusion

Le contexte manufacturier concerne donc toute une division industrielle répartie sur une région relativement vaste. Les différentes composantes de cette division (clients, usines, stocks) ont des caractéristiques spécifiques de par leur localisation, leurs capacités de production ou de stockage et leurs commandes. Tous ces éléments forment un ensemble hétéroclite, qu'il est difficile de gérer.

Nombre de références (format, fermeture, décor)	500
Nombre de clients	100
Nombre d'usines	10
Nombre de lignes par usines	3 à 4
Cadence moyenne	1500 boîtes par minute
Temps de changement de fermeture	qq heures
Temps de changement de format	qq jours
Période de planification	1 mois

TAB. 0.1 - *Ordre de grandeur des données*

0.2 Un besoin d'optimisation globale

L'originalité de cette problématique réside dans la considération de toute une division industrielle et non plus seulement d'un site, ou d'un ilôt de production. Ce souci de globalisation intervient dans un grand projet de réduction des coûts engagé par la société Pechiney. Ce projet a pour objectif d'économiser 20% des coûts, hors matières premières, sur un horizon de deux ans. Cet objectif est tellement ambitieux que les approches d'optimisations locales se sont avérées insuffisantes car chaque unité est déjà relativement bien gérée. Ainsi, la plupart des départements se sont tournés vers des approches plus globales, remettant en cause leur organisation et leur stratégie.

0.2.1 Une production déjà bien étudiée localement

Chaque usine, ou site de production a déjà beaucoup travaillé sur l'optimisation de sa production et des études sont encore en cours.

Par exemple, une attention particulière a été portée sur les changements d'outils. Ils sont planifiés avec précaution et il existe au sein de chaque usine des équipes de spécialistes dédiés à des tâches bien précises, notamment aux réglages des machines lors de ces changements.
Chaque usine travaille sur son optimisation locale. Historiquement, les clients s'adressent directement à une usine pour passer leurs commandes et l'usine gère son carnet de commandes au mieux. D'ailleurs, cette gestion est régulièrement contrôlée par des tableaux de résultats, et une comparaison entre les différents sites est réalisée. Ceci a pour conséquence de créer une certaine concurrence entre les usines qui, de ce fait, ne cherchent ni à communiquer, ni à avoir une vue plus globale de l'ensemble du système de production. Dans ce contexte de concurrence interne, la recherche d'une optimisation commune à plusieurs sites de production, pourrait s'avérer dangereuse pour l'un ou l'autre de ces sites qui se verrait affecter

des productions moins rentables pour lui, afin d'améliorer la productivité globale de la division industrielle. Le résultat serait meilleur pour l'ensemble des sites considérés, mais individuellement moins bon pour les sites défavorisés.

Ainsi, la comparaison des résultats entre les sites semble avoir un effet motivateur pour chacun des sites, mais inhibe certainement les coopérations inter-sites.

0.2.2 Vers une optimisation globale

L'optimisation des sites de production pris individuellement étant déjà l'objet de nombreuses études, il est intéressant de savoir à quel autre niveau, une optimisation est possible. L'idée est alors d'avoir une vue globale de l'ensemble du système de production et de centraliser une partie des informations pour relever les incohérences qui existent entre les différents sites de production.

Les clients passent leurs commandes pour une période donnée. En fonction du carnet de commandes ainsi établi, et éventuellement complété par des prévisions de ventes, nous cherchons à déterminer la production à réaliser par les différentes lignes de production sur un horizon donné (horizon découpé en périodes). La demande doit être satisfaite et sa saisonalité impose une anticipation et un stockage des produits. L'objectif est la minimisation des coûts, l'ensemble des coûts étant constitué :

- des coûts de production (dépendants de la ligne de production et du produit),
- des coûts de changement de production (dépendants de la ligne de production et des produits enchaînés),
- des coûts de transport (dépendants des sites de production, de stockage et de livraison),
- des coûts de stockage (dépendants du lieu et de la durée du stockage).

0.3 Conclusion

Le problème posé par la société Pechiney pour l'optimisation de ses divisions industrielles produisant des boîtes de boisson ou des tubes métallo-plastique, est très général et générique.

En effet, cette problématique concerne une production multisite (où les sites peuvent être très éloignés les uns des autres), multiproduit (où des temps de changement sont nécessaires entre la fabrication de produits différents) et multipériode (pour considérer une demande saisonnière), ce qui caractérise un grand

nombre de divisions industrielles, quels que soient les produits fabriqués.

Nous noterons que la particularité du problème étudié, dans le cas exposé par la société Pechiney, est l'importance relative des coûts de transport par rapport aux coûts de production. Dans ce problème, il n'est pas possible de privilégier la planification de la production au dépend de la distribution (cas où les coûts de production seraient largement supérieurs), ni de s'intéresser en priorité à la distribution (cas où les coûts de distribution seraient prépondérants). Mais il est indispensable de considérer les deux aspects (production et distribution) en même temps, puisque tous deux ont des coûts de même ordre de grandeur.

PARTIE I

Un problème d'ordonnancement sur machines parallèles

Chapitre 1

Introduction à l'ordonnancement

Un des points centraux de cette thèse concerne l'ordonnancement et plus particulièrement l'ordonnancement de machines parallèles. L'ordonnancement est un champ d'investigation qui a connu un essor important ces quarante dernières années, tant par les nombreux problèmes identifiés que par l'utilisation et le développement de nombreuses techniques de résolution.
L'objet de ce chapitre est de présenter les notions fondamentales concernant les problèmes d'ordonnancement. Pour cela il convient de définir ce qu'est un problème d'ordonnancement, de montrer la diversité de ces problèmes en indiquant quelques domaines d'application, d'exposer une classification qui sera utilisée dans la suite du document, de donner quelques éléments sur la complexité des problèmes et de présenter quelques méthodes classiques de résolution de ces types de problème.

1.1 Définition et domaines d'application des problèmes d'ordonnancement

Dans un problème d'ordonnancement, quatre notions fondamentales interviennent. Ce sont les travaux (ou jobs), les ressources, les contraintes et les objectifs. Un travail se définit comme un ensemble de tâches ou d'opérations qui doivent être exécutées. Une ressource est un moyen matériel ou humain à disposition pour la réalisation d'un travail. Les contraintes représentent les limites imposées par l'environnement, tandis que l'objectif est le critère d'optimisation. Ordonnancer un ensemble de tâches consiste alors à programmer leur exécution dans le temps en leur allouant les ressources requises, et en fixant leurs dates de début d'exécution [65]. Dans les problèmes d'ordonnancement classiques, deux hypothèses importantes sont communément considérées : a) à chaque instant, une machine ne peut exécuter qu'une seule tâche, b) à chaque instant, une tâche peut être exécutée par une machine au plus. Notons que ces hypothèses peuvent être levées pour des problèmes d'ordonnancement plus spécifiques, comme l'ordonnancement par

"batch", où une machine peut exécuter plusieurs tâches en simultané, ou l'ordonnancement avec chevauchement, où une tâche peut être en cours d'exécution sur plusieurs machines à un même instant. L'objectif est de trouver une solution optimale, ou sous-optimale, dans le sens d'un critère d'évaluation donné, par l'application d'algorithmes adaptés.

Ce type de problème est très souvent rencontré dans le milieu industriel, où il s'agit de répartir l'ensemble du travail sur les machines ou ateliers de production, en respectant au mieux un ensemble de contraintes (technologiques, temporelles,...) et en visant un objectif (cadence de production, minimisation des coûts, ...). En informatique, on fait également régulièrement appel aux problèmes d'ordonnancement pour allouer les processeurs à l'exécution des programmes. Ces deux domaines, même s'ils concernent des activités très différentes, ont des problématiques semblables, qui peuvent être modélisées et résolues de la même façon. Nous citons brièvement d'autres domaines d'application des problèmes d'ordonnancement : il s'agit des domaines de la construction (suivi de projet), de l'administration (gestion du personnel, emploi du temps) ou encore, de toute structure composée d'une cellule automatisée.

Les ordonnancements sont communément représentés par des diagrammes de Gantt. Ceux-ci indiquent, selon une échelle temporelle, l'occupation des machines par les différentes tâches, les temps morts et les éventuelles indisponibilités des machines dues aux changements entre produits, par exemple (voir exemple fig 1.1).

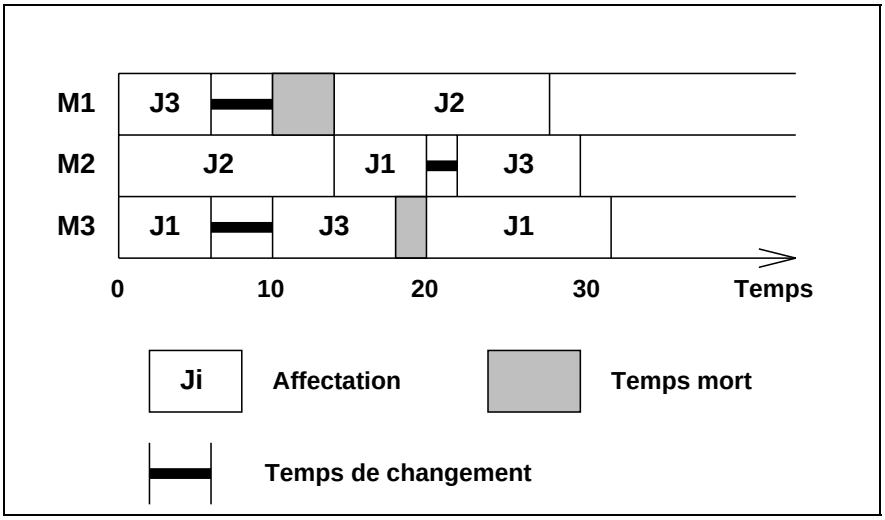


FIG. 1.1 - *Exemple de diagramme de Gantt*

1.2 Classification

Étant donné la diversité des problèmes d'ordonnancement, nous utilisons couramment un formalisme issu des travaux de Conway et al. [32], et Rinnoy Kan [123], permettant de distinguer les problèmes d'ordonnancement entre eux et de les classer. Ce formalisme, repris par Blazewicz et al. [15], comporte trois champs $\alpha/\beta/\gamma$ permettant de décrire les différentes entités d'un problème d'ordonnancement.

1.2.1 Champ α : organisation des ressources

Le champ α est composé de deux sous-champs $\alpha_1\alpha_2$. Les ressources peuvent être parallèles (chaque machine peut exécuter chaque tâche) ou dédiées (spécialisées à l'exécution d'une ou plusieurs tâches). Dans le cas de machines parallèles, elles peuvent être soit identiques et avoir la même vitesse pour toutes les tâches, soit uniformes, chaque machine a alors sa propre vitesse qui ne dépend pas de la tâche exécutée, soit non liées, cas plus général, où la vitesse d'exécution dépend de la machine et de la tâche exécutée.

Le système peut être plus ou moins simple. L'exécution d'une tâche peut être réalisée soit en une seule opération, soit en plusieurs opérations successives exécutées sur des machines différentes. Dans ce dernier cas, plusieurs structures ont été définies : les problèmes de flow-shop, lorsque chaque tâche comporte le même enchaînement d'opérations, les problèmes d'open-shop, où l'ordre des opérations est quelconque et les problèmes de job-shop, où chaque tâche a une gamme d'opérations qui lui est propre.

Le paramètre $\alpha_1 \in \{\emptyset, P, Q, R, F, O, J\}$ représente le type de ressources principales (machines) utilisées :

- $\alpha_1 = \emptyset$ une seule machine. C'est le cas le plus simple qui peut être vu comme un cas particulier des autres configurations.
- $\alpha_1 = P$: machines parallèles identiques. Les ressources sont composées de machines de même cadence, disposées en parallèle pouvant exécuter toutes les tâches.
- $\alpha_1 = Q$: machines parallèles uniformes. Les cadences des machines sont différentes deux à deux, mais restent indépendantes des tâches.
- $\alpha_1 = R$: machines parallèles non liées. Les cadences des machines sont différentes deux à deux et dépendent des tâches exécutées.
- $\alpha_1 = F$: flow-shop. Les tâches sont décomposées en plusieurs opérations qui doivent être exécutées sur l'ensemble des machines, disposées en série, selon un même routage.

$\alpha_1 = O$: open-shop. Les opérations des tâches doivent être exécutées sur certaines machines sans restriction sur le routage des tâches.

$\alpha_1 = J$: job-shop. Les opérations des tâches doivent être exécutées sur l'ensemble des machines disposées en série, mais peuvent avoir des routages différents.

Le paramètre $\alpha_2 \in \{\emptyset, m\}$ dénote le nombre de machines composant le système. Lorsque $\alpha_2 = \emptyset$, le nombre de machines est variable, tandis que lorsque $\alpha_2 = m$, le nombre de machines est égal à m ($m > 0$).

1.2.2 Champ β : contraintes et caractéristiques du système

Ce champ $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8$ décrit les caractéristiques des ressources et des tâches. Pour répondre à des problèmes industriels des plus divers, la variété de ces caractéristiques est très importante.

Tout d'abord, différents modes d'exécution sont possibles : soit avec préemption (l'exécution d'une opération peut être interrompue puis reprise sur la même machine ou sur une autre), soit sans préemption (lorsqu'une opération est commencée, elle doit être terminée avant de pouvoir exécuter une autre opération sur la même machine). Le paramètre $\beta_1 \in \{\emptyset, pmtn\}$ indique cette possibilité de préemption ($\beta_1 = pmtn$). Si $\beta_1 = \emptyset$, la préemption n'est pas autorisée.

Le paramètre $\beta_2 \in \{\emptyset, res\}$ caractérise les ressources supplémentaires nécessaires à l'exécution d'une tâche (outils, ressources de transport). $\beta_2 = \emptyset$ indique qu'il n'y a pas de ressource complémentaire. $\beta_2 = res$ spécifie les ressources complémentaires.

Le paramètre $\beta_3 \in \{\emptyset, prec, tree, chain\}$ reflète les contraintes de précédence entre tâches spécifiant qu'une tâche doit être exécutée avant une autre. Les valeurs $\emptyset, prec, tree, chain$ représentent respectivement des tâches indépendantes, des relations de précédence générales, des relations de précédence particulières sous forme d'arbres ou de chaînes.

Le paramètre $\beta_4 \in \{\emptyset, r_j\}$ décrit les dates d'arrivée (ou dates de disponibilité, ou dates au plus tôt) des différentes tâches dans le système. Ces dates peuvent être identiques et égales à zéro pour toutes les tâches ($\beta_4 = \emptyset$) ou différentes suivant les tâches ($\beta_4 = r_j$).

Le paramètre $\beta_5 \in \{\emptyset, p_j = p, \underline{p} \leq p \leq \bar{p}\}$ détaille les temps d'exécution des différentes tâches. Ces temps peuvent être ou ne pas être fonction de la machine.

Différentes restrictions peuvent également être considérées pour simplifier certains problèmes.

$\beta_5 = \emptyset$: les tâches ont des temps d'exécution arbitraires.

$\beta_5 = p_j = p$: toutes les tâches ont un temps d'exécution égal à p.

$\beta_5 = \underline{p} \leq p \leq \bar{p}$: les temps d'exécution des tâches sont compris entre $\underline{p}$ et $\bar{p}$.

Ensuite, le paramètre $\beta_6 \in \{\emptyset, d_j, \tilde{d}_j\}$ indique les éventuelles dates d'échéance (ou dates au plus tard) des tâches, dates pour lesquelles les tâches doivent être terminées.

$\beta_6 = \emptyset$: les tâches n'ont pas de date d'échéance.

$\beta_6 = d_j$: chaque tâche a une date d'échéance souhaitée. Généralement, un non respect de ces dates entraîne une pénalisation.

$\beta_6 = \tilde{d}_j$: chaque tâche a une date d'échéance impérative (date limite), qu'il faut absolument respecter.

Le paramètre $\beta_7 \in \{\emptyset, s, s_i, s_{ij}, nwt\}$ permet de spécifier des contraintes sur les enchaînements de tâches très souvent introduites pour représenter au mieux les problèmes réels. Il est parfois nécessaire de considérer un temps de changement entre l'exécution de deux tâches différentes sur une même machine pour représenter les changements et réglages d'outils. Ces temps de changement peuvent être constant (s), fonction de la nouvelle tâche (s_i) ou bien fonction de l'enchaînement des deux tâches (s_{ij}). Également, pour le cas des industries où l'on manipule de la matière en fusion, il convient de pouvoir imposer que toutes les opérations d'une tâche soient exécutées sans temps d'attente ($\beta_7 = nwt$ (no-wait)).

Enfin, le paramètre $\beta_8 \in \{\emptyset, M_j\}$ indique, dans le cas de machines parallèles, des restrictions sur la polyvalence des machines. L'ensemble M_j représente l'ensemble des machines capables de réaliser la tâche j. Lorsque β_8 est vide, toutes les machines sont capables d'exécuter toutes les tâches.

1.2.3 Champ γ : critères d'optimisation

Le troisième champ, le champ γ, concerne les critères d'évaluation d'un ordonnancement. Les objectifs visés sont liés à une bonne utilisation des ressources, une minimisation du délai global ou encore le respect d'un maximum de contraintes. Nous donnons ici les critères les plus couramment utilisés.

$\gamma = C_{max}$: makespan. Ce critère vise à minimiser la date de sortie du système de la dernière tâche ($C_{max} = \max_j C_j$)[1].

1. C_j : date de fin d'exécution de la tâche j.

$\gamma = \sum w_j C_j$: somme pondérée des dates de fin d'exécution[2]. Ce critère représente la somme des encours des tâches dans le système.

$\gamma = L_{max}$: décalage temporel maximal. Ce critère mesure la plus grande violation des dates d'échéances souhaitées ($L_{max} = \max_j L_j = \max_j(C_j - d_j)$).

$\gamma = \sum w_j T_j$: somme pondérée des retards ($T_j = \max(C_j - d_j, 0)$), qui permet d'évaluer les pénalités dûes aux retards.

$\gamma = \sum w_j U_j$: somme pondérée du nombre de tâches en retard[3].

1.3 Complexité

L'expérience montre que certains problèmes sont plus faciles que d'autres à résoudre. Une théorie de la complexité a été développée et permet mathématiquement de classer les problèmes faciles et difficiles en deux classes : les classes $\mathcal{P}$ et $\mathcal{NP}$. Nous exposons dans la partie qui suit, les grands principes de la théorie de la complexité. Le lecteur intéressé par de plus amples informations pourra consulter différents ouvrages complètement ou partiellement dédiés à la complexité dont les livres de Garey et Johnson [54], Sakarovitch [128], Xuong [144] ou encore Brucker [19].

1.3.1 Les classes $\mathcal{P}$ et $\mathcal{NP}$

Pour pouvoir exposer la notion de classe de problèmes, il est tout d'abord nécessaire de distinguer les problèmes de décision des problèmes d'optimisation. Un *problème de décision* est un problème pour lequel la réponse est "oui" ou "non". Il est possible d'associer à chaque *problème d'optimisation*, un problème de décision en introduisant un seuil k correspondant à la fonction objectif f. Le problème de décision devient : "existe-t-il un ordonnancement réalisable (S) tel que $f(S) \leq k$?"

Il est alors possible de définir la classe $\mathcal{P}$ qui regroupe les problèmes de décision résolubles par des algorithmes polynomiaux. Un *algorithme polynomial* est définit comme un algorithme dont le temps d'exécution est borné par $O(p(x))$ où p est un polynôme et x est la longueur d'entrée d'une instance du problème. Les algorithmes dont la complexité ne peut pas être bornée polynomialement sont qualifiés d'*exponentiels*.

La classe $\mathcal{NP}$ regroupe les problèmes qui peuvent être résolus en temps polynomial par un algorithme non déterministe[4]. Pour ces algorithmes, si à chaque

2. w_j : poids de la tâche j.
3. $U_j = 1$ si la tâche j est en retard, 0 sinon.
4. Algorithme qui comporte des instructions de choix

instruction, le bon choix est effectué, le temps de calcul est polynomial. Si au contraire tous les choix sont énumérés, l'algorithme devient déterministe et son temps de calcul devient exponentiel.

Les algorithmes "ordinaires" sont évidemment des cas particuliers des algorithmes non déterministes. Aussi tout problème de décision qui peut être résolu par un algorithme polynomial, et qui donc appartient à la classe $\mathcal{P}$, appartient également à la classe $\mathcal{NP}$. D'où $\mathcal{P} \subseteq \mathcal{NP}$.

Parmi les problèmes de la classe $\mathcal{NP}$, une large classe de problèmes, les problèmes $\mathcal{NP}$-*complets*, sont équivalents entre-eux quant à l'existence d'un algorithme polynomial pour les résoudre. C'est à dire, s'il existe un algorithme polynomial pour résoudre un seul de ces problèmes, alors il en existe un pour chaque problème de la classe $\mathcal{NP}$. Afin de décrire cette classe d'équivalence, définissons tout d'abord la *réduction polynomiale* entre deux problèmes. Soient $P1$ et $P2$, deux problèmes de décision. On dit que $P1$ se réduit polynomialement à $P2$ (noté $P1 \propto P2$) s'il existe un algorithme de résolution de $P1$, qui fait appel à un algorithme de résolution de $P2$, et qui est polynomial lorsque la résolution de $P2$ est comptabilisée comme une opération élémentaire [128]. On dit alors d'un problème de décision qu'il est $\mathcal{NP}$-complet si tout problème de la classe $\mathcal{NP}$ se réduit polynomialement à lui. Les problèmes d'optimisation combinatoire dont le problème de décision associé est $\mathcal{NP}$-complet sont qualifiés de $\mathcal{NP}$-*difficiles*.

1.3.2 Prouver la $\mathcal{NP}$-complétude d'un problème

La démonstration d'appartenance d'un problème de reconnaissance à la classe des problèmes $\mathcal{NP}$-complets est très importante puisque, une fois cette démonstration faite, on peut considérer comme peu vraisemblable l'existence d'un algorithme polynomial pour résoudre ce problème. Prouver qu'un problème de décision P est $\mathcal{NP}$-complet consiste en les quatre étapes suivantes :

1 Montrer que P est dans $\mathcal{NP}$.

2 Choisir P', un problème $\mathcal{NP}$-complet connu.

3 Construire une transformation f de $P' \rightarrow P$.

4 Prouver que f est une réduction polynomiale.

Ainsi, il est nécessaire de connaître des problèmes classiques, connus pour être $\mathcal{NP}$-complets. Le premier problème qui a été prouvé comme étant $\mathcal{NP}$-complet par Cook [33], est le problème de *satisfaisabilité* SAT qui peut être défini comme suit : étant donné une expression booléenne de la forme d'un produit de sommes (par exemple), le problème est dit satisfaisable s'il existe une affectation en 0 et

1 de ses variables de manière à ce que l'expression booléenne prenne la valeur 1.

SAT Instance : une expression booléenne E sous forme normale conjonctive
Question : E est-elle satisfaisable?

1.3.3 Quelques problèmes $\mathcal{NP}$-complets

Nous présentons ici quelques autres problèmes $\mathcal{NP}$-complets dont la $\mathcal{NP}$-complétude n'a pu être démontrée que grâce à l'existence d'un premier problème $\mathcal{NP}$-complet. Ce sont les six problèmes de base exposés par Garey et Johnson [54], qui détaillent ensuite la littérature concernant la complexité des problèmes.

3-satisfaisabilité : Étant donné un ensemble de clauses de dimension 3, construites à partir d'un ensemble fini de variables, existe-t-il une affectation de ces variables qui satisfait toutes les clauses?

Couplage de dimension trois : Étant donné un ensemble M de triplets ($M \subseteq X \times Y \times Z$, où X, Y, Z sont disjoints et de même cardinalité q), existe-t-il $M' \subseteq M$ tel que $|M'| = q$ et les triplets de M' sont deux à deux disjoints?

Recouvrement : Étant donné un graphe $G = (V, E)$ et un entier k, existe-t-il $X \subseteq V$ tel que $|X| \leq k$ et toute arête de G a au moins une de ses extrémités dans X?

Clique : Étant donné un graphe $G = (V, E)$ et un entier k, existe-t-il $X \subseteq V$ tel que $|X| \geq k$ et tous les sommets de X sont deux à deux adjacents?

Cycle Hamiltonien : Étant donné un graphe $G = (V, E)$, existe-t-il un cycle passant une fois et une seule par chacun des sommets de V?

Partition : Étant donnés n entiers positifs $s_1, s_2, \ldots, s_n$, existe-t-il un sous ensemble $J \subseteq I = \{1, 2, \ldots, n\}$ tel que $\sum_{i \in J} s_i = \sum_{i \in I \setminus J} s_i$

La figure 1.2 représente les transformations utilisées pour prouver la $\mathcal{NP}$-complétude des six problèmes présentés.

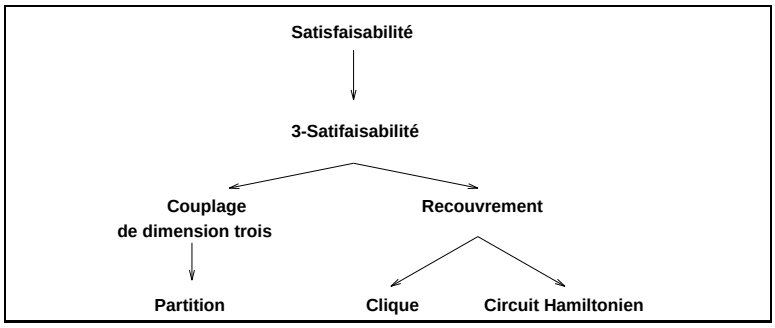


FIG. 1.2 - *Enchaînement des preuves de $\mathcal{NP}$-complétude*

1.3.4 Hiérarchie de complexité entre les problèmes d'ordonnancement

L'étude des relations existantes entre les différents problèmes d'ordonnancement revêt un grand intérêt, dans la mesure où cela permet d'appliquer des algorithmes de résolution connus pour certaines classes de problèmes à d'autres classes qui leurs sont réductibles. Prenons l'exemple de l'ordonnancement sur machine parallèles : il est évident que le problème $P//C_{max}$ est un cas particulier du problème $Q//C_{max}$ où toutes les machines auraient la même vitesse. Ainsi, $P//C_{max}$ est réductible à $Q//C_{max}$, qui lui est plus général et tout algorithme développé pour $Q//C_{max}$ peut être appliqué au problème $P//C_{max}$. Nous rappelons que lorsque le nombre de machines n'est pas précisé, il est considéré comme variable.

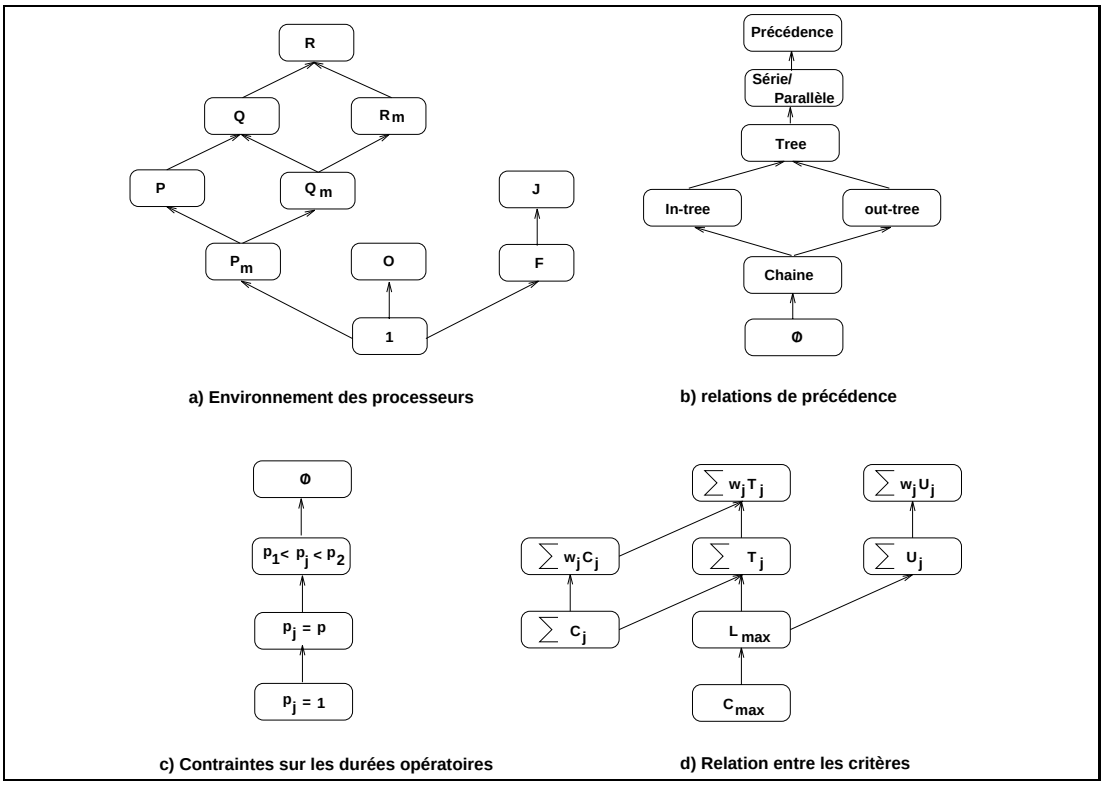


FIG. 1.3 - *Hiérarchie de complexité entre différents problèmes*

Il est intéressant de voir comment la modification d'un seul élément de la classification d'un problème peut affecter sa complexité. Pour cela, nous exposons, figure 1.3, certaines hiérarchies existant entre différents problèmes. Pour une représentation plus complète de ces relations, il est possible de se référer à Pinedo [112] et Blazewicz et al [15].

Le cas a) indique la hiérarchie de compléxité entre des problèmes de configurations

de machines différentes, lorsque l'environnement reste identique. Comme nous le montre la figure, le cas le plus simple est le cas à une machine. Puis le système devient plus complexe en considérant plusieurs machines en parallèle ou en série. La difficulté augmente encore, si le nombre de machines est non borné et si l'on considère, non plus des machines identiques, mais des machines uniformes ou non liées. En ce qui concerne les relations de précédence, le cas b) nous montre clairement les degrés de complexité. La figure c) nous donne l'évolution de la difficulté, en fonction des valeurs des durées opératoires. Enfin, la figure d) donne la relation existant entre les principaux critères d'optimisation.

1.4 Méthodes classiques de résolution

Nous avons vu précédemment qu'il existe des problèmes d'ordonnancement de complexité différente. Les problèmes appartenant à la classe $\mathcal{P}$ ont des algorithmes polynomiaux permettant de les résoudre. Ces algorithmes sont spécifiques au problème à résoudre et nous ne les citerons pas ici. Pour les problèmes appartenant à la classe $\mathcal{NP}$, l'existence d'algorithmes polynomiaux semble peu réaliste. Ainsi, différentes méthodes de résolution (méthodes exactes ou heuristiques), sont largement utilisées pour appréhender les problèmes $\mathcal{NP}$-difficiles. Nous exposons dans cette partie, les grandes lignes des méthodes les plus connues classées en trois catégories : les méthodes exactes, les heuristiques constructives et les méthodes amélioratives. Le lecteur intéressé par de plus amples détails pourra se reporter aux références données.

1.4.1 Méthodes exactes

Dans ce paragraphe, il est question de trois types de méthodes exactes : la programmation dynamique, la méthode de séparation et évaluation et la résolution à partir d'une modélisation analytique. Comme nous le verrons, le temps de calcul de ces méthodes est exponentiel ce qui explique qu'elles ne sont utilisables que sur des problèmes de petites tailles.

1.4.1.1 Programmation dynamique

Introduite par Bellman dans les années 50 [12], *la programmation dynamique* décompose un problème de dimension n en n problèmes de dimension 1. Le système est alors constitué de n étapes que l'on résout séquentiellement, le passage d'une étape à une autre se faisant à partir des lois d'évolution du système et d'une décision [25].
Le principe d'optimalité de Bellman est basé sur l'existence d'une équation récursive permettant de décrire la valeur optimale du critère à une étape en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble

de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille $k-1$ et porte le nombre de sous-ensembles considérés à 2^n (où n est le nombre d'éléments considérés dans le problème). La programmation dynamique est donc de complexité exponentielle. Pourtant pour les problèmes $\mathcal{NP}$-difficiles au sens faible, c'est-à-dire pour lesquels il existe des algorithmes de résolution pseudo-polynomiaux, il est souvent possible de construire un algorithme de programmation dynamique pseudo-polynomial, pouvant alors être utilisé pour des problèmes de tailles raisonnables.

1.4.1.2 Méthode de séparation et évaluation

La méthode de séparation et évaluation est basée sur une énumération implicite et intelligente de l'ensemble des solutions réalisables. Pour cela, la *séparation* consiste à décomposer le problème initial en plusieurs sous-problèmes qui sont à leur tour décomposables. Ce processus peut se visualiser sous forme d'un arbre d'énumération. Pour chaque sous-problème (noeud de l'arbre), la procédure d'*évaluation* calcule (dans le cas d'un problème de minimisation) une borne inférieure de la solution obtenue à partir de ce sous-problème. Au préalable, une borne supérieure de la solution optimale a été calculée et est utilisée pour éviter l'exploration de noeuds dont la valeur de la borne inférieure est supérieure à la valeur de la borne supérieure. Cette borne supérieure est réactualisée lorsqu'une solution réalisable de valeur inférieure est atteinte.
Ainsi, l'exploration de certaines branches de l'arbre est coupée, ce qui permet de ne pas énumérer réellement toutes les solutions. Il faut donc remarquer que l'efficacité d'un algorithme de séparation et d'évaluation est déterminée par la qualité des bornes utilisées et par de bonnes conditions de branchement.

1.4.1.3 Modélisation analytique et résolution

La modélisation analytique d'un problème permet, non seulement de mettre en évidence l'objectif et les différentes contraintes du problème, mais également, parfois de le résoudre. L'idéal est d'obtenir un programme linéaire dont les variables sont réelles. Dans ce cas, il existe bon nombre de solveurs pouvant le résoudre. Dès que le problème comporte des coûts fixes, ou des décisions nécessitant l'utilisation de variables entières, les modèles deviennent plus difficiles à résoudre. Il en est de même lorsque le modèle n'est pas linéaire, mais quadratique par exemple.

Néanmoins, il est parfois surprenant de voir qu'un probleme particulier de taille intéressante peut être appréhendé et résolu par la programmation mathématique. Il est donc justifié de commencer à étudier un problème en proposant une ou plusieurs modélisations analytiques. De plus, cette démarche a été simplifiée car il existe aujourd'hui plusieurs langages de modélisation, tels que AMPL, GAMS, LINGO, MPL, permettant d'écrire les programmes linéaires de façon formelle, proche de l'écriture mathématique. Ces langages de modélisation peuvent soit

générer des fichiers lisibles par des solveurs, tels que CPLEX, OSL, XPRESS, soit être directement couplés à ces solveurs. Malheureusement, tous les problèmes ne peuvent être résolus par cette approche car la résolution de programmes linéaires mixtes, par exemple, demande souvent beaucoup trop de temps de calcul.

1.4.2 Méthodes heuristiques constructives

Pour les problèmes de grandes tailles, les méthodes exactes ne sont pas envisageables de par leur temps de calcul. Il est dans ce cas possible d'utiliser des méthodes approximatives qui donnent des solutions certes sous-optimales, mais obtenues rapidement. Ces solutions peuvent ensuite servir de solution initiale pour les méthodes amélioratrices (voir section 1.4.3). La *performance* de tels algorithmes est généralement calculée par le rapport entre la valeur de la solution calculée par l'heuristique et la valeur de la solution optimale: $R_A(I) = A(I)/OPT(I)$ pour le pire des cas. D'autres analyses de performances peuvent être menées. Par exemple, l'analyse en moyenne regarde le comportement moyen de l'heuristique en calculant $R_A(I)$, non pas seulement pour le pire des cas, mais également pour la moyenne de plusieurs instances. De plus, lorsque la solution optimale n'est pas calculable (parce que les problèmes sont de trop grande taille, par exemple), il est également possible d'étudier expérimentalement le comportement de l'heuristique en comparant ses performances soit avec les performances d'autres heuristiques, soit avec des bornes inférieures de la solution optimale.

Les méthodes par construction progressive sont des méthodes itératives où à chaque itération, une solution partielle est complétée. La plupart de ces méthodes sont des *algorithmes gloutons* car elle considèrent les éléments (processeurs ou tâches, suivant les cas) dans un certain ordre sans jamais remettre en question un choix, une fois qu'il a été effectué. De principe très simple, ces algorithmes permettent de trouver une solution très rapidement. Parmi ces méthodes nous en exposons deux types.
Regardons tout d'abord, les *algorithmes de liste*. Ces algorithmes consistent, dans une première phase, à calculer une liste qui donnera l'ordre de prise en compte des éléments. Cette liste construite a priori, à partir d'un critère bien défini, n'est pas remise en cause au cours de l'ordonnancement. La deuxième phase de l'algorithme se réduit à considérer les éléments (processeurs ou tâches) dans l'ordre de la liste pour construire l'ordonnancement.
Un deuxième type de méthode par construction progressive, consiste à choisir, au cours de la construction, l'affectation d'une tâche sur une machine en utilisant des *règles de priorité*. Ces méthodes peuvent également être vues comme des méthodes sérielles dynamiques où les listes sont reconstruites à chaque étape, selon un critère qui peut évoluer dans le temps (le nombre de tâches disponibles, par exemple). Notons que si le choix de la tâche à ajouter est guidée par l'application d'un théorème de dominance, qui permet d'établir des résultats concernant

l'existence de solutions optimales si certaines hypothèses sont vérifiées [112], la méthode peut donner la solution optimale. Une revue détaillée des règles classiques couramment utilisées en ordonnancement est proposée par Panwalkar et Iskander [108]. Pour chaque problème, de nombreuses règles ont été développées et les articles présentant ces règles sont très nombreux et spécifiques à un problème donné. Nous ne détaillerons donc pas la littérature sur ce point.
A partir de ces schémas de construction, des heuristiques plus ou moins complexes ont été développées pour répondre à des problèmes spécifiques. Nous en exposerons quelques-unes, dans la revue de la littérature sur les problèmes d'ordonnancement sur machines parallèles (voir chapitre 2).

1.4.3 Méthodes amélioratrices

Nous exposons, dans ce paragraphe, les méthodes d'améliorations locales les plus connues. Ces méthodes sont initialisées par une solution réalisable, calculée soit aléatoirement soit à l'aide d'une des heuristiques constructives exposées précédemment, et recherchent à chaque itération une amélioration de la solution courante par des modifications locales. Cet examen se poursuit jusqu'à ce qu'un critère d'arrêt soit satisfait. L'utilisation de ces heuristiques itératives suppose que l'on puisse définir pour toute solution S, un voisinage de solution, $N(S)$, contenant les solutions voisines (proches dans un certain sens). En général, le voisinage d'une solution est généré en appliquant, plusieurs fois et de façon différente, à cette solution, une petite transformation (échanges, par exemple). A chaque solution S, nous associons le coût de cette solution, $c(S)$.

1.4.3.1 Méthode de descente

C'est l'une des heuristiques de recherche locale les plus simples. Elle consiste à rechercher dans le voisinage de la solution courante, une solution de coût plus faible. Elle procède ainsi jusqu'à arriver à un optimum local.
Cette méthode a l'avantage d'être rapide, mais s'arrête dès qu'un optimum local est atteint, même si celui-ci n'est pas de bonne qualité.

Parmi ces méthodes, nous décrivons ici les *méthodes d'échanges* de type r-opt.
Ces méthodes d'optimisation ont été initialement proposées par Lin [96] pour résoudre le Problème du Voyageur de Commerce (PVC), mais elles s'appliquent également à tout problème combinatoire dont la solution consiste en une permutation de r composantes parmi n.
Le terme r-optimal indique qu'une solution ne peut plus être améliorée en échangeant r éléments. La méthode consiste donc à sélectionner r composantes et à regarder si en interchangeant ces composantes, on obtient une meilleure solution. Nous remarquons donc qu'une solution n-optimale est une solution optimale (dans le cas d'un problème de taille n). Ainsi, plus r augmente, plus on se rapproche de

la solution optimale, mais plus les calculs sont difficiles. En effet, il y a $\binom{n}{r}$ façons de choisir r composantes et $r!$ façons de les échanger. En pratique, on se limite à $r = 2$ ou 3.

1.4.3.2 Recuit simulé

Cette classe de méthodes d'optimisation est dûe aux physiciens Kirkpatrick, Gelatt et Vecchi [84]. Elle s'inspire des méthodes de simulation de Metropolis (année 50) en mécanique statistique.
L'analogie historique s'inspire du recuit des métaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalent à un optimum global.
La méthode du recuit simulé, appliquée aux problèmes d'optimisation, considère une solution initiale et recherche dans son voisinage une autre solution pouvant devenir solution courante. L'originalité de cette méthode est qu'il est possible de se diriger vers une solution voisine de moins bonne qualité avec une probabilité non nulle. Ceci permet d'échapper aux optima locaux. Au début de l'algorithme, un paramètre T, apparenté à la température, est déterminé et décroit tout au long de l'algorithme pour tendre vers 0. De la valeur de ce paramètre va dépendre les probabilités d'acceptation des solutions détériorantes.
La performance du recuit simulé dépend de la règle de refroidissement (c'est à dire la décroissance du paramètre T) que l'on utilise. Un refroidissement trop rapide mènerait vers un optimum local pouvant être de très mauvaise qualité. Un refroidissement trop lent serait très coûteux en temps de calcul. Le réglage des différents paramètres (température initiale, nombre d'itérations par palier de température, décroissance de la température, ...) peut donc être long et difficile.

L'intérêt des cette classe de méthode est qu'il existe une preuve de la convergence, ainsi, lorsque certaines conditions sont vérifiées, on a la garantie d'obtenir la solution optimale.

1.4.3.3 Recherche tabou

Ces méthodes dont l'origine peut remonter à 1977 [58], ont été formalisées plus tard, en 1986, par Glover [59]. Elles n'ont aucun caractère stochastique et utilise la notion de mémoire pour éviter de tomber dans un optimum local.
Le principe de l'algorithme est le suivant : à chaque itération le voisinage (complet ou un sous-ensemble du voisinage) de la solution courante est examiné et la solution minimisant l'augmentation de coût est sélectionnée. Pour éviter les phénomènes de cyclage, la méthode a l'interdiction de revisiter une solution récemment visitée. Pour cela, une liste taboue contenant les dernières solutions visitées est tenue à jour. Chaque nouvelle solution considérée chasse de cette liste la solution

la plus anciennement visitée. La longueur de la liste, paramètre à définir, détermine donc le nombre d'itérations pendant lesquelles une solution ayant été visitée ne pourra être reconsidérée. Ainsi, la recherche de la solution courante suivante se fait dans le voisinage de la solution courante actuelle sans considérer les solutions appartenant à la liste taboue. Tout au long de l'algorithme, la meilleure solution doit être conservée car l'arrêt se fait rarement sur la meilleure solution. En effet, l'arrêt de cette méthode peut se faire soit après un certain nombre d'itérations, soit lorsqu'il n'y a pas eu d'amélioration de la meilleure solution depuis un certain nombre d'itérations. Des versions plus élaborées permettant des recherches plus efficaces, ont été proposées par la suite [60], [61].
Cette méthode donne de très bons résultats pratiques, malgré l'inexistence de résultats théoriques garantissant la convergence de l'algorithme vers la solution optimale.

1.4.3.4 Algorithmes génétiques

Cette classe de méthodes est basée sur une imitation des phénomènes d'adaptation des êtres vivants. L'application de ces méthodes aux problèmes d'optimisation a été formalisée par Goldberg en 1989 [62].
Les algorithmes génétiques fonctionnent sur une analogie avec la reproduction des êtres vivants. On part d'une population (ensemble de solutions) initiale sur laquelle des opérations de reproduction, de croisement ou de mutation vont être réalisées dans l'objectif d'exploiter au mieux les caractéristiques et propriétés de cette population. Ces opérations doivent mener à une amélioration (en terme de qualité des solutions) de l'ensemble de la population puisque les bonnes solutions sont encouragées à échanger par croisement leurs caractéristiques et à engendrer des solutions encore meilleures. Toutefois, des solutions de très mauvaises qualités peuvent aussi apparaître et permettent d'éviter de tomber trop rapidement dans un optimum local.
Les difficultés pour appliquer les algorithmes génétiques résident dans le besoin de coder les solutions, et dans les nombreux paramètres à fixer (taille de la population, coefficients de reproduction, probabilité de mutation, ...). De plus, ces algorithmes demandent un effort de calcul très important.

Chapitre 2

Ordonnancement sur machines parallèles : état de l'art

De nombreux états de l'art dans le domaine de l'ordonnancement comme l'ouvrage de Lawler et al [92], du Gotha [65], de Pinedo [112] ou encore de Blazewicz et al [15] consacrent une partie importante à l'ordonnancement sur machines parallèles. Nous citons également les articles de Cheng et Sin [24], de Guinet et al. [70] et la thèse de Perraud-Echalier [111] qui présentent un état de l'existant intéressant sur le problème.
Ce chapitre est consacré à un état de l'art sur la complexité des problèmes de machines parallèles et les méthodes existantes. Le chapitre est divisé en quatre parties. La première traite des problèmes d'ordonnancement sur machines parallèles avec minimisation de la durée totale d'exécution (C_{max}). La deuxième partie expose les résultats sur les problèmes avec minimisation de la somme des dates de fins d'exécution ($\sum C_i$). Enfin la troisième partie est consacrée aux problèmes d'ordonnancement sur machines parallèles avec temps de changement dépendant de la séquence (s_{ij}), et à leur analogie avec le Problème de Tournées de Véhicules (PTV). Bien sûr, étant donné le très grand nombre de travaux réalisés dans ce domaine, nous n'avons pas pu avoir connaissance de tout et il peut y avoir des oublis. Nous avons malgrè tout, essayé d'en rapporter un grand nombre. Il est, cependant, à noter que nous n'avons pas du tout étudié les problèmes mettant en cause des dates d'arrivée ou des dates d'échéance de tâches.
A l'issue de chaque partie, un tableau récapitulatif résume les principaux résultats de complexité des problèmes. Seuls les problèmes les plus durs résolubles polynomialement et les problèmes $\mathcal{NP}$-difficiles les plus faciles sont donnés. Les problèmes ouverts sont également indiqués.

Avant de commencer, nous rappelons la hiérarchie de complexité existant entre les différentes configurations de machines (ou processeurs). La configuration la plus simple est composée de machines identiques, où toutes les machines peuvent

exécuter toutes les tâches. Une tâche i a alors une durée opératoire p_i quelque soit la machine qui l'exécute. Ensuite vient le cas des machines uniformes où les machines sont polyvalentes mais ont des vitesses différentes. Ainsi, la durée d'exécution de la tâche i sur la machine k est $p_{ik} = p_i/s_k$, où s_k est la vitesse de la machine k. Enfin, il existe un cas plus difficile, où les machines sont non liées et la durée d'exécution p_{ik} d'une tâche i dépend à la fois de la tâche et de la machine k sur laquelle elle est exécutée. Nous remarquons qu'il est alors possible de spécifier qu'une machine ne peut exécuter certaines tâches en leur donnant des durées opératoires infinies. Comme nous le verrons au cours de ce chapitre, dans la plupart des cas, considérer des machines non liées complexifie fortement les problèmes. Pourtant l'étude de cette configuration est justifiée car elle permet de rester proche des réalités industrielles, où il existe souvent plusieurs machines pouvant réaliser une même opération avec des temps et des coûts différents en fonction de la génération technologique de la machine, de ses spécificités propres et de ses outils disponibles.

2.1 Minimisation du C_{max}

2.1.1 Processeurs identiques

2.1.1.1 $P//C_{max}$

Le problème consistant à ordonnancer un ensemble de tâches sur un ensemble de machines parallèles identiques, sans contraintes supplémentaires, tout en cherchant à minimiser la durée totale de production, est $\mathcal{NP}$-difficile puisque même le problème restreint à deux machines ($P2//C_{max}$) a été démontré comme étant $\mathcal{NP}$-difficile par une réduction du problème de bi-partitionnement par Karp [81].

Ainsi, puisqu'il n'est pas possible de trouver un algorithme polynomial permettant de résoudre ce problème, des heuristiques de résolution ont été proposées.

Une classe de ces heuristiques est composée d'algorithmes de liste, où les tâches sont affectées aux machines en suivant l'ordre d'une liste construite à partir d'un certain critère (voir section 1.4.2). Pour plus de précisions sur ces algorithmes appliqués aux problèmes de machines parallèles, voir, [66]. Ces algorithmes, appliqués au problème $P//C_{max}$ ont des performances[1] égales à $R_{Liste} = 2 - 1/m$. L'heuristique LPT (Longest Processing time) est la plus connue pour le problème $P//C_{max}$. Elle classe les tâches par ordre décroissant des durées opératoires. De complexité $O(nlogn)$, cette heuristique a une performance de $R_{LPT} = 4/3 - 1/3m$. Plus tard, Ibarra et Kim [78] ont prouvé que $R_{LPT} = 1 + 2(m-1)/n$ pour $n \geq 2(m-1)\pi$ et $\pi = max_i(p_i)/min_i(p_i)$. D'autres algorithmes de listes ont été

1. Performance d'une heuristique : $R_H = Z^H/Z^*$ où, Z^H est la valeur du critère de la solution donnée par l'heuristique et Z^* est la valeur du critère pour une solution optimale

développés. Le lecteur intéressé par ce type d'algorithmes peut se référencer à Lawler et al [92].
Nous citons également l'algorithme MULTIFIT issu de l'analogie entre le problème $P//C_{max}$ et le problème de sac à dos (Coffman et al. [29]). Cette heuristique détermine une valeur C comme capacité maximale des machines puis considère les machines une à une et leur affecte des tâches en suivant la règle FFD (First Fit Decreasing). FFD affecte la tâche la plus longue respectant la capacité restante de la machine. L'algorithme vérifie la faisabilité de la solution trouvée pour la capacité C donnée, et recherche la valeur de C minimale pour laquelle une solution réalisable existe.
D'autre part, un algorithme de programmation dynamique a également été proposé pour $P//C_{max}$ par Rothkopf [125]. Il permet de trouver la solution optimale en un temps $O(nC^m)$ et ne peut donc être utilisé que pour de petites valeurs de m et de C (borne supérieure du C_{max}).

2.1.1.2 $P/pmtn/C_{max}$

En relaxant la contrainte de non-préemption, le problème devient facile et peut être résolu optimalement en un temps polynomial, en appliquant la méthode de Mc Naughton [100]. Elle consiste à calculer la durée totale de production optimale $C^*_{max} = max\{\max_i(p_i), \frac{1}{m}\sum_i p_i\}$, puis d'ordonnancer les tâches une à une en complétant chaque machine jusqu'à C^*_{max}. La complexité de cet algorithme est donc $O(n)$.

2.1.1.3 $P/prec/C_{max}$

Considérons maintenant le cas où il existe des contraintes de précédence entre les tâches. Ullman [139] a tout d'abord prouvé que la restriction de ce problème aux tâches unitaires était $\mathcal{NP}$-difficile. Plus tard, une réduction du problème de clique a permis de montrer que le seul problème de décision consistant à déterminer s'il existe un ordonnancement de longueur maximale égale à trois est $\mathcal{NP}$-complet [92]. L'utilisation des algorithmes de listes, comme pour le problème $P//C_{max}$, peut générer des ordonnancements de comportement non attendus. En effet, les anomalies découvertes par Graham [66], montre que la longueur d'un ordonnancement construit à partir d'algorithmes de listes peut augmenter si par exemple, le nombre de processeurs augmente, si des durées opératoires diminuent, si des contraintes de précédence sont relaxées ou encore si la liste de priorité change.

Il existe cependant des cas particuliers pour lesquels une résolution polynomiale est envisageable. En effet, dans le cas $P/in-tree, p_i = 1/C_{max}$ (ou $P/out-tree, p_i = 1/C_{max}$), Hu [77] a proposé un algorithme polynomial basé sur la notion de niveau de tâches. De même, le cas $P2/prec, p_i = 1/C_{max}$ est un autre cas particulier pour lequel Coffman et Graham [30] ont proposé un algorithme en $O(n^2)$ utilisant l'algorithme de Hu.

2.1.1.4 $P/pmtn, prec/C_{max}$

Considérons maintenant le problème de l'ordonnancement sur machines parallèles identiques avec des contraintes de précédence entre les tâches et l'autorisation de préemption. Dans le cas général, ce problème est $\mathcal{NP}$-difficile [139]. Néanmoins, deux cas particuliers ($P2/pmtn, prec/C_{max}$, $P/pmtn, forest/C_{max}$) sont résolus par l'algorithme de Muntz et Coffman [103] basé sur une réduction des chemins critiques en partageant les processeurs entre les tâches de plus haut niveau. Le niveau d'une tâche i étant composé du temps nécessaire pour compléter la tâche i, plus la somme des temps d'exécution des tâches situées sur le plus long chemin entre i et une tâche terminale. Ainsi, le niveau d'une tâche en cours d'exécution décroit.

2.1.2 Processeurs uniformes

2.1.2.1 $Q//C_{max}$

Le problème de la minimisation de la durée totale d'exécution d'un ensemble de tâches sur un ensemble de processeurs identiques étant déjà $\mathcal{NP}$-difficile, dans le cas de durées opératoires arbitraires, il en est de même pour $Q//C_{max}$. Diverses heuristiques ont alors été présentées pour ce problème. L'une d'entre elles, classe les tâches par ordre décroissant de leur durée d'exécution et les machines par ordre décroissant de leur vitesse. L'affectation des tâches aux machines se fait en suivant les listes, dans cet ordre. Dès qu'une machine se libère, la première tâche non affectée de la liste, lui est attribuée.

Dans le cas de durée unitaires ($Q/p_i = 1/C_{max}$), Blazewicz et al. [15] exposent deux approches polynomiales de résolution. La première consiste en une formulation du problème en un problème de transport solvable en $O(n^3)$ [67]. La deuxième, en $O(m^2)$, est basée sur l'existence d'une borne inférieure $C' = n/\sum_{k=1}^{m} s_k$. Les tâches sont affectées une à une aux machines, tant que cela est possible, de façon à ne pas dépasser C'. Les tâches restantes (au plus m) sont affectées une à une à la machine minimisant leur date de fin.

2.1.2.2 $Q/pmtn/C_{max}$

Comme pour le cas des machines indentiques, autoriser la préemption rend le problème plus facile et solvable en un temps polynomial par la règle LRPT-FM (Longest Remaining Processing Time on the Fastest Machine first) [112] qui permet également de résoudre le problème $Q/pmtn, r_j/C_{max}$. Il est a noté que Horvath et al. [76] ont proposé un algorithme à partir de l'algorithme de Muntz et Coffman, développé pour le problème $P/pmtn, tree/C_{max}$, dans laquelle chaque processeur est partagé équitablement entre les tâches de même niveau. Cet algorithme permet également de résoudre le problème $Q2/pmtn, prec/C_{max}$.

2.1.3 Processeurs non liés

Le cas des processeurs non liés étant celui qui nous intéresse, nous allons, dans cette partie, faire une revue un peu plus détaillée des résultats et algorithmes existants pour cette configuration.

2.1.3.1 $R//C_{max}$

Pour ce problème $\mathcal{NP}$-difficile ($P2//C_{max}$ étant déjà $\mathcal{NP}$-difficile), diverses heuristiques ont été proposées. Nous en présentons ici quelques-unes dans l'ordre chronologique.

Tout d'abord, en 1976, Horowitz et Sahni [75] développent une résolution par programmation dynamique, en $O(min(nF, 2^n))$ du problème $R2//C_{max}$ (où F est une borne supérieure du $Cmax$). Basé sur cet algorithme, ils proposent une heuristique pour résoudre $R//C_{max}$ en un temps $O(n^{2m}/\epsilon)$ où ϵ est l'erreur relative.

Puis, Ibarra et Kim exposent cinq heuristiques en $O(n)$, $O(nlogn)$ ou $O(n^2)$ basées sur une affectation au mieux des tâches considérées dans un certain ordre. La performance de ces algorithmes est égale, au pire, à m fois l'optimum et cette borne est atteinte pour quatre de ces algorithmes. L'heuristique la plus prometteuse, pour laquelle il n'a pas été prouvé que la borne pouvait être atteinte, est l'ECT (Earliest Completion Time) qui choisit à tout moment, parmi les tâches non encore affectées, celle qui pourra être achevée en premier.

Plus tard, Davis et Jaffe [35] montrent que $C_{max}^{ECT}/C_{max}^{*} = 1 + log_2 m$. Ils proposent également différents algorithmes qui n'affectent sur un processeur que des tâches pour lesquelles le processeur est relativement efficace. L'un de ces algorithmes, polynomial en $O(mnlogn)$, est de performance garantie inférieure à $2,5\sqrt{m}$. Un autre algorithme, exponentiel en $O(m^m + mnlogn)$ a une performance de $1,5\sqrt{m}$.

Plus tard, en 1985, Potts [114], présente l'heuristique deux phases LPE (Linear Programming and Enumeration). La première phase résout une relaxation du programme linéaire associé au problème admettant des tâches partagées entre plusieurs processeurs. Il montre qu'il y a au plus $m-1$ tâches fractionnées et la deuxième phase consiste en l'affectation par énumération de ces tâches restantes. Cet algorithme de complexité $O(m^{m-1})$ (pour l'énumération) a un ratio de performance garantie égale à 2. Pour réduire la complexité de l'algorithme, Potts propose d'utiliser une heuristique capable d'ordonnancer $m-1$ tâches en un temps polynomial à la place de l'énumération. Pour le cas de deux machines, la performance de l'algorithme, alors en $O(n)$, est de $(1+\sqrt{5})/2$ et atteint $3/2$ après une petite modification.

Par la suite, des résultats de complexité intéressants sont présentés par Lenstra et al. [95] qui montrent qu'aucun algorithme polynomial ne peut avoir une performance meilleure que 3/2 à moins que $\mathcal{P}$ ne soit égal à $\mathcal{NP}$. Ils présentent parallèlement un algorithme polynomial de performance égale à 2.

Des méthodes de recherches locales, utilisant comme point de départ une solution donnée par une heuristique de descente, sont également appliquées à ce problème par Glass et al. [57]. Leurs expérimentations montrent que les algorithmes génétiques et la méthode de descente donne de moins bons résultats que les autres algorithmes testés. La méthode de descente reste pourtant intéressante de par sa rapidité d'exécution. Par contre, la recherche taboue, le recuit simulé et les algorithmes génétiques couplés avec un algorithme de descente, sont des méthodes très compétitives de déviation moyenne inférieure à 10% sur les exemples donnés.

Enfin, récemment, Martello, Soumis et Toth [99] proposent des bornes pour le problème $R//Cmax$ basées sur la relaxation lagrangienne. Utilisant ces bornes, ils présentent des algorithmes exactes et d'approximation. L'une de leurs heuristiques, TARGET, est très intéressante, car elle cherche à minimiser la durée totale de production en se fixant une borne à essayer de respecter. Au cours de la recherche de l'ordonnancement, si une machine dépasse cette borne, la durée de production de cette machine devient la nouvelle borne. L'affectation des tâches aux machines se fait en fonction d'un calcul de score qui favorise les tâches, qui mal placées, engendredaient un dépassement de la borne. Nous étudierons plus en détail, au chapitre 3, cette heuristique et son adaptation à notre problème.

2.1.3.2 $R/pmtn/C_{max}$

L'autorisation de préemption permet une nouvelle fois de casser la complexité du problème qui devient facile. Lawler et Labetoulle [91] propose un algorithme deux-phases qui dans un premier temps résout un programme linéaire calculant la valeur optimale du C_{max} et la part des tâches affectées à chaque processeur. La deuxième phase utilise cette solution pour construire un ordonnancement.

2.1.4 Tableau récapitulatif pour C_{max}

Le tableau suivant résume les principaux résultats de complexité pour les problèmes de machines parallèles avec minimisation de la durée totale de production.

Problème	Complexité	Références
$P2/p_i = 1, prec/C_{max}$	P	Coffman et Graham [30]
$P/p_i = 1, tree/C_{max}$	P	Hu [77]
$P/pmtn, tree/C_{max}$	P	Muntz et Coffman [103]
$Q2/pmtn, prec/C_{max}$	P	Horvath, Lam et Sethi [76]
$Q/p_i = 1/C_{max}$	P	Graham et al. [67]
$R/pmtn/C_{max}$	P	Lawler et Labetoulle [91]
$P2//C_{max}$	NP	Karp [81]
$P/p_i = 1, prec/C_{max}$	NP	Ullman [139]
$P/p_i = 1, pmtn, prec/C_{max}$	NP	Ullman [139]

TAB. 2.1 - *Résultats majeurs pour C_{max}*

2.2 Minimisation de la somme des dates de fin d'exécution ($\sum C_i$)

2.2.1 Sans préemption

Horn [74] et Bruno et al [20] ont formulé le problème $R//\sum C_i$ en un programme linéaire en nombres entiers. La structure de ce programme est telle que le problème est solvable en un temps polynomial. $R//\sum C_i$ étant polynomial, il en est de même pour $P//\sum C_i$ et $Q//\sum C_i$ puisqu'il est possible de trouver des réductions polynomiales de ces problèmes vers $R//\sum C_i$, et la méthode utilisée pour $R//\sum C_i$ est utilisable pour $P//\sum C_i$ et $Q//\sum C_i$. Néanmoins, des méthodes plus simples ont été développées pour les machines parallèles identiques et uniformes. Par contre, dès que l'on associe des poids aux tâches (correspondants à une priorité, une urgence,...) le problème devient rapidement $\mathcal{NP}$-difficile. En effet, Bruno et al. [20] ont mis en évidence une réduction du problème du sac à dos (connu pour être $\mathcal{NP}$-complet) au problème de décision associé au problème $P2//\sum w_i C_i$.

2.2.1.1 $P//\sum C_i$

Considérons la recherche d'un ordonnancement d'un ensemble de tâches sur un ensemble de machines parallèles identiques minimisant la somme des dates de fin d'exécution des tâches. La généralisation de l'algorithme SPT (Shortest Processing Time), développé pour le problème $1//\sum C_i$, est optimal pour le cas

de plusieurs machines en parallèle [32]. Cet algorithme de liste classe les tâches par ordre croissant de leur durée d'exécution et les affecte, dans cet ordre, à la première machine libre. La complexité de cet algorithme est $O(nlogn)$ pour la construction de la liste.

2.2.1.2 $P/prec/\sum C_i$

Lorsque l'on considère des tâches ayant des contraintes de précédence, le problème devient rapidement difficile. En effet, comme l'indiquent Lenstra et Rinnoy Kan [94], le problème restreint à deux machines avec des durées opératoires unitaires ($P2/prec, p_i = 1/\sum C_i$) est $\mathcal{NP}$-difficile. Seul le cas $P/out-tree, p_i = 1/\sum C_i$ peut être résolu polynomialement par une adaptation de l'algorithme de Hu développé pour $P/out-tree, p_i = 1/C_{max}$.

2.2.1.3 $Q//\sum C_i$

Lorsque l'on se place dans un système de processeurs uniformes, il faut considérer leurs vitesses d'exécution et il n'est donc pas possible d'appliquer directement l'algorithme SPT. Horowitz et Sahni [75] propose un algorithme polynomial dont ils prouvent l'optimalité. Cet algorithme affecte les tâches dans l'ordre croissant de leurs durées opératoires au processeur dont le flot va être le moins augmenté.

2.2.1.4 $R//\sum C_i$

Considérons maintenant le cas de processeurs non liés. La formulation de ce problème en un problème d'affectation, proposée par Horn [74], est basée sur le constat suivant : une tâche i placée en dernière position sur un processeur m contribue pour une valeur p_{im} à la fonction objectif, l'avant dernière tâche j du processeur contribue pour $2\times p_{jm}$, etc. Ainsi, pour résoudre le problème $R//\sum C_i$, il faut affecter les tâches i aux positions k sur les machines m avec un coût $k\times p_{im}$. Bruno et al. [20] transforme ce problème en un problème de transport dont la résolution, en $O(max(mn^2, n^3))$, consiste en la recherche d'un flot maximal de coût minimal.

2.2.2 Avec préemption

2.2.2.1 $P/pmtn/\sum C_i$

Mac Naughton [100] ayant montré que la préemption n'apportait rien pour la minimisation du flot moyen sur machines parallèles identiques, la résolution du problème $P/pmtn/\sum C_i$ est la même que pour $P//\sum C_i$. Dans le cas du flot pondéré, Brucker [19] montre que pour le problème $P/pmtn/\sum w_iC_i$, il existe un ordonnancement optimal non préemptif. Le problème $P2//\sum w_iC_i$ étant $\mathcal{NP}$-difficile, il en est alors de même pour $P2/pmtn/\sum w_iC_i$.

2.2.2.2 $Q/pmtn/\sum C_i$

Dans la cas d'un ordonnancement préemptif sur des machines uniformes, il est possible de montrer qu'il existe un ordonnancement optimal dans lequel $C_i \leq C_j$ si $p_i < p_j$ [85]. Ainsi, un ordonnancement optimal peut être obtenu en utilisant l'adaptation suivante de la règle SPT : les tâches sont ordonnées par ordre croissant de leurs durées opératoires et ordonnancées préemptivement, dans cet ordre, de façon à minimiser leur date de fin. Ainsi, la tâche i_1 de plus petite durée opératoire est affectée à la machine la plus rapide, la deuxième plus courte tâche i_2 étant affectée à la deuxième machine la plus rapide. Dès que la tâche i_1 est terminée, la tâche i_2 est préemptée et mise sur la première machine, et ainsi de suite. La complexité de cet algorithme est en $O(nlogn + mn)$ pour le calcul de la liste et l'affectation.

2.2.2.3 $R/pmtn/\sum C_i$

Le problème de la minimisation du flot moyen sur des machines parallèles non liées est un problème ouvert quant à sa complexité. Il n'existe pas, à notre connaissance, d'algorithme proposé pour résoudre ce type de problème, même dans le cas de deux machines.

2.2.3 Tableau récapitulatif pour $\sum C_i$

Le tableau suivant résume les principaux résultats de complexité.

Problème	Complexité	Références
$R//\sum C_i$	$\mathcal{P}$	Horn [74], Bruno et al. [20]
$Q/pmtn/\sum C_i$	$\mathcal{P}$	Labetoulle et al. [85]
$P2/out-tree, p_i = 1/\sum C_i$	$\mathcal{P}$	Hu [77]
$P2/prec, p_i = 1/\sum C_i$	$\mathcal{NP}$	Lenstra et Rinnoy Kan [94]
$P2//\sum w_i C_i$	$\mathcal{NP}$	Bruno et al. [20]
$P2/pmtn/\sum w_i C_i$	$\mathcal{NP}$	Mac Naughton [100]
$R2/pmtn/\sum C_i$	Ouvert	

TAB. 2.2 - *Résultats majeurs pour* $\sum C_i$

2.3 Problèmes avec temps de changement

Nous considérons dans cette partie les problèmes où le temps nécessaire pour passer de l'exécution d'une tâche à une autre dépend de l'enchaînement de ces tâches. Ce type de problème est très souvent rencontré dans l'industrie où un même équipement réalise plusieurs types de tâches.

Ainsi, dans le problème posé par la société Pechiney, la modification d'une taille de boîtes sur une ligne peut entrainer un arrêt de la ligne, pour cause de réglage, d'une durée pouvant aller jusqu'à six jours ce qui pénalise très fortement la production. Un autre exemple provient de l'industrie chimique où le temps nécessaire pour nettoyer les cuves avant de fabriquer un nouveau produit, dépend de ce qui a été fabriqué précédemment. Une situation similaire apparaît dans l'industrie plastique avec la fabrication d'objets de différentes couleurs. Lors du passage de la production d'un objet d'une couleur à un objet d'une autre couleur, une quantité de plastique est jetée jusqu'à ce que la bonne couleur soit atteinte. Dans ce cas, la perte induite par le changement concerne non seulement une perte de temps disponible à la production, mais également une perte de matière.

Dans la littérature, ces problèmes sont appelés problèmes d'ordonnancement avec temps de changement dépendant de la séquence et sont caractérisés par la présence d'un s_{ij}, représentant le temps de changement nécessaire pour passer de la tâche i à la tâche j sur la même machine, dans le champ β de la classification exposée paragraphe 1.2. Divers articles traitent de problèmes considérant deux types de changements : les changements majeurs (très longs) entre des tâches de familles différentes et les changements mineurs (moins longs) entre des tâches de types différents appartenant à une même famille. Nous notons, dans ces cas là, S_{ij} (avec une majuscule) les changements majeurs et s_{ij} (avec une minuscule) les changements mineurs. De même, certains auteurs s'intéressent à la minimisation des coûts de changement de production que nous notons Cs_{ij}.

Récemment, C. Artigues a présenté une thèse traitant de la prise en compte de temps de préparation dans les problèmes d'ordonnancement [3] et dans laquelle un état de l'art présente différents modèles et approches développées pour traiter des problèmes d'ordonnancement avec temps de préparation. Il propose une nouvelle approche basée sur les graphes et une procédure d'insertion permettant de rajouter dynamiquement une tâche dans un ordonnancement de façon à minimiser l'augmentation du retard maximal [4], [5].

Ce paragraphe est divisé en deux parties. Tout d'abord, nous exposons, l'analogie entre les problèmes d'ordonnancement sur machines parallèles avec temps de changement dépendant de la séquence et le très classique Problème de Tournées de Véhicules (P.T.V.). Puis, dans un deuxième temps, nous relatons les

résultats importants sur l'ordonnancement de machines parallèles avec temps de changement dépendant de la séquence.

2.3.1 Analogie avec le Problème de Tournées de Véhicules

Le problème de la minimisation de la durée totale de production sur une machine unique avec temps de changement de production dépendant de la séquence $(1/s_{ij}/C_{max})$ et le problème de la minimisation des temps de changement $(1/s_{ij}/\sum s_{ij})$ sont équivalents et peuvent, comme l'indique Baker [7], être interprétés comme des Problèmes de Voyageur de Commerce (P.V.C.), problème $\mathcal{NP}$-difficile. L'approche la plus populaire utilisée pour traiter le Problème du Voyageur de Commerce, est l'approche par séparation et évaluation [71]. Un résumé de ces approches est proposé par Balas et Toth [8]. De leur côté, Golden et Stewart présentent un résumé des méthodes de construction et d'amélioration développées pour le P.V.C. [63].

Pour le cas des machines parallèles, le problème de la minimisation de la durée total de production $(P/s_{ij}/C_{max})$ et le problème de la minimisation des temps de changements $(P/s_{ij}/\sum s_{ij})$ ne sont plus équivalents. Une analogie de ces problèmes peut tout de même être faite avec des variantes différentes du Problème de Tournées de Véhicules (que nous notons P.T.V.). Dans ce problème, il ne s'agit plus d'un seul voyageur visitant un ensemble de villes, mais d'un ensemble de véhicules. Détaillons plus précisément l'analogie, indiquée par Parker et al. [110], entre le problème $P/s_{ij}/\sum s_{ij}$ et une formulation classique du P.T.V. :

"Un ensemble de véhicules, basés dans un dépôt central, doivent livrer plusieurs clients répartis dans différentes villes. Le problème de base consiste à définir pour chaque véhicule la route à suivre de façon à livrer tous les clients en minimisant la longueur totale des parcours. Les véhicules partent et reviennent au dépôt central."

Les différents éléments composant l'analogie entre le problème d'ordonnancement sur machines parallèles et le P.T.V. (voir Fig 2.1) sont exposés ci-après :

L'ensemble des clients : les clients sont les tâches à réaliser.

L'ensemble des véhicules : les véhicules sont les différentes machines disponibles pour réaliser les tâches. Un véhicule visitant un client est équivalent à une machine exécutant une tâche.

Les temps de voyage entre clients : ces temps représentent, dans le problème d'ordonnancement de machines parallèles, les temps nécessaire pour passer d'une tâche à une autre sur une même machine.

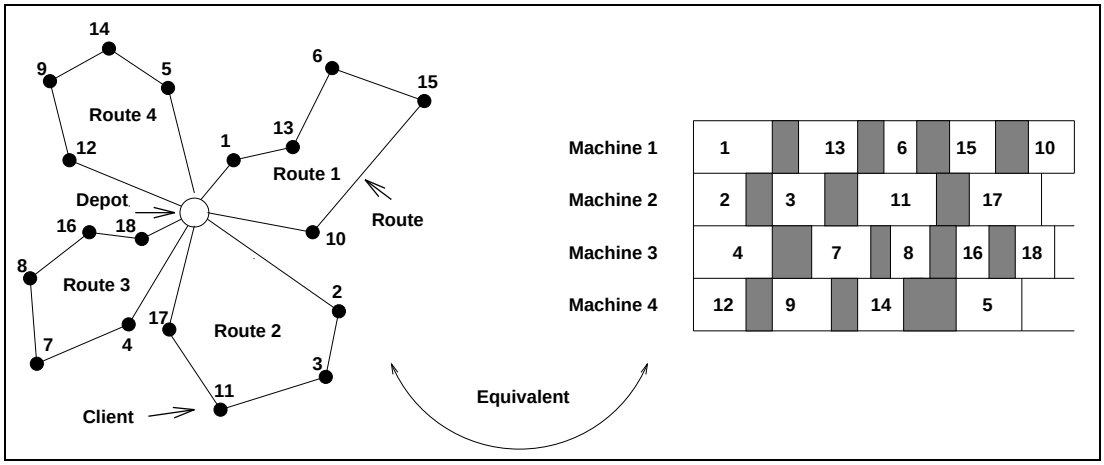


FIG. 2.1 - *Analogie avec le P.T.V.*

Le dépôt : cette ville initiale représente, pour le problème d'ordonnancement, l'état initial des machines.

Le critère : la somme de la longueur des parcours de chaque véhicule doit être minimisée et représente, dans le problème d'ordonnancement, la somme des temps de changement.

De plus, des contraintes supplémentaires peuvent être ajoutées au problème de base pour élargir le domaine de validité de cette analogie. Ainsi, pour être équivalent au problème $P/s_{ij}/C_{max}$, le P.T.V. doit en plus considérer des temps d'arrêt chez les clients (temps de ramassage), équivalents aux temps de production, et le critère devient la minimisation de la plus longue route. D'autres contraintes concernent, par exemple, des restrictions (restriction sur les capacités des véhicules, sur la longueur des routes, sur le nombre de villes visitées), des contraintes sur la visite des clients (relations de précédence entre certains clients, fenêtres de temps acceptables pour les livraisons) ou encore sur la configuration du système (les véhicules ne sont plus situés en un dépôt unique mais dans plusieurs dépôts répartis sur le territoire).

Ainsi, en se basant sur cette analogie, il semble possible d'adapter les nombreuses méthodes développées pour le P.T.V. aux problèmes d'ordonnancement sur machines parallèles. Malheureusement, la plupart de ces méthodes s'intéressent à une flotte de véhicules homogènes (ayant la même vitesse, par exemple) qui correspondent à des machines identiques. L'extension de ces méthodes au cas des machines parallèles non liées n'est souvent pas immédiate.

Nous ne présentons pas ici de revue complète sur la littérature du Problème de Tournées de Véhicules, car ce problème a été très largement étudié ces trente dernières années et de multiples algorithmes exacts ou heuristiques ont été proposés.

Aussi, nous nous limiterons à certaines méthodes qui ont déjà été adaptées à des problèmes d'ordonnancement. Le lecteur intéressé par des informations complémentaires sur les méthodes développées pour le P.T.V. pourra se reporter à certains articles ayant pour objectif de présenter un état de l'art sur les méthodes exactes (Laporte et Nobert [87]) et/ou sur les heuristiques existantes (Clark et Wright [28], Christofides [27], Bodin et Golden [16], Solomon [131], Laporte [86], Desrosiers et al. [37]). Nous citons également la thèse de Y. Rochat [124] et la bibliographie très complète de Laporte et Osman [88] qui présente 500 références sur quatre types de problèmes de tournées : le Problème du Voyageur de Commerce, le Problème de Tournées de Véhicules, le Postier Chinois et le Postier Rural.

2.3.2 Cas de processeurs identiques

Comme nous l'avons vu précédemment, le problème $P/s_{ij}/f$ (avec f pouvant correspondre à différents critères d'optimisation), peut être vu comme un problème de tournées de véhicules et est donc un problème très complexe. En fait, ce problème d'ordonnancement contient deux sous-problèmes difficiles : un problème d'affectation des tâches aux machines, et un problème d'ordonnancement proprement dit, qui consiste, une fois le problème d'affectation résolu, à déterminer les dates de lancement des opérations. Ces deux sous-problèmes ne peuvent être traités séparément, mais doivent être considérés simultanément.

Dans le cadre de l'étude des problèmes d'ordonnancement sur machines parallèles avec temps de changement plusieurs critères sont couramment étudiés : la minimisation de la somme des temps (ou des coûts) de changement, la minimisation de la durée totale de production et la minimisation du flot moyen.

2.3.2.1 $P/s_{ij}/\sum s_{ij}$ ou $P/s_{ij}/\sum Cs_{ij}$

Sumichrast et Baker [132] présentent un algorithme, visant à minimiser les coûts de changements, basé sur les solutions d'une série de sous-problèmes entiers en 0-1. Leur horizon étant divisé en jours et les demandes étant exprimées en jour-machines, ils déterminent le nombre maximal de machines pouvant rester dans le même état, tout au long de la période.

Dans son mémoire de doctorat, Aguilera [2] présente différentes méthodes de résolution pour le problème $P/s_{ij}/\sum Cs_{ij}$ issu d'une problématique industrielle de fabrication de bouteilles de verre. Ce problème, où les coûts de changement sont élevés avait également été étudié par Gonthier, lors de son travail de thèse [64] dans une optique gestion de production. Les méthodes, proposées par Aguilera sont basées pour la plupart sur des méthodes développées pour le P.T.V. et sont divisées en trois classes : méthodes exactes, méthodes de relaxation, méthodes

heuristiques.

Ces travaux ne considèrent que la minimisation des coûts de changement de production (ou des temps de changement) et ne prennent pas en compte la pénalisation sur les capacités de production induites par ces changements.

2.3.2.2 $P/s_{ij}/C_{max}$

Le problème de la minimisation de la durée totale de production sur machines parallèles identiques sans temps de préparation ($P//C_{max}$) est $\mathcal{NP}$-difficile (voir [81]). Ainsi, la plupart des méthodes considérant ce problème en y ajoutant des temps de préparation, sont des heuristiques.

Ovacik et Uzsoy [106] montrent que pour le problème $P/s_{ij}/C_{max}$, les ordonnancement sans délai ne sont pas dominants, ce qui pénalisent fortement les méthodes basées sur des algorithmes de liste qui génèrent de tels ordonnancements.

Guinet [69] s'appuie sur l'analogie avec le P.T.V. pour proposer une heuristique deux-phases inspirée de la méthode Hongroise et basée sur le réemploi des ressources.

França et al. [51], proposent une heuristique tabou décomposée en trois phases : construction d'une solution initiale, amélioration de cette solution par recherche tabou, amélioration de la séquence de la machine la plus chargée. Le voisinage est recherché à partir de procédures d'insertion particulières. Cette méthode peut permettre de résoudre presque optimalement des problèmes allant jusqu'à cinquante tâches avec néanmoins, des temps de calcul élevés.

La minimisation de la durée totale est également étudiée pour des problèmes comprenant des temps de changement majeurs et mineurs.

Ainsi, Tang [133], s'intéresse au problèmes d'ordonnancement sur machines parallèles avec deux types de changements (changements majeurs entre famille de produits et changements mineurs entre types de produits d'une même famille) qui dépendent de la tâche à venir uniquement ($P/S_j, s_j/C_{max}$). Il propose dans ce cadre deux bornes sur la valeur de la durée totale de production et une heuristique décomposant le problème en deux niveaux. Tout d'abord, au premier niveau (niveau agrégé) la part du temps passé pour chaque machine sur chaque famille est calculée en tenant compte des changements majeurs, grâce à une adaptation de l'algorithme MULTIFIT développé pour $P//C_{max}$ (voir section 2.1.1.1). Au deuxième niveau (désagrégé), une autre adaptation du MULTIFIT, permet à partir du résultat du premier niveau de calculer la taille exacte du lot de chaque type à réaliser sur les machines. Sur les exemples traités, cette méthode donne

des solutions de performance égale à $4,5\%$.

Rajgopal et Bidanda [119], étudient un problème de fabrication de pneus et se ramène à un problème particulier d'ordonnancement sur machines parallèles avec deux types de changements (majeurs et mineurs) tels que $S_{ij} = S$ pour toute famille et $s_{ij} = s$ pour tout type (problème $P/S, s/C_{max}$). Ils proposent des bornes pour le C_{max} ainsi que différentes heuristiques basées sur des réductions successives du C_{max} en économisant à chaque itération un changement majeur et un changement mineur.

2.3.2.3 $P/s_{ij}/\sum C_i$

Cheng et Chen [23] s'intéressent au problème de la minimisation du flot moyen sur deux machines parallèles identiques avec temps de changement dépendant de la séquence et un temps d'exécution égale pour toutes les tâches ($P2/s_{ij}, p_j = p/\sum C_i$) et montrent par une réduction du problème de partition que même ce problème d'ordonnancement simplifié est $\mathcal{NP}$-difficile.

2.3.2.4 $P/s_{ij}/f$

So [130] étudie le problème $P/S_{ij}, s_{ij}/f$ où les machines ont une capacité bornée, les temps de changements sont de deux types et f est la maximisation de la somme des gains apportés par l'exécution de chaque tâche. Il présente trois heuristiques : une heuristique séquentielle, une heuristique par décomposition et une heuristique gloutonne. La première heuristique, utilise la programmation dynamique pour résoudre le problème sur une machine. Puis la procédure est relancée pour les tâches restantes et la seconde machine, et continue tant que toutes les machines n'ont pas été affectées. L'heuristique par décomposition traite deux sous-problèmes. Tout d'abord une sélection de l'ensemble des tâches à affecter à l'ensemble des machines est calculée en utilisant la programmation dynamique. Puis les tâches de cet ensemble sont ordonnancées sur chaque machine prise individuellement. Cette procédure est répétée selon un schéma itératif jusqu'à ce qu'une solution réalisable soit trouvée. Enfin l'heuristique gloutonne sélectionne la tâche de gain le plus élevé, non encore affectée, et l'affecte à une machine de capacité suffisante. Il s'avère, que d'après les tests exécutés, l'heuristique gloutonne est la plus attractive de par sa rapidité d'exécution et ses bonnes performances.

2.3.3 Cas de processeurs non identiques

Dans ce paragraphe nous regroupons la littérature traitant des machines parallèles uniformes et non liées. En effet, peu de littérature porte sur ces sujets à cause de la complexité des problèmes engendrés, même si ces problèmes sont très intéressants du point de vue industriel.

2.3.3.1 *Minimisation des coûts de changement*

Aguilera et al. [1] étudient un problème industriel avec machines parallèles uniformes et minimisation des coûts de changement ($Q/s_{ij}/\sum Cs_{ij}$). Ils proposent une heuristique deux-phases qui commence par affecter des tâches aux machines puis optimise l'ordonnancement des machines.

2.3.3.2 *Minimisation du makespan (C_{max})*

Elmaghraby et Guinet [41] étudient le cas des machines parallèles non liées ($R/S_{ij}/C_{max}$). Pour cela, ils étendent la méthode du réemploi, proposée pour le cas des machines parallèles identiques, et présentent une heuristique deux-phases. La première phase, l'allocation des tâches aux machines, se base sur la résolution d'un problème de chargement de machines construit à partir du concept du réemploi. La deuxième phase détermine la séquence optimale sur chaque machine.

2.3.3.3 *Minimisation du flot moyen*

Le problème de la minimisation du flot moyen avec machines uniformes ($Q/s_{ij}/\sum C_i$) est étudié par Guinet [68] qui propose un modèle et une heuristique consistant premièrement à ré-écrire le problème d'ordonnancement en un Problème de Tournées de Véhicule où le ré-emploi est autorisé, puis à résoudre le dual de ce problème.

2.3.3.4 *Critères divers*

Lorsque l'on étudie des problèmes issus du monde industriel, il n'est pas toujours possible de se ramener aux critères classiques exposés ci-dessus. Ainsi, plusieurs personnes se sont intéressées à des critères particuliers (dérivés des critères classiques ou non), pour coller à des réalités industrielles.

Dietrich [39] considère un problème de machines parallèles non liées et recherche un ordonnancement minimisant à la fois la durée totale de production et le flot moyen ($R/s_{ij}/f$), problème rencontré lors de l'exécution de processus de tests sur les sites IBM. Elle propose une heuristique deux-phases qui commence par affecter les jobs aux machines en cherchant à équilibrer la charge de travail sur l'ensemble des machines, puis ordonne les tâches sur chaque machine de façon à réduire les temps de changement et à minimiser le flot moyen.

Assad et Ball [6] étudient un problème industriel intervenant dans la production de matelas. Ils recherchent à déterminer le plan de production journalier d'un ensemble de machines non liées servant à coudre les matelas. Leur approche est

originale puisque, contrairement aux méthodes deux phases classiques qui favorisent l'affectation des tâches aux machines puis recherchent un ordonnancement, ils privilégient le regroupement de tâches similaires (avec de faible temps de changement) puis recherchent leur affectation sur les machines.

Kedad et al. [83] proposent une heuristique deux phases pour traiter un problème d'ordonnancement rencontré dans les ateliers de conditionnement de produits alimentaires. Il consiste en la recherche d'un ordonnancement d'un ensemble de tâches, ayant des dates d'arrivées différentes, sur des machines parallèles non liées avec minimisation des temps improductifs (changement, attente). Cette heuristique se base sur un calcul de priorité de la tâche par rapport à la machine pour proposer une affectation. Puis, une deuxième phase basée sur des procédures d'améliorations locales recherche à améliorer l'ordonnancement. De plus, dans sa thèse, Kedad [82] présente également un ensemble de méthodes stochastiques pour ce problème avec contraintes temporelles sur les tâches noté $R/r_j, d_j, s_{ij}, M_j/f$ (pour la signification des différentes notations, se reporter à la section 1.2).

Chapitre 3

Etude du problème $R/s_{ij}/f$

La problématique initiale, posée par la société Pechiney, est très complexe de par le nombre d'éléments qu'elle prend en compte. Cette problématique peut être qualifiée de multiproduit, multisite et multipériode. Dans ce chapitre, nous nous consacrons à l'aspect multiproduit et proposons d'étudier un problème d'ordonnancement déterministe associé à la problématique dans laquelle les caractéristiques principales sont : un organe de production constitué de machines parallèles non liées, des temps de changements entre produits dépendants de la séquence et un objectif de minimisation globale d'un ensemble de coûts.

Nous considérons, dans ce chapitre, un sous-problème, que nous appelons Problème Statique Restreint (P.S.R.), qui consiste en la recherche d'un ordonnancement d'un ensemble de tâches sur un ensemble de machines parallèles non liées, avec temps de changement entre tâches, dépendant de l'enchaînement. L'objectif est la minimisation d'une fonction objectif f qu'il reste à déterminer. Ce problème, noté $R/s_{ij}/f$ d'après la notation introduite au chapitre 1, est assez peu étudié dans la littérature, comme nous l'avons vu précédemment (cf 2.3.3), [41], [39], [6], [83].

Ce chapitre est divisé en plusieurs parties. Tout d'abord, nous décrivons plus précisément le Problème Statique Restreint que nous avons défini, et le positionnons par rapport à la problématique générale présentée au chapitre 0. Puis nous en proposons une modélisation sous forme d'un programme linéaire permettant de définir plus précisément le problème. Cette modélisation sera également utilisée pour des résolutions exactes. Dans la partie suivante, nous exposons diverses bornes pour les problèmes $R/s_{ij}/Cmax$ et $R/s_{ij}/\sum(Coûts)$. Ensuite, nous présentons les deux approches de résolution exacte mises en œuvre et leurs limites. La section suivante pose le problème du critère à évaluer lors de la recherche de méthodes heuristiques, et propose une procédure permettant de considérer à la fois le critère temps et le critère coût. Puis nous exposons trois méthodes heuristiques développées pour traiter ce problème ainsi qu'une procédure d'amélioration. Fi-

nalement, nous expérimentons la procédure de résolution sur le problème initial.

3.1 Description du Problème Statique Restreint (P.S.R.)

La problématique initiale étant très générale, nous avons extrait un sous-problème que nous appelons P.S.R. (Problème Statique Restreint). Ce problème, dont nous supposons que toutes les données sont connues et fixées, se focalise sur l'organe de production de la problématique initiale constitué d'un ensemble de machines en parallèle. Il procède à l'affectation et l'ordonnancement d'un ensemble de tâches sur un ensemble de machines de caractéristiques différentes, en parallèle. Dans la suite de ce chapitre, nous parlerons indifféremment de lignes de production, de machines ou de processeurs. Afin de définir le P.S.R., nous avons effectué quelques modifications sur le problème initial.

3.1.1 Modification de la notion de produit

Tout d'abord, nous avons modifié la notion de produit. Dans la problématique industrielle, un client peut commander n'importe quel produit. Nous rappelons ici, qu'un produit est défini par un métal, un format et un type de col de fermeture. L'organe de production étant réparti sur plusieurs sites, le choix de l'affectation de la production d'un produit sur une machine doit se faire en tenant compte des problèmes liés à la production, mais également de la distribution, puisque ces deux aspects sont liés en terme de coûts. En effet, du choix de l'affectation de la fabrication d'un produit sur une machine va non seulement dépendre les temps et coûts liés à cette fabrication, mais également les coûts liés à la distribution du produit depuis sa machine de fabrication (et donc depuis l'usine de fabrication) jusqu'au client l'ayant commandé. Pour contourner cette difficulté, nous choisissons de caractériser un produit non seulement par ses caractéristiques techniques, mais également par le client qui le commande. Ainsi, la notion de destination est intégrée dans la notion de produit et les coûts de distribution d'un produit, défini de cette façon, ne dépendent que du produit et de son lieu de fabrication. Ces coûts peuvent alors être agrégés avec les coûts de production qui dépendent également du produit et du lieu de fabrication. Nous définissons alors Cp_{im} le coût de production-distribution associé à la fabrication du produit i sur le processeur m tel que : $Cp_{im} = Cp'_{im} + Ct_{im}$ où Cp'_{im} est le coût de production simple du produit i sur le processeur m et Ct_{im} est le coût de distribution associé au produit i fabriqué sur le processeur m. Ainsi, lorsque dans ce chapitre, nous parlerons de coûts de production d'un produit sur une ligne de production, il faudra sous-entendre coûts de production et de distribution relatifs à la production de ce produit sur cette ligne.

3.1.2 Détermination de l'ensemble des tâches à effectuer

Ensuite, nous définissons l'ensemble des tâches à exécuter en fonction des demandes des clients. Les pénalités en temps et en coûts induites par les changements de production étant importantes, nous avons décidé de ne pas autoriser la préemption pendant l'ordonnancement. Ainsi, le calcul de la taille des lots à réaliser se fait en amont, et n'est pas considéré ici. Lorsqu'une commande correspond à un très grand volume de produits, cette commande est divisée en plusieurs petites commandes de l'ordre de la taille des autres commandes, ce qui permet éventuellement, de répartir la production de cette grande commande sur plusieurs machines. Une tâche correspond alors à la production d'une quantité donnée d'un certain produit.

3.1.3 Étude du cas monopériode

Finalement, nous regardons dans cette partie de l'étude, le cas statique d'une seule période. Ainsi, pour la période considérée, l'ensemble des tâches à réaliser, la capacité des machines, représentant le temps disponible à la production pendant la période, et les autres paramètres sont connus et fixés. De plus, dans le cadre de l'étude d'une seule période, les aspects de stockage ne sont pas considérés.

3.1.4 Description des composantes du problème

De manière formelle, les composantes du problème que nous étudions dans ce chapitre peuvent être décrites de la façon suivante :

M	: Nombre de processeurs.
N	: Nombre de tâches (nous supposons $N > M$).
T	: Longueur de la période considérée.
$J = \{1, ..., N\}$	: Ensemble des tâches à réaliser.
$P = \{1, ..., M\}$	: Ensemble des processeurs disponibles.
Tp_{im}, Cp_{im}	: Temps et coût de production de la tâche i sur le processeur m.
Tc_{ijm}, Cc_{ijm}	: Temps et coût de changement lors du passage de la tâche i à la tâche j sur le processeur m.

3.2 Modélisation

Nous proposons dans cette partie une modélisation du problème défini dans la section précédente, sous forme d'un programme linéaire en nombres entiers avec variables bivalentes. Nous noterons ce problème $R/s_{ij}, Cmax < T/\sum(Coûts)$ pour indiquer que nous cherchons à respecter la période T et à minimiser l'ensemble des coûts. Pour cela, nous utilisons les indices suivants :

$i = 1..N$: indice relatif aux tâches.
$j = 1..N$: indice relatif aux tâches.
$m = 1..M$: indice relatif aux processeurs.
$l = 1..L(m)$: indices relatif aux positions sur le processeur m.

avec $L(m)$ le nombre maximal de tâches que le processeur m peut exécuter et L_m l'énumération de toutes les positions valides sur le processeur m ($L_m = \{1..L(m)\}$), vérifiant les inégalités (3.1) et (3.2). L'inéquation (3.1) indique qu'au moins une des machines doit accepter plus de N/M tâches, tandis que l'inéquation (3.2) s'assure que la somme des positions valides sur l'ensemble des processeurs est supérieure ou égale au nombre de tâches à exécuter.

$$N/M \leq \max_m L(m) \leq N \tag{3.1}$$

$$N \leq \sum_m L(m) \leq N \times M \tag{3.2}$$

Nous définissons les variables bivalentes suivantes :

$$y_{ilm} = \begin{cases} 1 & \text{Si la tâche } i \text{ est exécutée sur } m \text{ en } l^e \text{ position.} \\ 0 & \text{Sinon.} \end{cases}$$

Ces variables y_{ilm} permettent à elles seules de décrire un ordonnancement réalisable, mais elles ne permettent pas de comptabiliser linéairement les temps de changement nécessaires pour le passage d'une tâche à une autre sur le même processeur. Ainsi, d'autres variables bivalentes x_{ijm}, représentant la succession entre les tâches, doivent être introduites afin d'obtenir un programme linéaire.

$$x_{ijm} = \begin{cases} 1 & \text{Si la tâche } j \text{ suit immédiatement la tâche } i \text{ sur le processeur } m. \\ 0 & \text{Sinon.} \end{cases}$$

Nous pouvons remarquer que les variables x_{ijm} se déduisent des variables y_{ilm} de la façon suivante : $x_{ijm} = 1$ si et seulement si il existe l tel que y_{ilm} et $y_{j(l+1)m}$ soient égaux à 1.

$$x_{ijm} = \sum_{l=1}^{l=L(m)-1} (y_{ilm} \times y_{j(l+1)m}) \qquad \forall (i,j) \in J \times J, \forall m \in P$$

Cette équation n'est pas linéaire. Elle sera ré-écrite sous la forme suivante, afin d'être utilisée dans le programme linéaire.

$$y_{ilm} + y_{j(l+1)m} - x_{ijm} \leq 1 \qquad \forall (i,j) \in J \times J, \forall m \in P, \forall l \in L_m$$

Nous cherchons à affecter l'ensemble des tâches aux processeurs de façon à minimiser l'ensemble des coûts (production-distribution, changement de production)

et respecter la période T. Ce problème se traduit par le programme linéaire PL1 présenté ci-dessous. PL1 est caractérisé par : l'ensemble des tâches à réaliser J, l'ensemble des processeurs disponibles P, les temps de production Tp_{im}, les coûts de production Cp_{im}, les temps de changement Tc_{ijm}, les coûts de changement Cc_{ijm} et la période à respecter T.

(PL1 $(J, P, Tp_{im}, Cp_{im}, Tc_{ijm}, Cc_{ijm}, T)$)

$$\min \sum_{m\in P} \sum_{i\in J} \Big(\sum_{l\in L_m} Cp_{im} \times y_{ilm} + \sum_{j\in J} Cc_{ijm} \times x_{ijm} \Big) \tag{3.3}$$

$$\sum_{i\in J} \Big(\sum_{l\in L_m} Tp_{im} \times y_{ilm} + \sum_{j\in J} Tc_{ijm} \times x_{ijm} \Big) \leq T \qquad \forall m \in P \tag{3.4}$$

$$\sum_{m\in P} \sum_{l\in L_m} y_{ilm} = 1 \qquad \forall i \in J \tag{3.5}$$

$$\sum_{i\in J} y_{ilm} \leq 1 \qquad \forall m \in P, \forall l \in L_m \tag{3.6}$$

$$\sum_{i\in J} y_{i(l+1)m} - \sum_{i\in J} y_{ilm} \leq 0 \qquad \forall m \in P, \forall l \in L_m \backslash L(m) \tag{3.7}$$

$$y_{ilm} + y_{j(l+1)m} - x_{ijm} \leq 1 \qquad \forall (i,j) \in J \times J, \forall m \in P, \forall l \in L_m \backslash L(m) \tag{3.8}$$

$$y_{ilm} = 0 \text{ ou } 1 \qquad \forall i \in J, \forall l \in L_m, \forall m \in P \tag{3.9}$$

$$x_{ijm} = 0 \text{ ou } 1 \qquad \forall (i,j) \in J \times J, \forall m \in P \tag{3.10}$$

La fonction objectif 3.3 vise la minimisation de l'ensemble des coûts de production (incluant les coûts de distribution) et de changement.

L'équation (3.4) assure que l'ensemble des temps associés aux productions des tâches et aux changements entre les tâches, affectées à chaque processeur, ne dépassent pas la période. Les équations (3.5), (3.6) et (3.7) procèdent à l'affectation des tâches. L'équation (3.5) s'assure que chaque tâche est affectée, tandis que l'inéquation (3.6) empêche plusieurs tâches d'occuper une même position et l'inéquation (3.7) interdit les trous (c'est à dire les positions vacantes au milieu d'un ordonnancement) qui rendraient impossible la prise en compte des temps et coûts de changement. Finalement, la dernière inéquation (3.8) permet de donner à x_{ijm} la valeur 1 lorsqu'il existe l tel que $y_{ilm} = y_{j(l+1)m} = 1$ afin de pouvoir comptabiliser les temps et coûts de changement de production dans l'équation (3.4) et dans la fonction objectif (3.3).

Pour comptabiliser le nombre de variables et de contraintes de cette modélisation, posons L la somme du nombre de positions valides autorisées sur chacun des processeurs ($L = \sum_m L(m)$), c'est-à-dire, le nombre de combinaisons représentées par $l \times m$. Nous avons alors $L \times N$ variables y_{ilm} et $M \times N^2$ variables x_{ijm} pour

M contraintes (3.4), N contraintes (3.5), L contraintes (3.6), L contraintes (3.7) et $L \times N^2$ contraintes (3.8).

Il n'est pas possible de déterminer L avant la résolution du programme linéaire, ou tout au moins, avant la connaissance de l'instance à résoudre, car cela reviendrait à fixer un nombre maximal de positions valides autorisées par machine, ce qui peut écarter la solution optimale. Nous ne pouvons alors qu'estimer le pire et le meilleur des cas.

Dans le pire des cas, on considère qu'un processeur peut réaliser toutes les tâches. Dans ce cas, $L(m)$ devient égal à N (quelque soit m) et L égal à $N \times M$. Le nombre de variables binaires est égal à $2 \times N^2 \times M$ et le nombre de contraintes de l'ordre de $M \times N^3$.

Dans le meilleur des cas, on parvient à déterminer optimalement, avant la résolution du problème, le nombre de tâches réalisées sur chaque machine. Comme toutes les tâches doivent être exécutées, nous avons alors $L = N$. Nous obtenons un nombre de variables binaires de l'ordre de $M \times N^2$ et de l'ordre de N^3 contraintes.

Le modèle PL1 nous a permis d'exposer clairement chacune des contraintes que doit respecter l'ordonnacement. Nous utiliserons ce modèle, au cours de ce chapitre pour la résolution exacte de petits problèmes (problèmes ne comportant que quelques processeurs et quelques tâches) et dans le chapitre 4 lors de la décomposition du problème global multisite.

3.3 Calcul de bornes

L'objet de cette partie est de calculer un ensemble de bornes inférieures pour le problème d'ordonnancement d'un ensemble de processeurs parallèles non liés avec temps de changement, entre tâches, dépendant de la séquence. Des bornes sont recherchées pour deux critères correspondant à deux aspects importants de notre problème : la minimisation de la durée totale d'éxecution (makespan) et la minimisation des coûts.

3.3.1 Bornes inférieures pour le problème $R/s_{ij}/Cmax$

Nous recherchons tout d'abord des bornes inférieures pour la minimisation de la durée totale d'exécution de l'ordonnancement d'un ensemble de tâches sur processeurs parallèles non liés. Nous remarquons que ces bornes sont également valides pour des problèmes sans temps de changement entre tâches, puisqu'il suffit alors de remplacer les temps de changement par zéro. Trois bornes basées sur des

observations sur les ordonnancements sont proposées.

3.3.1.1 Chaque tâche est allouée à un processeur

Toute tâche doit être exécutée. Au mieux, chaque tâche est affectée sur le processeur capable de l'exécuter en un temps minimum. La durée totale de l'ordonnancement est donc supérieure (ou égale) au plus grand des plus petits temps de production. D'où une première borne inférieure $C1$, définie par :

$$C1 = \max_i(\min_m Tp_{im}) \tag{3.11}$$

3.3.1.2 Toutes les tâches sont allouées à un processeur

Il existe donc au moins un processeur avec $\lfloor \frac{N-1}{M} \rfloor + 1$ tâches. La durée totale de l'ordonnancement est donc supérieure (ou égale) au meilleur temps mis par un processeur pour exécuter $\lfloor \frac{N-1}{M} \rfloor + 1$ tâches. Ce temps est supérieur (ou égal) aux $\lfloor \frac{N-1}{M} \rfloor + 1$ plus petites sommes (temps de production + temps de changement) moins le plus grand temps de changement considéré.

$$C2 = \min_m(\sum_{i \in I_m} (Tp_{im} + \min_j Tc_{ijm}) - \max_{i \in I_m}(\min_j Tc_{ijm})) \tag{3.12}$$

avec I_m représentant l'ensemble des $\lfloor \frac{N-1}{M} \rfloor + 1$ tâches sélectionnées, c'est-à-dire, les $\lfloor \frac{N-1}{M} \rfloor + 1$ tâches qui ont la plus petite somme (temps de production + temps de changement). I_m vérifie, pour tout processeur m :

$$Card(I_m) = \lfloor \frac{N-1}{M} \rfloor + 1$$

et $\forall i \in I_m$, $\forall k \notin I_m$

$$Tp_{km} + \min_j Tc_{kjm} \geq Tp_{im} + \min_j Tc_{ijm}$$

3.3.1.3 Toutes les tâches sont exécutées

Au mieux, chaque tâche est affectée à son "meilleur processeur" (en terme de temps). La longueur de l'ordonnancement est donc supérieure (ou égale) à la moyenne des meilleurs temps. Cette moyenne est constituée de N temps de production et de $N-M$ temps de changement de production. Il faut donc déterminer quels sont les $N-M$ couples (temps de production + temps de changement) à considérer, et les M temps de production restants.
En fait, un temps de changement, suivant une tâche, doit être effectué sur le même processeur que la tâche elle même. Nous recherchons donc pour chaque tâche i quelles sont les deux machines qui permettent d'une part d'exécuter la tâche, d'autre part de l'exécuter et d'effectuer un changement, le plus rapidemment

possible. L'écart temporel e_i représente la différence entre ces deux temps, c'est-à-dire la perte de temps introduite par la prise en compte du temps de changement suivant la production de la tâche i.

$$e_i = \min_m(Tp_{im} + \min_j Tc_{ijm}) - \min_m Tp_{im}$$

La moyenne se calcule alors à partir des plus petits temps de production pour chaque tâche auxquels nous ajoutons les $N - M$ plus petits écarts (composant l'ensemble I).
D'où

$$C3 = \frac{1}{M}[\sum_{i \in J} \min_m Tp_{im} + \sum_{i \in I} e_i] \tag{3.13}$$

avec I représentant le $N - M$ tâches dont l'écart e_i est plus petit. I vérifie :

$$Card(I) = N - M$$

et $\forall i \in I, \forall k \notin I$

$$e_k \geq e_i$$

Ainsi, la borne inférieure pour le problème $R/s_{ij}/Cmax$ est la plus grande des trois bornes proposées ci-dessus :

$$Cinf = max(C1, C2, C3)$$

3.3.2 Borne inférieure pour le problème $R/s_{ij}/\sum(Coûts)$

La borne inférieure proposée pour la minimisation des coûts se rapproche de la borne $C3$ définie pour la minimisation de la durée totale de production. En effet, la transposition des bornes $C1$ ou $C2$ pour la minimisation des coûts donnerait, dans tous les cas, une moins bonne borne que la transposition de $C3$. Toutes les tâches doivent être exécutées. Au mieux, toutes les tâches sont affectées à leur meilleur processeur (en terme de coûts). Le coût total de l'ordonnancement est donc supérieur (ou égal) à la somme des meilleurs coûts. Cette somme est au moins composée de N coûts de production et $N - M$ coûts de changement de production (cas où les M machines sont utilisées).
De même que pour $C3$, il faut déterminer quels sont les $N - M$ couples (coût de production + coût de changement) à considérer et les M coûts de production restants. Nous définissons alors la notion d'écart financier e'_i représentant la différence entre le meilleur coût de production de i et sa meilleure somme (coût de production + coût de changement). Ainsi, e'_i représente le surcoût engendré par la prise en compte d'un changement après l'exécution de la tâche.

$$e'_i = \min_m(Cp_{im} + \min_j Cc_{ijm}) - \min_m Cp_{im}$$

D'où

$$Binf = \sum_{i \in J} \min_{m} Cp_{im} + \sum_{i \in I'} e'_i \tag{3.14}$$

avec I' représentant les $N - M$ tâches dont le surcout e'_i est le plus petit. I' vérifie :

$$Card(I') = N - M$$

et $\forall i \in I', \forall l \notin I'$

$$e'_l \geq e'_i$$

3.3.3 Validité des bornes

Différentes expérimentations ont été menées pour vérifier la validité des bornes. Pour cela, nous avons comparé, pour chacun des deux critères temps et coûts, la valeur des bornes et la valeur de la solution optimale, pour plusieurs types de problèmes. Ici la valeur la solution optimale est recherchée (en utilisant CPLEX 5.0) pour l'un des deux critères, sans tenir compte du deuxième.

Le tableau 3.1 reporte les résultats. La première colonne indique le type de problème : nombre de processeurs, nombre de tâches et entre parenthèses, le nombre d'instances traitées. Les trois colonnes suivantes évaluent la valeur de la borne pour la durée totale de production en donnant la valeur optimale (moyenne de toutes les instances), la valeur de la borne (moyenne de toutes les instances) et l'écart en pourcentage (en moyenne). Les trois dernières colonnes reportent le même résultat pour le coût.

Problème	Cmax			Coût		
mach-tâches	Optimum	Borne	Ecart (%)	Optimum	Borne	Ecart (%)
4-8 (10)	122	95	22	1707	1697	5,8
3-6 (10)	125	98	21	1536	1529	4,7
3-10 (10)	204	184	9,8	2318	2302	6,7
2-10 (10)	330	297	9,9	3550	3535	4,1

TAB. 3.1 - *Efficacité des bornes*

Le premier résultat indiqué sur le tableau concerne la validité de la borne sur le Cmax. Il s'avère que pour des problèmes avec peu de tâches par machine, la borne se trouve à un peu plus de 20% de la solution optimale. Lorsque le nombre de tâches par machine augmente, cette borne descend en dessous de 10%. Les instances étudiées dans cette expérimentation sont caractérisées de la façon suivante : temps opératoires variant aléatoirement dans l'intervalle [25, 100] et temps de changement variant dans l'intervalle [13, 50].

Les résultats concernant la borne sur le coût semble être très bons. En fait, nous nous sommes aperçus, au cours de l'expérimentation, que les résultats variaient beaucoup en fonction de la prépondérance ou non des coûts de distribution. En effet, lorsque ceux ci sont très supérieurs aux coûts de production et de changement, la borne se rapproche à moins de 1% de l'optimum. Au contraire, lorsque les coûts de distribution sont du même ordre que les coûts de production (ce sont les résultats reportés dans le tableau), la borne a tendance à s'éloigner un peu de l'optimum mais donne tout de même un très bonne valeur. Ainsi, la validité de cette borne est à considérer en fonction du type de problème à résoudre.

3.4 Méthodes exactes

Dans cette section, nous exposons deux méthodes exactes utilisées pour résoudre le problème $R/s_{ij}, T/\sum(coûts)$, où T est la longueur de la période. La première méthode consiste en la résolution du programme linéaire exposé section 3.2 en utilisant le solveur CPLEX 5.0. La deuxième est une procédure de séparation et évaluation spécifique à ce problème utilisant les bornes décrites dans la section précédente.
L'expérimentation de ces méthodes porte sur une comparaison de leurs temps d'exécution et leurs limites d'utilisation en terme de taille des problèmes qu'elles peuvent résoudre en un temps raisonnable.

3.4.1 Résolution du programme linéaire avec CPLEX 5.0

La première méthode consiste à résoudre le programme linéaire en nombres entiers (PL1) composé des équations 3.3 à 3.10. Comme nous l'avons signalé lors de la présentation de ce programme linéaire, la détermination du nombre de positions valides par machine est un point crucial, car de là dépend le nombre de variables. Il est intéressant de limiter au maximum le nombre de variables du programme linéaire, de façon à espérer avoir une résolution plus rapide, sans écarter la solution optimale en éliminant trop de possibilités.

Nous proposons d'utiliser, pour limiter le nombre de variables, une borne supérieure du nombre de positions valides par machine qui représente le nombre maximal de tâches qu'une machine peut exécuter sans dépasser la longueur de la période. Cette borne est calculée pour chaque machine. Pour cela, nous classons les tâches dans l'ordre croissant de leur somme (durée d'exécution + plus petit temps de changement associé), sur la machine considérée, et comptabilisons le nombre de tâches, sélectionnées dans cet ordre, suffisant pour dépasser la capacité de la machine. Ce nombre représente donc le nombre maximal de tâches que peut exécuter la machine pendant la période.

Formellement, cette borne se calcule de la façon suivante: pour chaque machine m, soit k l'indice relatif aux tâches classées dans l'ordre croissant des $(Tp_{im} + \min_j Tc_{ijm})$. Soit $L(m)$ la borne supérieure du nombre de positions valides sur la machine m.

$L(m) = min(K)$ tel que:

$$\sum_{k=1}^{k=K} (Tp_{km} + \min_j Tc_{kjm}) - max(\min_{j,k \leq K} Tc_{kjm}) \geq T \qquad (3.15)$$

En utilisant cette borne, nous avons écrit un programme C++ qui, à partir d'un ensemble de données (nombre de machines, nombre de tâches, temps et coûts associés à la production, aux changements et à la distribution) générées aléatoirement, mais en tenant compte des ordres de grandeurs de la problématique industrielle, écrit le programme linéaire en nombres entiers associé. Ensuite, nous utilisons CPLEX 5.0, sur un Pentium 133MHz pour résoudre ce programme linéaire. Lorsqu'il n'y a pas de solution réalisable, c'est-à-dire de solution respectant la contrainte de la période, CPLEX nous le signale en indiquant qu'il n'existe pas de solution entière au problème.

Pour la résolution de ce type de problème avec CPLEX, nous avons mené une expérimentation pour connaître quels seraient les meilleurs paramètres à utiliser. En effet, la documentation de CPLEX 5.0 indique qu'"à l'inverse des programmes purement linéraires, les paramètres par défaut utilisés par CPLEX ne donnent pas, la plupart du temps, les meilleures performances pour résoudre les programmes linéaires mixtes" [79]. Une stratégie pour améliorer les performances est proposée. Nous l'avons suivie et avonc donc, pour chacun des paramètres recherché le meilleur choix.
Pour ce qui est des choix des directions de branchement et des variables sur lesquelles se font les branchements, les paramètres par défaut utilisés par CPLEX semblent être les plus efficaces.
Par défaut, la stratégie de recherche adoptée par CPLEX est la *"best bound search"*, c'est-à-dire que l'on favorise le noeud ayant la meilleure borne. Il s'avère qu'en appliquant la stratégie *"best estimate search"*, CPLEX met, en moyenne, un peu moins de temps à résoudre les problèmes (jusqu'à 25% plus vite pour certaines instances).
La méthode la plus efficace pour la résolution des sous-problèmes apparaît être la résolution du dual *"dual simplex"* qui correspond à la méthode par défaut choisie par CPLEX.
La recherche automatique de SOS (*"Special Ordered Set"*), permettant normalement de procéder à des stratégies de branchement adaptées, augmente considérablement le temps d'exécution.
Choisir la résolution du *"dual simplex"* à la place de *"primal simplex"* pour la résolution de la relaxation associée à la racine de l'arbre de recherche, permet

parfois, mais pas systématiquement, de diminuer le temps nécessaire à la résolution.
Nous avons également testé plusieurs ordres dans l'écriture des contraintes. En effet, cet élément a de l'importance, puisque lorsqu'aucune priorité n'a été définie, CPLEX considère les variables dans l'ordre où elles sont rencontrées.

En résumé, les différentes expérimentations montrent que les paramètres ont de l'influence sur le temps de résolution des problèmes, en fonction du type de problème. Pourtant, dans notre cas, cette influence n'intervient pas toujours dans le même sens suivant les instances et semble limitée (réduction de 25 % du temps seulement). Aussi, nous avons choisi, de façon à être le plus neutre possible, et parce que cela ne semble par aberrant, de garder pour notre expérimentation les paramètres par défaut utilisés par CPLEX.

3.4.2 Procédure de séparation et évaluation

La deuxième méthode est une procédure de séparation et évaluation que nous avons développée pour ce problème particulier. Comme nous l'avons vu lors de la présentation de la méthode dans la section 1.4.1.2, les procédures de séparation et évaluation sont basées sur une énumération intelligente de l'ensemble des solutions. Une bonne procédure de séparation et évaluation est une procédure qui utilise de bonnes bornes (bornes supérieures et inférieures) lui permettant de réduire au maximum la taille de l'arbre de recherche.

D'une part, dans le cas d'une minimisation, l'utilisation d'une borne supérieure, quand elle existe, nous permet d'éviter de poursuivre la construction d'une solution si la solution non terminée est déjà de coût plus élevé que la borne. Nous choisissons d'initialiser cette borne par référence à une solution construite à l'aide d'une des heuristiques détaillées section 3.6 : si la solution proposée par l'heuristique respecte la période, la borne supérieure est initialisée au coût de cette solution, si au contraire, la solution ne respecte pas la période, la borne supérieure est initialisée à l'infini, car il n'est pas possible de savoir à l'avance si une solution réalisable existe. Puis, au cours de l'exécution, dès qu'une meilleure solution est trouvée, cette borne est réactualisée.

D'autre part, l'utilisation d'une borne inférieure permet de stopper la construction d'une solution quand les éléments restants à considérer vont, à coup sûr, engendrer une solution de coût trop élevé. Ainsi, dans notre procédure, chaque fois qu'une tâche entre dans l'ordonnancement en cours de construction, nous réactualisons la borne inférieure qui calcule le coût minimal relatif à l'insertion des tâches restantes, de façon à ne pas explorer des branches de l'arbre menant à des solutions trop coûteuses. De plus, lorsque l'on décide de changer le processeur en cours d'affectation, nous calculons une borne inférieure sur le temps nécessaire

à l'ordonnancement des tâches restantes, sur les processeurs encore disponibles, de manière à vérifier que la solution en cours de construction peut mener à une solution réalisable.

Ainsi, la méthode de séparation et évaluation est composée d'une procédure récursive (Ajout($mach$, pos)) qui affecte ou non, en fonction du coût de la solution et des bornes utilisées, une tâche sur le processeur (la machine) $mach$ à la position pos. Le schéma général de la procédure est donné ci-dessous :

Procédure principale

// Initialisation de la borne supérieure
Si l'heuristique trouve une solution réalisable (respectant la période)
alors $Bsup =$ *résultat de l'heuristique*
sinon $Bsup = \infty$
FinSi
// Appel de la procédure récursive
Ajout(1, 1)
// Résultat
Si $Bsup = \infty$
alors Il n'existe pas de solution réalisable
sinon Affichage de la solution optimale
FinSi

Ajout(mach, pos) *// Ajout d'une tâche sur mach en position pos*
// Ajout en fin du processeur en cours de chargement
Pour toutes les tâches i
Si i non déjà prise
alors Affectation de la tâche i sur $mach$ en position pos
Si $nr > 0$ et $coutsol + Binf < Bsup$ et $Cmax <$ *période*
alors Ajout($mach$, $pos + 1$)
sinon,
Si $nr = 0$ et $coutsol < Bsup$
alors Nouvelle solution optimale
$Bsup = coutsol$
FinSi
FinSi
Désaffectation de la tâche i du processeur $mach$
FinSi
FinPour
// Changement du processeur à charger
Si $mach < nbmachines$ et $coutsol + Binf < Bsup$
et $Cmax <$ *période* et $Cinf <$ *période*

alors Ajout($mach + 1$, 1)
FinSi

Schéma général de la procédure de séparation et évaluation.

$mach$	:	Numéro du processeur.
pos	:	Numéro de la position.
nr	:	Nombre de tâches non affectées.
$Bsup$	:	Borne supérieure (puis coût de la solution optimale).
$Binf$	:	Borne inférieure sur les coûts des tâches non selectionnées.
$Cmax$	:	Durée totale d'exécution de la solution courante.
$coutsol$	:	Coût de la solution courante.
$Cinf$	:	Borne inférieure de la durée d'exécution des tâches non sélectionnées sur l'ensemble des processeurs disponibles.

Lorsque l'on développe une procédure de séparation et évaluation, il faut toujours étudier le compromis entre la réduction du temps de traitement d'un problème en limitant le parcours de l'arbre par l'introduction de bornes supplémentaires et l'augmentation du temps de traitement induit par les calculs associés à ces bornes.

Problème mach-prod	Sans utiliser de bornes		En utilisant Binf		Gain en %		En utilisant Binf et Cinf		Gain en %	
	Temps	Noeuds	Temps	Noeuds	Temps	Noeuds	Temps	Noeuds	Temps	Noeuds
2-3 (100)	0	36	0	26	-	28%	0	18	-	50%
3-5 (100)	0,08	1136	0,06	536	25%	52,81%	0	285	-	75%
2-8 (40)	31,6	198043	5,35	35299	83%	82,17%	3,85	22699	87,8%	88,53%
3-8 (60)	57,69	265234	11,77	53083	79,6%	80%	4,71	20498	91,8%	92,3%

TAB. 3.2 - *Comparaisons des temps d'exécution en secondes*

Ainsi, nous avons comparé l'exécution de la procédure de séparation et évaluation avec et sans l'utilisation de la borne inférieure $Binf$ (bornes sur les coûts), en terme de durée d'exécution et de nombre de noeuds visités. Puis, nous avons ajouté à cette première borne, la borne $Cinf$ (sur la durée totale d'exécution des tâches restantes) lors du passage à une nouvelle machine. Les résultats sont reportés dans le tableau 3.2. La première colonne du tableau indique le type de problème étudié et le nombre d'instances. Les deux colonnes suivantes reportent les résultats pour une exécution sans utilisation de bornes. Puis, les quatre colonnes suivantes indiquent les résultats lors de l'utilisation de la borne $Binf$ et donnent le gain apporté. Enfin, les quatre dernières colonnes indiquent les mêmes

résultats lorsque les deux bornes $Binf$ et $Cinf$ sont utilisées.

Il apparaît clairement dans ce tableau, que l'utilisation des bornes proposées $Binf$ (sur les coûts) et $Cinf$ (sur les temps) est bénéfique pour la recherche d'une solution et que ceci est d'autant plus vrai que le problème est grand (réduction de plus de 90% de l'arbre de recherche et du temps d'exécution).

3.4.3 Expérimentation

Nous cherchons ici à évaluer l'efficacité des deux méthodes exactes présentées ci-dessus. Pour cela, nous comparons tout d'abord les méthodes en terme de durée d'exécution, puis exposons leurs limites. L'expérimentation reporte ici, seulement les résultats concernant les instances pour lesquelles des solutions, respectant la contrainte de longueur de la période, existent. En effet, il se peut, que la recherche de solution n'aboutisse pas lorsque la période est trop petite par rapport à l'ensemble des tâches à réaliser.

3.4.3.1 Comparaison des temps d'exécution

Cette comparaison est faite de deux façons : nous relevons pour chacune des deux méthodes le temps nécessaire à l'obtention de la solution optimale, mais également, le temps nécessaire à l'obtention d'une première solution réalisable ainsi que son coût.

Problème mach-prod (# instances)	Séparation et évaluation			Programmation linéaire			Coût PSE Temps
	Temps Opt	Temps 1^{ere} Sol	Coût 1^{ere} Sol	Temps Opt	Temps 1^{ere} Sol	Coût 1^{ere} Sol	1^{ere} Sol CPLEX
3-8 (10)	1,2	0,1	233	83,9	13	226	202
3-10 (10)	16	3,4	271	1218	29,4	248	234
4-10 (10)	22,5	7,4	289	1760	51	282	266
6-10 (10)	49,3	0,1	480	336,4	29	346	291

TAB. 3.3 - *Comparaisons des temps d'exécution en secondes et des coûts*

Le tableau 3.3 reporte les résultats pour différents types de problèmes. La première colonne indique le problème traité, les trois colonnes suivantes indiquent pour la procédure de séparation et évaluation, le temps nécessaire à l'obtention de la solution optimale, le temps nécessaire à l'obtention d'une première solution réalisable et le coût de cette première solution. Ensuite, trois autres colonnes reportent les mêmes résultats pour la résolution du programme linéaire PL1 en utilisant CPLEX 5.0. Enfin, la dernière colonne, donne la valeur du coût de la

solution trouvée par la procédure de séparation et évaluation au bout du temps nécessaire à CPLEX pour trouver la première solution.

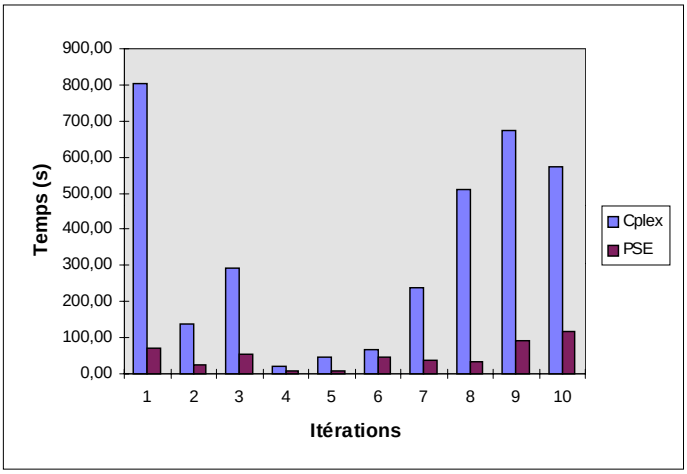


FIG. 3.1 - *Comparaison des temps d'exécution*

La première constatation inspirée par la lecture du tableau 3.3 est que la procédure de séparation et évaluation proposée est beaucoup plus rapide que la résolution du programme linéaire PL1, en utilisant CPLEX 5.0. La figure 3.1 illustre ce fait, en représentant pour dix itérations de problèmes à six machines et dix tâches le temps mis, pour la résolution à l'optimum, par la procédure de séparation et évaluation et par la résolution du programme linéaire avec CPLEX 5.0.

La deuxième constatation est que la première solution réalisable donnée par CPLEX est en moyenne meilleure que la première solution donnée par la procédure de séparation et évaluation.

Pourtant, en regardant plus précisément, nous pouvons voir que CPLEX met beaucoup plus de temps à proposer une première solution que la procédure de séparation et évaluation et surtout que dans le temps nécessaire à CPLEX pour obtenir cette première solution, la P.S.E trouve en moyenne une solution de coût inférieur. Ce résultat est détaillé par la figure 3.2 qui représente pour chacune des dix itérations des problèmes à six machines et dix tâches les valeurs des coûts des solutions trouvées par la P.S.E. et par CPLEX, au bout du temps nécessaire à CPLEX pour obtenir sa première solution. Sur cette figure, nous observons, comme dans le tableau 3.3, qu'en moyenne le coût de la solution obtenue par la P.S.E. est meilleur (seule pour l'itération 6, cela n'est pas vérifié), mais nous pouvons également voir que pour quatre itérations (itérations 1, 2, 5 et 10), la procédure de séparation et évaluation est déjà à l'optimum lorsque CPLEX trouve une première solution réalisable.

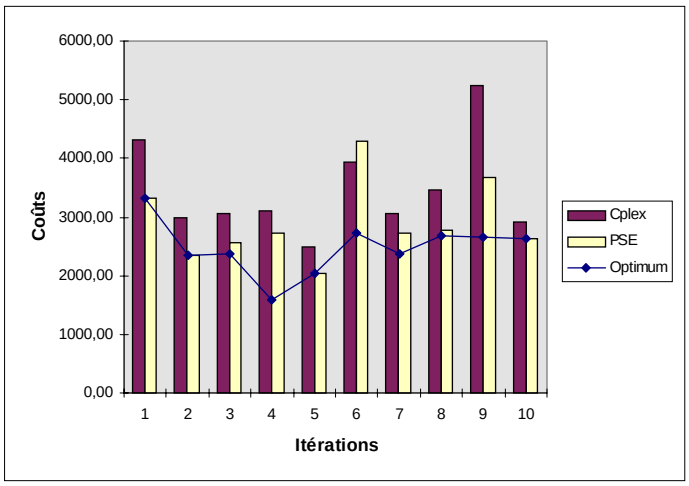


FIG. 3.2 - *Comparaison des coûts des solutions trouvées par la P.S.E. et CPLEX dans un même laps de temps*

3.4.3.2 Limites d'utilisation

Le temps nécessaire à l'obtention des solutions optimales variant beaucoup en fonction des instances traitées, il est intéressant de voir le pourcentage de problèmes de chaque taille pouvant être résolus à l'optimum dans un temps imparti. Ainsi, pour différentes tailles de problèmes et différents temps, nous indiquons dans le tableau 3.4, le pourcentage de problèmes résolus à l'optimum pendant ce temps.

Tps	30 s		60 s		90 s		180 s		300 s		600 s	
Pb	PSE	PL	PSE	PL	PSE	PL	PSE	PL	PSE	PL	PSE	PL
3-8	100%	18%	100%	36%	100%	63%	100%	91%	100%	100%	100%	100%
3-10	100%	0%	100%	0%	100%	0%	100%	0%	100%	8%	100%	46%
4-10	90%	0%	100%	0%	100%	0%	100%	0%	100%	10%	100%	40%
6-10	30%	10%	70%	20%	80%	30%	100%	40%	100%	60%	100%	80%
4-12	0%	-	0%	-	0%	-	0%	-	10%	-	30%	-

TAB. 3.4 - *Limites d'utilisation des méthodes exactes : Pourcentages de problèmes résolus dans un temps imparti*

Nous pouvons voir, dans ce tableau que même pour des problèmes très petits (par exemple, trois processeurs et dix tâches), la résolution du programme linéaire en utilisant CPLEX, demande plus de 10 minutes. Par contre, pour les quatre premiers types de problèmes étudiés (3-8, 3-10, 4-10, 6-10), la procédure

de séparation et évaluation permet toujours de trouver une solution en dix minutes. La dernière ligne, reportant le temps nécessaire à la procédure de séparation et évaluation pour résoudre des problèmes à quatre processeurs et douze tâches, met en évidence les limites d'utilisation d'une telle méthode pour des problèmes de taille moyenne, puisque seuls 30% des problèmes ont été résolus en moins de 10 minutes (le temps moyen de résolution est de 1401 secondes, soit près de 25 minutes).

3.4.3.3 Comportement de la procédure de séparation et évaluation

De façon à étudier plus précisément le comportement de la procédure de séparation et évaluation, nous reportons dans la figure 3.3 l'évolution de la valeur de la fonction objectif de la solution en fonction du temps, au cours de la résolution d'un problème à 5 machines et 12 tâches.

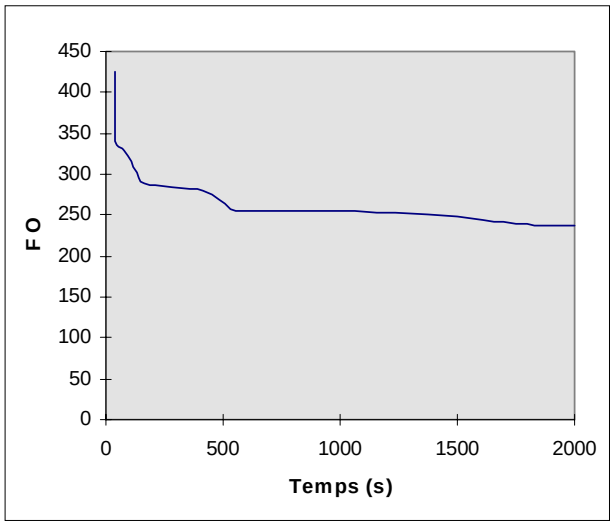


FIG. 3.3 - *Evolution de la fonction objectif au cours du temps*

La figure nous montre que le coût de la solution décroît très rapidemment au début de la recherche, puis l'amélioration au cours du temps est beaucoup plus lente. Il semble donc raisonnable d'utiliser, comme méthode heuristique pour des problèmes plus gros, la procédure de séparation et évaluation pendant un temps limité. La section suivante étudie cette possibilité.

3.4.3.4 Utilisation de la PSE comme heuristique

La procédure de séparation et évaluation utilisée comme une heuristique est lancée, puis stoppée au bout d'un temps donné. Dans ce cas, la solution optimale n'est pas forcément atteinte. Cependant, comme le montre la figure 3.3, il est possible, qu'en coupant la procédure au bout d'un temps donné, la solution obtenue,

à ce moment là, soit déjà de bonne qualité.

Pour savoir quels sont les problèmes pour lesquels il est possible d'appliquer une telle méthode, nous avons relevé, pour différentes tailles de problème, le temps nécessaire à l'obtention de la première solution. Nous avons effectué des moyennes sur 40 problèmes, car les temps varient fortement en fonction des instances (problèmes plus ou moins faciles à résoudre). Le tableau 3.5 reporte les temps moyens nécessaires à l'obtention de la première solution ainsi que les temps minimaux, maximaux et les écarts moyens.

Problème $M-N$	Temps moyen	Temps minimal	Temps maximal	Écart moyen
5-15 (40)	2	0	25	2,8
3-12 (40)	12	0	146	14,7
4-12 (40)	3	0	43	4,2
6-18 (40)	179	0	4001	303
4-16 (40)	360	0	2800	425

TAB. 3.5 - *Temps nécessaires (s) à l'obtention d'une première solution*

Nous pouvons voir que la difficulté de résolution, pour un nombre de tâches ou un nombre de processeurs donné, augmente avec le nombre moyen de tâches par processeur. Ainsi, les problèmes 3-12 sont plus longs à résoudre que les problèmes 4-12. En ce qui concerne les problèmes 4-16, le temps moyen de résolution est de 360 secondes, mais peut monter jusqu'à 2800 secondes pour des instances très difficiles. Les temps minimal égal à zéro indiquent les cas où l'heuristique, utilisée comme première borne supérieure, trouve une solution réalisable.

Les tailles de problèmes résolubles par l'heuristique basée sur une énumération tronquée sont malgré tout encore limitée, et il nous faut donc développer des méthodes heuristiques pour pouvoir traiter des problèmes de taille raisonnable.

3.5 Evaluation bi-critère

Dans cette partie, nous posons le problème du ou des critère(s) à optimiser lors de notre recherche heuristique de solutions répondant au problème $R/s_{ij}, T/\sum coûts$. En effet, le critère de ce problème semble bien défini (somme des coûts), pourtant, cette minimisation des coûts n'est pas l'objectif premier à considérer dans ce problème où nous recherchons une solution respectant une certaine période.

Dans cette partie, nous commençons par déterminer les motivations d'une étude bicritère, puis nous exposons quelques éléments essentiels de la théorie de la décision multicritère. Ensuite, nous décrivons quelques applications de cette théorie à des problèmes d'ordonnancement et terminons en présentant notre approche.

3.5.1 Motivations d'une telle étude

Le problème, tel qu'il est posé dans sa version initiale, consiste à trouver un ordonnancement d'un ensemble de tâches sur un ensemble de processeurs en parallèle, tel que cet ordonnancement respecte la longueur d'une période imposée tout en minimisant la somme des coûts (production, changement).

Lorsque l'on recherche une solution optimale exacte, nous avons vu que ce problème était noté $R/s_{ij}, Cmax < T/\sum(coûts), T$ étant la longueur de la période à respecter. Ce problème a été l'objet d'études dans la section précédente où nous avons vu que la recherche de la solution optimale n'aboutissait pas toujours. En effet, il est possible que le nombre de tâches à ordonnancer soit trop important et qu'aucun ordonnancement ne puisse respecter la période fixée. Nous définissons alors, dans notre contexte, la notion d'ordonnancement réalisable :

Définition 1 *Un ordonnancement est dit réalisable si sa longueur totale d'exécution est inférieure à la longueur de la période donnée.*

Cependant, dans le cadre d'applications industrielles, la personne chargée de la mise en place du plan de production recherche, dans un outil d'aide à la décision, des propositions d'ordonnancement. Elle préférera alors une méthode qui lui fait toujours des propositions, même si ces propositions ne respectent pas toujours la période, à une méthode qui lui signale, de temps en temps, qu'il n'existe pas de solution réalisable et ne lui fait aucune proposition.

De plus, nous avons vu que les méthodes exactes ne permettaient pas de traiter, dans notre contexte, des problèmes de tailles réelles (cinq machines, quinze tâches nécessitent, pour certaines instances, plus de douze heures de calcul). Aussi, il est nécessaire de rechercher des méthodes heuristiques permettant de résoudre au mieux les problèmes de grandes tailles. Or l'inconvénient dans l'utilisation d'heuritiques, est la dégradation de la qualité de la solution par rapport aux méthodes exactes. Ainsi, il est possible qu'en utilisant une heuristique, qui considère le respect de la période en tant que contrainte, nous ne trouvions pas de solution satisfaisant cette contrainte, même si une telle solution existe. La stratégie que nous avons alors adoptée vise le respect de la période, tout en ne l'imposant pas comme contrainte. Ce qui permet, éventuellement de proposer des solutions ne respectant pas cette période, mais s'en rapprochant. Pour cela, nous relaxons la contrainte de capacités (inéquation 3.4 de PL1) et intégrons les aspect temporels et de coûts dans une même fonction objectif qui devient alors fonction de deux

critères, de natures différentes, le temps et le coût.

Pour traiter ce problème bi-critère, il est nécessaire de rappeler quelques définitions basiques issues de la théorie de la décision multicritère et d'étudier ce qui a déjà été fait dans le domaine de l'ordonnancement en environnement multicritère. C'est ce que nous proposons de faire dans les deux parties qui suivent.

3.5.2 Eléments de la théorie de la décision multicritère

Dès 1973, Panwalkar et al. [107], ont fait le constat que les décideurs, en ordonnancement, considèrent plusieurs critères avant de formuler leurs choix ou décisions. De plus, l'application de méthodes monocritères pour la recherche de solutions en environnement multicritère, peut être très mauvaise puisque la solution optimale pour un critère peut s'avérer être très mauvaise pour un autre critère, et une solution présentant un bon compromis entre les critères aurait de bonnes chances de satisfaire le décideur. Ainsi, il est souvent nécessaire de considérer l'ensemble des critères qui interviennent dans le problème. Nous définissons la notion de critère en conflit.

Définition 2 *Deux critères sont en conflit, si l'optimisation de l'un des critères peut entraîner une dégradation de l'autre critère.*

Lorsque les critères ne sont pas en conflit, c'est-à-dire, lorsque l'optimisation d'un des critères optimise les autres, le problème peut être traité comme un problème monocritère. Par contre, dès que les critères sont en conflit, il est nécessaire de connaître l'importance donnée par le décideur à chacun des critères pour les coupler de façon intelligente. Malheureusement, les critères intervenant dans les décisions sont très souvent non commensurables et sont donc non traduisibles en un coût et difficilement comparables entre eux. Ainsi, l'évaluation de l'importance des critères est très délicate d'autant plus que le processus de décision du décideur est la plupart du temps non modélisable. Comme le fait remarquer A. Schärlig, l'importance des critères est souvent exprimée par un nombre que l'on appelle poids. Il ajoute qu'en réalité, les coefficients d'importance que sont les poids n'ont pas la même signification selon qu'ils interviennent dans une méthode ou dans une autre [129].

Ainsi, dans un environnement multicritère, une des difficultés concerne la définition de l'optimalité. Idéalement, la solution optimale serait celle qui minimise tous les critères en même temps. Or une telle solution ne peut exister lorsque les critères sont en conflit. Il est alors nécessaire de définir la notion de dominance d'une solution [52], [127].

Définition 3 *Une solution σ^* est non dominée dans un environnement de minimisation multicritère composé de n critères (C_1, ...C_n), s'il n'existe pas de*

solution σ telle que :

$$C_1(\sigma) \leq C_1(\sigma^*)$$

$$\ldots$$

$$C_n(\sigma) \leq C_n(\sigma^*)$$

où au moins une de ces inégalités est stricte.

Ainsi, une solution est non dominée par une autre, si elle est meilleure pour au moins l'un des critères. Il est alors possible de définir l'optimum au sens de Pareto, telle qu'il a été défini par Pareto, lui-même [109].

Définition 4 *Une solution est optimale au sens de Pareto (Pareto optimale, ou Pareto efficace) si elle n'est pas dominée par rapport à l'ensemble des critères.*

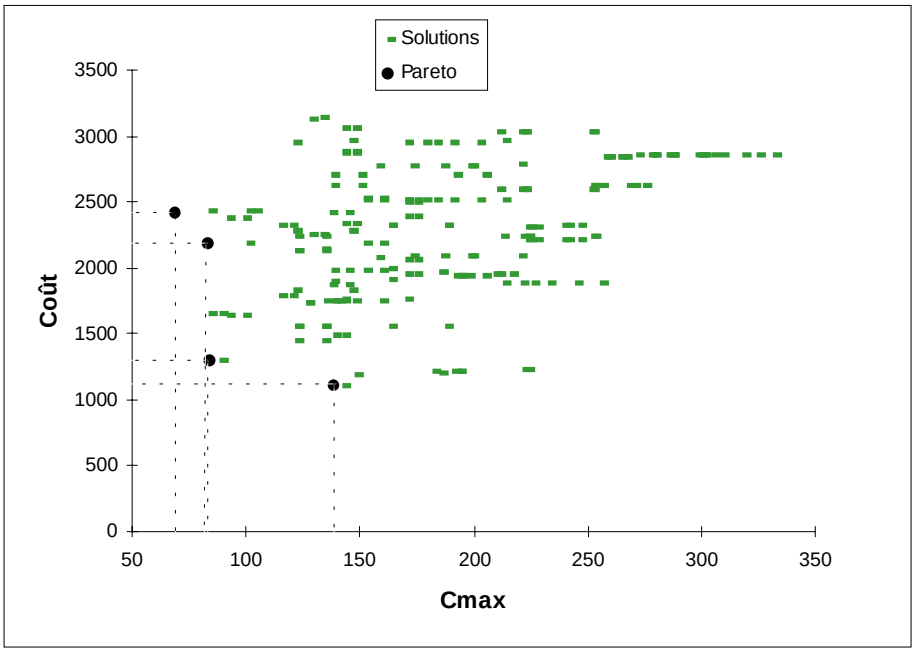


FIG. 3.4 - *Optima de Pareto*

Dans notre problème, l'environnement est bicritère. L'ensemble des solutions peut alors se représenter sur un plan en fonction de leurs valeurs pour chacun des critères. Pour cela, nous associons à chaque solution σ le point (temps(σ), coût(σ)) dans $\mathcal{R}^2$. La figure 3.4 montre, pour une instance de notre problème composée de trois machines et quatre tâches, l'ensemble des solutions existantes. Dans un objectif de minimisation des deux critères, temps et coût, les optima de Pareto sont les solutions non dominées, c'est-à-dire, celles dont le cadran inférieur gauche ne contient aucune autre solution. Elles sont représentées sur la figure par une

croix.

Cette figure montre également que la minimisation de l'un des critères ne minimise pas forcément l'autre. D'où l'intérêt de relever les optima de Pareto pour laisser au décideur le choix entre plusieurs solutions de bon compromis.

La décision multicritère est une théorie très complète que nous n'allons pas exposer ici (voir Roy et Vincke [127], ou Roy et Bouyssou [126] pour une présentation générale). Nous nous sommes restreints, dans cette partie, aux définitions essentielles à notre approche. Dans la section suivante, nous exposons les problèmes d'ordonnancement multicritères traités dans la littérature.

3.5.3 Multicritère et ordonnancement

Nous avons vu dans la partie précédente l'importance, en environnement multicritère, de considérer l'ensemble des critères. Pourtant, la plupart des recherches en ordonnancement portent sur des objectifs monocritères. Quelques travaux ont néanmoins été menés et Nagar et al. [104] ou plus récemment, T'Kindt et Billaut [136], proposent un état de l'art sur les problèmes d'ordonnancement multicritères et bicritères, sur machine simple ou plusieurs machines. Beaucoup de problèmes multicritères, en ordonnancement, concernent des problèmes bicritères sur machine unique.

Ainsi, Van Wassenhove et Gelders [141] ont étudié le cas d'une machine unique avec minimisation conjointe du flot et du retard maximal. Une solution optimale peut être obtenue pour chacun des deux critères pris séparément en un temps polynomial (règle SPT : Shortest Processing Time, pour le flot et règle EDD : Earliest Due Date pour le retard maximal). Ils montrent que la règle SPT donne également une solution Pareto optimale lorsque les deux critères sont considérés, ce qui n'est pas le cas pour la règle EDD. Ils proposent un algorithme pseudo-polynomial permettant de générer tous les points Pareto efficaces. Plus tard, Van Wassenhove et Baker [140] s'intéressent à un problème d'ordonnancement sur machine simple où temps et coût sont en conflit et proposent une approche bicritère. Leur algorithme résoud optimalement l'un des critères (règle EDD, par exemple) et modifient la séquence trouvée de façon à diminuer la valeur de l'autre critère. Ils génèrent ainsi, l'ensemble des points efficaces (non dominés).

De même, Nelson et al. [105] étudient l'ordonnancement d'une machine en considérant les critères de minimisation du flot, du nombre de tâches en retard et du retard maximal deux à deux, puis les trois critères ensemble. Là encore, chaque problème pris séparément est solvable polynomialement. Ils proposent des méthodes de séparation et évaluation, basées sur les méthodes associées à chaque critère, pour générer l'ensemble des points Pareto efficaces. Pour la combinaison

des trois critères, ils basent leur procédure de séparation et évaluation sur les résultats des problèmes bicritères.

Hoogeveen et van de Velde ont étudié plusieurs problèmes d'ordonnancement multicritères sur machine unique. Ainsi, ils proposent dans [72] des bornes inférieures pour le problème $1/nmit/\sum Ci + Lmax + Emax$, où $nmit$ indique que les temps morts sur la machine ne sont pas admis, $\sum Ci$ représente le flot total, $Lmax$ le retard maximal et $Emax$ l'avance maximale. Pour cela ils comparent deux approches : l'approche par amélioration potentielle maximale ("maximal potential improvement"), qui est une approche de type hiérarchique considérant un critère après l'autre, et l'approche de séparation des objectifs ("objective splitting") qui considère les trois critères simultanément. Ils constatent que la deuxième approche est plus rapide et de meilleure performance que la première, et proposent de l'améliorer de façon à obtenir, moyennant un peu de calculs supplémentaires, une meilleure borne. Également, Hoogeveen et van de Velde [73] s'intéressent à l'ordonnancement d'un ensemble de tâches sur une machine unique avec minimisation du flot et d'un coût maximal, défini par une fonction régulière de la date d'achèvement. Emmons [42] propose de traiter le problème de façon hiérarchique en cherchant à minimiser le flot pour un coût minimal déterminé par la règle de Lawler [90]. Hoogeveen et van de Velde recherchent polynomialement, pour une valeur du coût ne dépassant pas une certaine borne, l'ensemble de points Pareto optimaux, dont le nombre est borné par $n(n-1)/2+1$, si n est le nombre de tâches.

La particularité des travaux de Mittenthal et al. [102] est l'étude de fonctions non régulières des dates de fins d'exécution des tâches. Dans ces fonctions, l'un des critères (CT) est la mesure de la tendance centrale (fonction régulière, par exemple le flot moyen ou la date de fin de la tâche du milieu), tandis que l'autre critère (DSP) est la mesure de la dispersion des dates de fin (fonction non régulière, par exemple variance, déviation par rapport à la moyenne,...). Dans ce contexte, De et al [36] étudient le problème bicritère du flot moyen et de la variance et proposent un algorithme pseudo-polynomial de résolution. Dans [102], Mittenthal et al. présentent pour l'ensemble des problèmes CT-DSP, les résultats connus ainsi que leurs propres résultats sur les combinaisons connexes des deux critères.

Plus proche de notre problématique, Aguilera [2] et Bourgade et al. [18] considèrent un problème d'ordonnancement bicritère sur machine unique avec minimisation du retard maximal et des coûts de changement. Ils proposent d'adapter, pour la recherche de solutions efficaces, la procédure de séparation et évaluation de Little et al. [97], développée pour le problème du voyageur de commerce. Pour l'évaluation des solutions, deux fonctions objectif, agrégeant les deux critères, sont retenues et comparées. Il est à noter que l'une de ces fonctions objectif peut générer des solutions dominées au sens strict.

Les différents problèmes reportés ci-dessus ont deux points communs qui diffèrent avec notre problème. Tout d'abord, tous ces articles traitent de problèmes sur machine unique, ce qui élimine le problème d'affectation sous-jacent à l'utilisation de machines en parallèle. De plus, dans chacun des problèmes étudiés, l'un au moins des deux critères, est solvable polynomialement lorsqu'il est pris individuellement. Ainsi, la solution optimale obtenue pour ce critère est une bonne référence et peut servir de borne pour les méthodes multicritères. Dans notre cas, chacun des critères (makespan et coûts) conduit à un problème $\mathcal{NP}$-difficile, et il en est de même pour le problème bicritère. Il n'est donc pas possible d'utiliser de bonnes méthodes permettant de rechercher l'ensemble des solutions efficaces. Nous noterons tout de même, dans un contexte similaire au notre, l'étude récente de T'Kindt et al [137] concernant l'ordonnancement de machines parallèles non liées avec prise en compte de deux critères : la maximisation de la marge et la minimisation de $Imax$ représentant la différence entre le $Cmax$ et le $Cmin$ (plus petite date de fin d'une machine).

D'autres travaux, moins nombreux, ont été menés pour les problèmes de flowshop ou jobshop multicritères. Ainsi, Rajendran [118] étudie un problème de flowshop à deux étages avec minimisation conjointe de deux critères : le makespan (date maximale de fin d'exécution) et le flot moyen. Le premier critère est solvable polynomialement par l'algorithme de Johnson [80], tandis que le deuxième est $\mathcal{NP}$-difficile. Il propose un algorithme de séparation et évaluation s'appuyant sur la solution optimale obtenue pour le makespan, ainsi que deux heuristiques privilégiant chacune, l'un ou l'autre des critères. De plus, T'Kindt et al. [138], [14] ont proposé des méthodes exactes ainsi que des heuristiques pour un problème de flowshop sur deux machines bicritère, noté $F2//Lex(Cmax, \bar{C})$.

Belton et Elder [13], se sont interessés à un problème de jobshop où la performance des solutions est mesurée selon dix critères, dont la plupart, pris individuellement, rendent le problème $\mathcal{NP}$-difficile. Ils adoptent une approche heuristique de résolution, utilisant une somme pondérée des sept paramètres d'entrée, pour fournir une solution initiale au décideur. Ils insistent sur l'intérêt d'avoir des méthodes interactives permettant au décideur d'agir sur la solution proposée, mais aimeraient proposer un méchanisme de contrôle systématique permettant de réagir automatiquement, en fonction de la solution trouvée, sur les paramètres d'entrée afin de réexécuter l'heuristique et obtenir une meilleure solution. La réalisabilité d'un tel méchanisme de contrôle impose de trouver des corrélations entre les paramètres d'entrée et les performances mesurées, ce qui permettrait à une heuristique de contrôler les performances des solutions. Ils étudient alors les relations existantes entre les poids donnés aux paramètres d'entrée et les performances des solutions trouvées, et concluent que ces relations ne sont pas directes.

3.5.4 Notre approche

Maintenant que nous avons exposé les rudiments de l'optimisation bicritère et donné quelques exemples d'application, nous exposons ci-après l'approche que nous proposons pour traiter notre problème d'ordonnancement.

Nous supposons, dans cette partie, que nous avons à disposition un Oracle permettant d'ordonnancer, selon un certain critère de recherche, un ensemble de tâches sur un ensemble de processeurs non liés avec temps de changement entre les tâches dépendant de la séquence. Les méthodes proposées pour traiter ce problème sont détaillées dans les parties 3.6 et 3.7, mais il est nécessaire avant de les exposer, de déterminer les besoins d'optimisation.

Ainsi, la première question à se poser lors de l'optimisation d'un problème est celle du critère à optimiser ou, plus généralement, celle de la caractérisation de la solution optimale. Dans notre problème d'ordonnancement, la définition de la meilleure solution va dépendre en quelque sorte du résultat. En effet, si la méthode utilisée pour la résolution indique que le problème est réalisable, c'est-à-dire qu'il existe un ensemble d'ordonnancements respectant la période (cet ensemble pouvant être réduit à un seul élément), la solution optimale sera celle de moindre coût, parmi les éléments de cet ensemble. Par contre, si la méthode ne met pas en évidence de solution réalisable, il n'est pas possible de définir quelle sera la meilleure solution, et nous préférons, dans ce cas, donner un ensemble de solutions non dominées par rapport aux deux critères temps et coût. Si deux solutions sont égales sur les deux critères et non dominées par aucune autre solution, nous en choisissons arbitrairement une pour faire partie de l'ensemble des solutions optimales. Nous introduisons alors la notion de solution optimale pour notre problème.

Définition 5 *La ou les solutions optimales σ^* à notre problème vérifient l'une des conditions suivantes :*

1. $Cmax(\sigma^*) \leq T$ *et* $\forall \sigma \quad tq \quad Cmax(\sigma) \leq T, \quad Coût(\sigma) > Coût(\sigma^*)$
2. $\forall \sigma, \quad Cmax(\sigma) > T$ *et* $(Coût(\sigma) \geq Coût(\sigma^*)$ *ou* $Cmax(\sigma) \geq Cmax(\sigma^*))$

Ainsi, l'optimalité des solutions est clairement définie, et la comparaison entre deux solutions σ_1 et σ_2 sera faite selon le schéma suivant qui nous permet de déterminer, parmi un ensemble de solutions candidates, la ou les solutions optimales telles qu'elles ont été définies dans la définition 5.

Si $Cmax(\sigma_1) \leq T$
  alors Si $Cmax(\sigma_2) \leq T$
    alors Si $Coût(\sigma_1) \leq Coût(\sigma_2)$
      alors garder σ_1, rejeter σ_2
      sinon garder σ_2, rejeter σ_1
    sinon garder σ_1, rejeter σ_2
  sinon Si $Cmax(\sigma_2) \leq T$
    alors garder σ_2, rejeter σ_1
    sinon Si $Cmax(\sigma_1) \leq Cmax(\sigma_2)$
      alors si $Coût(\sigma_1) \leq Coût(\sigma_2)$
        alors garder σ_1, rejeter σ_2
        sinon garder σ_1 et σ_2
      sinon si $Coût(\sigma_1) \leq Coût(\sigma_2)$
        alors garder σ_1 et σ_2
        sinon garder σ_2, rejeter σ_1

Schéma de comparaison entre deux solutions.

Maintenant que nous avons défini ce qu'étaient les solutions optimales de notre problème, il reste à savoir comment obtenir un ensemble de solutions candidates. Nous avons vu précédemment (paragraphe 3.4.3) qu'il n'était pas possible de procéder à une énumération complète de l'ensemble des solutions et que seules des méthodes heuristiques sont applicables lorsque la taille du problème augmente. Les méthodes, que nous proposons plus loin, se basent sur un critère, à déterminer, pour procéder à l'ordonnancement. Il semble important que ce critère, qui va déterminer une direction de recherche et dont va dépendre les choix dans la construction de la solution, tienne compte des deux aspects temps et coût. Ainsi, nous avons choisi de linéariser les deux aspects dans une même fonction objectif [48], construite de la façon suivante :

$$f = \alpha \times Cmax + (1 - \alpha) \times A \tag{3.16}$$

avec $\alpha \geq 0$ et A représentant la somme des coûts.

$$A = \sum_{i \in J} Cp_{im}^{m(i)} + \sum_{j \in J} Cc_{i(j),j}^{m(i)} \tag{3.17}$$

où $m(i)$ est le processeur qui exécute i.
où $i(j)$ est la tâche précédant j.

Remarque 1 *Cette fonction objectif, telle qu'elle est définie ci-dessus détermine pour les méthodes de construction de l'ordonnancement, une direction de*

recherche. Ce n'est en aucun cas la fonction d'évaluation des solutions. Cette distinction, entre le critère utilisé pour la recherche de solution et la performance des solutions se retrouve dans [13], où une somme pondérée de sept paramètres est utilisée par l'heuristique, alors que la performance des solutions est mesurée selon dix critères.

Pour la mise en œuvre d'une telle fonction objectif, deux problèmes se posent : la normalisation des valeurs du temps et du coût et la détermination du poids α.

En effet, pour pouvoir comparer deux composantes d'une même fonction objectif, il est important que ces composantes varient selon une même échelle. Pour cela, Belton et Elder [13] proposent d'utiliser dans leur fonction objectif linéarisant plusieurs critères, des valeurs des données remise à l'échelle dans un même intervalle. Pour chacune des types de données, ils translatent les intervalles $[min, max]$ en l'intervalle $[0, 100]$. Ainsi, chacun des critères à un même niveau de priorité. Cette méthode est facilement applicable lorsque la fonction objectif est composée d'une combinaison de critères simples. Or, dans notre problème, ce n'est pas le cas, puisque le critère A est composé d'une somme de deux données : les coûts de production et les coûts de changement (voir l'équation 3.17). Or il existe un ratio entre ces deux coûts, qu'il est indispensable de conserver et qui serait perdu si l'on translatait chacun des deux coûts dans un même intervalle.

Ainsi, nous avons choisi une autre approche pour normaliser le temps et le coût. Nous utilisons les bornes inférieures calculées selon 3.3.1 et 3.3.2 et divisons les temps par la borne inférieure du makespan et les coûts par la borne inférieure calculée sur les coûts. Ainsi, interviennent dans la fonction objectif, non pas les valeurs réelles du temps et du coût (qui peuvent être d'ordre très différent) mais leur importance relative en fonction des bornes, ce qui donne du sens à la comparaison.

Le second problème à traiter consiste à déterminer les valeurs du coefficient α, c'est-à-dire de déterminer l'importance à donner à chacun des critères. A priori, il n'est pas possible de savoir s'il faut privilégier l'aspect temporel ou l'aspect financier pour obtenir les solutions que nous avons définies comme optimales pour notre problème (cf définition 5). Ainsi, nous proposons une procédure itérative qui va permettre, en fonction du résultat trouvé à l'itération précédente, de réajuster les poids, de façon à se diriger au mieux vers les solutions optimales. Cette procédure fonctionne sur le principe suivant : si la période semble facile à respecter, c'est-à-dire, si à l'itération précédente, une solution réalisable a été trouvée, il est possible de diminuer l'influence du temps dans la recherche de solutions et d'augmenter le poids associé aux coûts (en diminuant α). Au contraire, si lors de l'itération précédente, aucune solution réalisable n'est trouvée, il est nécessaire d'augmenter α de façon à privilégier les aspects temporels dans la recherche de

solution et espérer de trouver une solution réalisable. En fait, cette procédure itérative simule le comportement que pourrait avoir un décideur face aux solutions données par les différentes itérations. En fonction de la solution trouvée, le décideur choisira d'augmenter (ou de diminuer) l'importance donnée au temps par rapport aux coûts de façon à se diriger vers ce qu'il pense être la meilleure solution. C'est ce que fait cette procédure, en modifiant systématiquement le poids en fonction des solutions proposées.
Le schéma de la procédure est détaillé ci-dessous.

Initialisations
  $Nbiter = 0$.
  Exécuter l'Oracle avec $\alpha = 0.5$.
Affectations
  Tantque $Nbiter < N_{max}$
    Si $Cmax \leq T$
      alors $\alpha = \alpha/2$.
      sinon,
        Si $Cmax > T$
          alors $\alpha = (1 + \alpha)/2$.
        FinSi
    FinSi
    Exécuter l'Oracle, sauver la ou les meilleures solutions
    $Nbiter = Nbiter + 1$
  FinTantque

Schéma itératif modifiant les valeurs de α en fonction des ordonnancements obtenus.

où :
- $Cmax$: Durée totale de l'ordonnancement.
- T : Longueur de la période.
- Nbiter : Nombre d'itérations.
- N_{max} : Nombre maximal d'itérations.

Dans cette procédure, nous considérons qu'il existe un Oracle permettant d'affecter un ensemble de tâches sur des processeurs parallèles, moyennant un critère de recherche multicritère de la forme $\alpha Temps + (1 - \alpha)Coûts$.

Nous proposons de construire cet Oracle, à partir de deux composantes : une heuristique de construction, permettant de créer une première solution, et une procédure d'amélioration, permettant, à partir d'échanges locaux, d'améliorer la qualité de cette première solution.

Ces deux composantes sont décrites dans les deux sections suivantes. Pour l'heuristique de construction, nous proposons trois méthodes différentes. Pour la procédure d'amélioration, nous proposons une procédure basée sur des échanges locaux et étudions si l'application de cette procédure aux différentes solutions des heuristiques de construction a un comportement différent. Ainsi, nous recherchons le meilleur couple, heuristique de construction et procédure d'amélioration, à utiliser pour constituer l'Oracle d'affectation.

3.6 Heuristiques de construction

Nous présentons dans cette partie trois heuristiques ayant comme particularité commune de considérer à la fois le temps et le coût. Ces heuristiques se basent sur des indicateurs différents pour affecter les productions aux machines mais toutes utilisent deux paramètres qui sont le temps d'affectation d'une tâche i sur un processeur m (TA_{im}), et son coût associé (CA_{im}). Ces paramètres sont défnis de la façon suivante :

$$TA_{im} = \begin{cases} Tp_{im} & \text{Si la tâche } i \text{ est exécutée sur } m \text{ en première position.} \\ Tp_{im} + Tc_{j_m im} & \text{Si la tâche } i \text{ est exécutée sur } m \text{ après } j_m \end{cases} \tag{3.18}$$

$$CA_{im} = \begin{cases} Cp_{im} & \text{Si la tâche } i \text{ est exécutée sur } m \text{ en première position.} \\ Cp_{im} + Cc_{j_m im} & \text{Si la tâche } i \text{ est exécutée sur } m \text{ après } j_m \end{cases} \tag{3.19}$$

3.6.1 Heuristique *"Priorités"*

La première heuristique proposée pour la construction de l'ordonnancement, vise à minimiser le C_{max} de l'ensemble des machines tout en limitant les coûts associés [46]. En cherchant à minimiser le C_{max}, nous équilibrons la charge de travail (en temps) sur chaque machine et espérons ainsi respecter la période. Nous présentons ici l'indicateur utilisé puis donnons les grandes étapes de l'heuristique.

3.6.1.1 L'indicateur

L'indicateur TC (Temps-Coût) utilisé prend en compte à la fois les aspects temporels et les aspects financiers du problème. Il est construit à partir des temps d'affectation (TA_{im}) et des coûts d'affectation associés (CA_{im}) de la façon suivante, pour chaque produit i sur chaque processeur (ou machine) m :

$$TC_{im} = \alpha \times TA_{im} + (1 - \alpha) \times CA_{im}$$

Cet indicateur varie donc au cours du temps. Dès qu'un produit est affecté sur une machine, tous les indicateurs concernant cette machine sont à recalculer.

3.6.1.2 Principe

Cette heuristique gloutonne construit l'ordonnancement de chaque machine en parallèle. Chaque fois qu'une machine se libère, une tâche (production à réaliser) est choisie et affectée à la machine libre. Le choix de la tâche se fait en fonction de listes de priorité sur les tâches et sur les machines. L'heuristique est divisée en cinq étapes.

Etape 1 Créer la liste de priorité sur les tâches.

Etape 2 Choisir et affecter une première production sur chaque machine.

Etape 3 Pour toutes les productions non affectées, initialiser ou mettre à jour la liste de priorité sur les machines.

Etape 4 Pour la prochaine machine libre, choisir et affecter une tâche.
Si moins de $n-(m-1)$ productions ont été affectées, retour à l'étape 1.

Etape 5 Affecter les $m-1$ dernières tâches.

Nous explicitons ici chaque étape.

Étape 1 : Liste de priorité sur les tâches

Cette liste détermine l'ordre de considération des tâches. Au sommet de la liste doit donc se trouver la tâche qui augmenterait le plus la longueur de l'ordonnancement et les coûts associés si elle n'était pas affectée à son meilleur processeur. Nous définissons pour cela le poids de production Wp_{im} d'une tâche i sur une machine m comme étant la somme pondérée du temps et du coût de production.

$$Wp_{im} = \alpha \times Tp_{im} + (1-\alpha) \times Cp_{im}$$

Trois critères (C1, C2 et C3) de construction de la liste ont été étudiés [44].

- *C1:* Ce critère se base sur une somme simple des temps et coûts de production ($\sum_m Wp_{im}$). C1 favorise donc "la tâche la plus lourde".

- *C2:* Ce critère se base sur la perte moyenne encourue lorsque l'on n'affecte pas une tâche à sa meilleure machine ($(\sum_m (Wp_{im} - \min_m Wp_{im})/(M-1))$. C2 privilégie la tâche de perte moyenne plus élevée.

- *C3 :* Ce critère calcule deux regrets R1 et R2. R1 est la différence entre les deux plus petits poids d'une tâche tandis que R2 est la différence entre le plus petit et le plus gros. R3 peut être calculé comme étant la somme de R1 et R2. Ainsi une tâche avec un R3 important est, soit une tâche avec une très bonne machine, soit une tâche avec une très mauvaise machine. Il faut donc la privilégier.

Étape 2 : Affectation des premières tâches

L'objectif est de déterminer pour chaque machine la première production à exécuter. Dans l'analogie avec le PTV, ces premières affectations sont appelées "seed customers". Tout l'ordonnancement est basé sur le choix de ces affectations.
L'idée générale est d'affecter sur des machines différentes des tâches ayant un grand changement de production entre elles. Définissons Wc_{ij} le poids du changement lors du passage de i à j (remarquons que Wc n'est pas symétrique).

$$Wc_{ij} = \alpha \times T_{ij} + (1 - \alpha) \times Cc_{ij}$$

La règle de sélection est la suivante : tout d'abord les deux tâches avec le plus grand Wc_{ij} sont sélectionnées. Puis on sélectionne les tâches les plus éloignées des tâches déjà sélectionnées, c'est à dire celles ayant la plus grande somme $\sum_u Wc_{ui}$ avec u représentant les tâches déjà sélectionnées.

Une fois les tâches sélectionnées, il faut les affecter. Pour cela nous affectons, en suivant l'ordre de la liste de priorité des tâches, chaque tâche sur sa meilleure machine (plus petit poids de production Wp_{im}) encore libre.

Étape 3 : Liste de priorité sur les machines

Pour une tâche donnée, les temps de production et de changement ainsi que les coûts associés varient en fonction de la machine considérée et de sa production en cours. Nous utilisons, l'indicateur TC, comme défini plus haut.

Pour chaque tâche nous classons les machines en quatre catégories : très bonne, bonne, mauvaise et très mauvaise suivant la valeur de l'indicateur et une échelle 0-1 construite entre la meilleure machine (plus petit TC_{im}) et la plus mauvaise (plus grand TC_{im}). Un indicateur entre 0 et I_1 classera une machine dans la catégorie des très bonnes. Entre I_1 et I_2, dans la catégorie des bonnes; entre I_2 et I_3 dans les mauvaises et enfin entre I_3 et 1 dans les très mauvaises.
Au départ, le poids de changement est calculé à partir de la tâche initiale. Ensuite au cours de l'algorithme, chaque fois qu'une production est affectée à une machine, il faut recalculer les listes de priorité des machines.

Étape 4 : Procédure d'affectation

C'est de là que vient l'originalité de cet algorithme. Cette procédure gloutonne affecte des tâches aux machines jusqu'à ce que $N - M - 1$ tâches soient affectées.

La procédure sélectionne parmi les tâches non encore affectées, celle qui ira sur la prochaine machine libre.

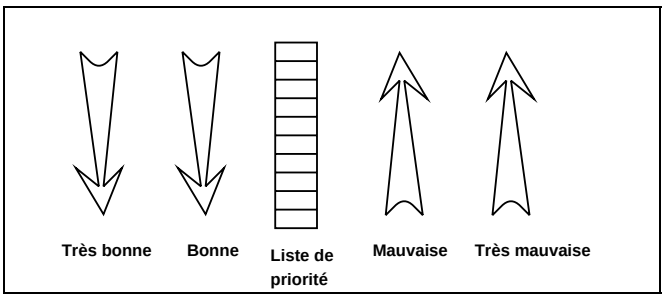


FIG. 3.5 - *Schéma de la procédure d'affectation*

Elle fonctionne de la manière suivante: dans l'ordre de la liste de priorité sur les tâches nous recherchons la première tâche ayant classé la nouvelle machine libre comme très bonne. Si une tâche est trouvée, elle est affectée. Sinon, nous recherchons la première tâche ayant classé la machine comme bonne. De même si une tâche est trouvée, elle est affectée. Dans le cas contraire, cela veut dire que l'on va affecter une tâche sur une mauvaise machine. Il est alors préférable de sélectionner une tâche moins prioritaire (au sens de la liste des priorités créée). Nous regardons donc dans l'ordre inverse de la liste de priorité des tâches la première qui a classé la machine dans la catégorie des mauvaises (ou à défaut des très mauvaises). La figure 3.5 illustre le fonctionnement de cette procédure.

Étape 5 : Affectation des $M-1$ dernières tâches

Comme les ordonnancements des machines sont construits simultanément, la charge de travail des machines est bien équilibrée. La procédure d'affectation autorise parfois qu'une tâche soit affectée à une mauvaise machine. Lorsque cela se produit en cours d'ordonnancement, ce n'est pas trop grave car cela permet d'occuper une machine au lieu de la laisser libre, par contre, en fin d'ordonnancement, affecter une tâche à une mauvaise machine pourrait entraîner de grandes disparités au niveau de la charge de travail et des coûts inutiles. Nous avons donc choisi d'affecter les dernières tâches différemment.

Nous considérons les tâches non encore affectées dans l'ordre de la liste de priorité et les affectons à leur meilleure machine, en tenant compte des tâches que nous venons d'affecter bien sûr. Ainsi cela permet d'éviter une trop grande dégradation de l'ordonnancement occasionné par le manque de choix dans la sélection des tâches à la fin de l'ordonnancement.

3.6.1.3 Mise en oeuvre

Différents paramètres interviennent dans l'heuristique *priorités*. En outre, il est nécessaire de déterminer les valeurs de I_1, I_2, I_3 qui permettent de calculer, pour

chaque tâche, sa liste de priorité des machines. Nous avons donc exécuté l'heuristique avec différentes valeurs de ces paramètres et comparé la qualité de la solution obtenue. L'expérience, montre que les valeurs $I_1 = 0.05$, $I_2 = 0.2$ et $I_3 = 0.5$ donnent de meilleurs résultats dans l'ensemble. Ces coefficients permettent, en effet, d'affecter à la machine qui vient de se libérer, en priorités une tâche pour laquelle la machine est très bonne.

3.6.2 Heuristique *"Regrets"*

Le principe de cette heuristique est simple. Dès qu'une machine a terminé une tâche, nous lui affectons la tâche de plus petit regret de la façon suivante. Soit m la machine venant de se libérer. L'heuristique calcule pour toutes les tâches non encore affectées leur poids d'affectation sur cette machine m.
Le poids d'affectation de la tâche i sur la machine m, PA_{im} est alors calculé suivant une combinaison linéaire du temps et du coût, comme dans l'équation 3.16, par :

$$PA_{im} = \alpha TA_{im} + (1 - \alpha)CA_{im} \tag{3.20}$$

Puis, pour chaque tâche, non encore affectées, l'heuristique recherche son poids d'affectation minimal $PAmin_i$, sur l'ensemble des machines, avec :

$$PAmin_i = \min_m [PA_{im}] \tag{3.21}$$

Enfin, l'heuristique sélectionne la tâche de regret minimal et l'affecte à la machine m. Le regret, pour une tâche i est défini par l'équation 3.22 et représente la différence entre le poids d'affectation sur la machine m (la machine pour laquelle on recherche une tâche) et le poids d'affectation minimal.

$$R_{im} = PA_{im} - PAmin_i \tag{3.22}$$

Nous faisons remarquer ici, que le regret d'une tâche sur une machine évolue en fonction de l'ordonnancement en cours de construction. En effet, le poids d'affectation d'une tâche sur une machine dépend de la dernière tâche affectée à cette machine, et doit donc être recalculé à chaque nouvelle affectation.

Pour connaître la prochaine machine qui va se libérer, nous tenons à jour une liste composée de l'ensemble des machines avec leur date de libération, triées selon cette date. Ainsi, à un instant donné, le prochain processeur qui va se libérer est le premier processeur de la liste.

L'algorithme est donné ci-après :

Initialisations
Mettre toutes les machines dans la liste des libérations
Affectations
Tantque il existe des tâches non affectées
Pour la prochaine machine m qui se libère
Pour toutes les tâches i non affectées
Pour toutes les machines m'
calculer le meilleur poids d'affectation de i sur m'
FinPour
calculer le poids PA_{im} d'affectation de i sur m
calculer le regret de la tâche i ($R_{im} = PA_{im}$ - $PAmin_i$)
FinPour
Affecter la tâche de regret minimal
FinPour
Mettre à jour la liste des libérations des machines
FinTque

Heuristique basée sur des regrets

3.6.3 Heuristique *"Cible"*

Cette heuristique, inspirée de l'heuristique "TARGET" développée par Martello, Soumis et Toth [99] pour le problème $R//Cmax$, essaie de trouver un ordonnancement dont la longueur ne dépasse par une certaine borne, que nous associons ici à la période. Si lors de la construction, une machine dépasse la borne initialement prévue, la durée de production de cette machine devient alors la nouvelle borne à ne pas dépasser. L'heuristique de construction procède à des ajouts en fin d'ordonnancement en se basant sur des pénalités calculées à partir d'un score d'affectation qui tient compte de la charge des machines et des possibilités d'affectation future. Au cours de la recherche de l'ordonnancement, nous notons B_m la capacité restante sur la machine m, c'est à dire la différence entre la cible visée et la longueur de production de la machine, et J_R l'ensemble des tâches non encore affectées.
Les scores d'affectation sont calculés à l'aide du critère de recherche de type $\alpha Temps + \beta Coûts$, en utilisant la démarche de Martello et al. Ils sont définis de la façon suivante (TA_{im} et CA_{im} étant les temps et coûts d'affectation définis précédemment).

$$S_{im} = \tag{3.23}$$
$$\begin{cases} \alpha TA_{im} + (1-\alpha)CA_{im} & \text{Si } TA_{im} + \min_{k\in J_R} TA_{km} \leq B_m \\ \alpha B_m + (1-\alpha)CA_{im} & \text{Si } TA_{im} + \min_{k\in J_R} TA_{km} \geq B_m \geq TA_{im} \\ \alpha(TA_{im} - B_m) \times Q + (1-\alpha)CA_{im} & \text{Si } TA_{im} \geq B_m \end{cases}$$

Le premier calcul du score correspond au cas où il est encore possible de mettre au moins une autre tâche après l'affectation de i sur m. Le deuxième cas, dont le score est plus élevé, représente le cas où le placement de i sur m empêchera tout autre placement sans dépassement de la cible. Enfin, le troisième cas, correspond au cas, où l'affectation de i sur m entraîne un dépassement de la cible. Il est fortement pénalisé, puisque le temps dépassé est multiplié par Q, une grande valeur, lors du calcul du score.

Ayant calculé les scores d'affectation, il est possible de calculer les pénalités, comme étant la différence entre les deux meilleurs scores (pénalités proposées par Martello et al. [99]).

$$\Delta_i = \min_m 2S_{im} - \min_m S_{im} \tag{3.24}$$

avec $\min 2_m S_{im}$ représentant le deuxième minimum.

L'algorithme sélectionne la tâche de plus forte pénalité, qui représente la tâche qui n'étant pas affectée à la machine lui donnant un score minimal, serait la plus pénalisante.

A chaque étape, Martello, Soumis et Toth proposent de ne recalculer les scores que pour la machine m ayant reçu une nouvelle tâche, car le temps libre de cette machine B_m est diminué et les scores s'en trouvent modifiés. Cela permet également, lorsque des temps et coûts de changement existent, de les considérer en recalculant les scores dès qu'une machine a eu une nouvelle affectation.
Pourtant, en effectuant les mises à jour des scores d'affectation que pour la machine ayant eu une nouvelle affectation, il est possible de perdre en précision car l'ensemble des tâches restantes (J_R) a été modifié par l'affectation, et il se peut que des scores le soit également, puisque pour les autres machines, $\min_{k \in J_R} TA_{km}$ peut avoir changé. Ainsi, il semble nécessaire de recalculer l'ensemble des scores (pour toutes les machines) dès qu'une affectation a été réalisée.

Pour connaître le prix à payer (en terme de temps de calcul) pour recalculer tous les scores à chaque étape, nous avons expérimenté les deux versions de cette heuristique. Cible1 représente la version avec recalcul complet à chaque étape, tandis que Cible2 ne recalcule les scores que pour la machine venant de recevoir une nouvelle affectation. Nous avons comparé pour différents problèmes la qualité de la solution et le temps de calcul de chacune des deux versions. Cette expérimentation a été menée dans un objectif de minimisation de la durée totale de production (c'est-à-dire $\alpha = 1$). Le tableau 3.6 reportent les résultats (moyenne de dix exécutions).

Ce tableau nous montre que Cible2 est beaucoup plus rapide que Cible1, ce qui est

Problème (mach,prod)	Cible 1		Cible 2	
	Cmax	Temps	Cmax	Temps
5-100	1245	1,3	1231	0,3
12-90	474	2,1	469	0,5
15-80	350	4,6	351	1,0
20-200	643	71,9	620	5,9

TAB. 3.6 - *Comparaisons du Cmax et des temps d'exécution en secondes*

normal puisque les calculs sont moins nombreux. Une autre constatation, beaucoup plus surprenante est que Cible2 donne dans la majorité des cas de meilleurs résultats que Cible1. Ceci n'est pas seulement vérifié pour les exemples reportés dans le tableau, mais se généralise lorsque l'on compare les deux heuristiques sur d'autres exemples.

Nous avons donc choisi d'utiliser l'heuristique Cible2, que nous appelerons désormais simplement *Cible*, pour la suite de notre expérimentation. L'algorithme de l'heuristique est le suivant :

Initialisation du temps libre initial pour chaque machine
  Pour toutes les machines m
    B_m = T /* Temps libre = période */
    Pour toutes les tâches i
      Calculer le score d'affectation S_{im} de i sur m
    FinPour
  FinPour
Boucle d'affectation
  Tantque il existe des tâches non affectées
    Pour toutes les tâches non affectées i
      Calculer la pénalité Δ_i de i
    FinPour
    Sélectionner la tâche de plus forte pénalité,
    l'affecter à sa machine m' de score minimal
    Pour toutes les tâches non affectées i
      Calculer le score d'affectation $S_{im'}$ de i sur m'
    FinPour
  FinTque

Heuristique Cible

3.6.4 Comparaisons

Afin de choisir l'heuristique à utiliser pour résoudre le problème $R/S_{ij}, T/\sum(Coûts)$, nous avons expérimenté, sur plusieurs jeux de données et sur plusieurs types de problèmes, les différentes heuristiques. Nous avons utilisé, pour la recherche de solutions, un indicateur sous la forme d'une combinaison linéaire du temps et du coût (comme présenté par l'équation 3.16), et pour l'évaluation, soit seulement le temps ou seulement le coût, soit les deux critères en utilisant la définition de l'optimalité (définition 5) présentée lors de l'explication du problème multicritère.

Pour une meilleure comparaison, nous avons mis en parallèle les heuristiques proposées dans la section précédente avec les bornes calculées suivant la section 3.3 et une heuristique *"Aléa"* construisant un ordonnancement aléatoirement.

3.6.4.1 Génération des jeux de données

Pour expérimenter les différentes heuristiques, nous avons généré plusieurs types de problèmes. Il convient, avant de décrire l'expérimentation de donner les caractéristiques des problèmes générés.

Pour générer un problème, le programme questionne l'utilisateur sur le nombre de processeurs et de tâches qu'il désire considérer. Ensuite, à partir de ces deux paramètres les données sont générées aléatoirement (ou plus exactement, aussi aléatoirement que le permet la fonction rand du C++) de la façon suivante ($\mathcal{U}$ représente la loi Uniforme) :

Durées opératoires Fonction de la tâche et de la machine.
$Tp_{im} = \mathcal{U}[minop, maxop] = \mathcal{U}[25, 100]$

Temps de changement Fonction des produits enchaînés et de la machine.
$Tc_{ijm} = \mathcal{U}[minch, maxch] = \mathcal{U}[12.5, 50]$

Coûts De façon à être proche de la réalité, nous avons définis deux types de coûts : des coûts horaires (Ch_m), dépendants des processeurs et représentant les coûts relatifs à l'utilisation du processeur et des coûts fixes (Cf_{im}), dépendants de tâches et des processeurs et représentant les coûts de transport, par exemple. Il sont générés de la manière suivante :
$Ch_m = \mathcal{U}[2, 4.5]$
$Cf_{im} = \mathcal{U}[375, 1500]$
Ainsi,
les couts de production $Cp_{im} = Tp_{im} \times Ch_m + Cf_{im}$ varient entre $[425, 1950]$
et les coûts de changements $Cc_{ijm} = Tc_{ijm} \times Ch_m$ varient entre $[25, 225]$

Période La longueur de la période est calculée en fonction du nombre de tâches et de processeurs de façon à générer, soit des périodes réalisables, soit non réalisables (en fonction des besoins de l'expérimentation).
$T_{réalisable} = (N/M + 1) \times (maxop + maxch)/2.5$
$T_{nonréalisable} = T_{réalisable}/2$

3.6.4.2 Aspects monocritères

Dans cette partie, nous évaluons les heuristiques en considérant les deux aspects temps et coûts séparément.

Minimisation de la durée totale de production

C'est le cas où α est égal à 1. Les coûts n'interviennent pas dans la recherche de l'ordonnancement. Seul le temps a de l'importance. Ces tests ont été menés pour deux types de problèmes; des problèmes où la période est facile à respecter, et des problèmes ou la période n'est pas respectable (période inférieure à la borne inférieure du makespan).

Les résultats, en terme de longueur des ordonnancements obtenus par les différentes méthodes, sont reportés dans le tableau 3.7, lorsque la période est respectable. Les valeurs indiquées dans ce tableau représentent une moyenne sur dix instances par problème.

	Problème	Période	Borne	Alea	Priorités	Regret	Cible
P1	5-80	1020	776	1553	1163	1064	<u>1007</u>
P2	12-90	480	305	778	496	<u>462</u>	474
P3	15-80	360	216	588	346	<u>331</u>	349
P4	20-200	660	390	1044	636	<u>558</u>	629
P5	30-200	420	248	730	408	<u>379</u>	412

TAB. 3.7 - *Comparaisons du Cmax des solutions des différentes heuristiques*

Ces résultats sont également transcrits par la figure 3.6, pour une meilleure lisibilité. Nous pouvons constater que l'ensemble des méthodes évolue de façon similaire d'un problème à l'autre. Pour les problèmes P2, P3, P4 et P5 l'heuristique des *"Regrets"* est la meilleure, et les deux autres heuristiques donnent des solutions très proches l'une de l'autre. Seul pour le problème P1, l'heuristique "Cible" donne la meilleure solution.

Un autre test a été réalisé dans le cas où les périodes ne sont pas respectables, pour voir si cela a une influence sur les qualités des solutions. Le tableau 3.8 en

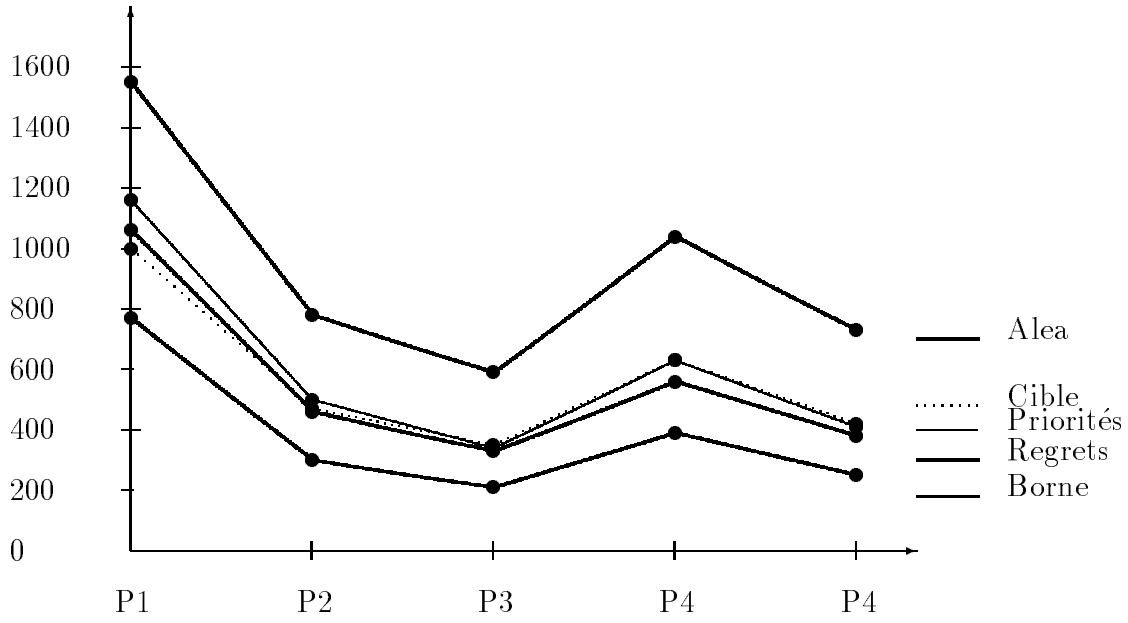


FIG. 3.6 - *Comparaison des heuristiques : cas où la période est respectable*

rapporte les résultats.

Dans ce cas, il s'avère que pour tous les problèmes, l'heuristique des regrets donne la meilleure solution. De plus, l'heuristique des priorités est généralement meilleure que l'heuristique "cible". Nous aurions pu nous attendre à une meilleure performance de l'heuristique "cible" qui est la seule parmi les trois à considérer la période visée. Mais il semble que la prise en compte des temps de changements diminue la performance de cette heuristique.

Il apparaît clairement, d'après cette expérimentation que l'heuristique *"Regrets"* est très intéressante d'autant plus qu'elle est également très rapide (5 fois plus rapide que l'heuristique "Cible", en moyenne, sur les exemples traités). Il semble donc que dans un objectif de minimisation de la durée totale de production, cette heuristique soit la plus performante des trois.

	Problème	Période	Borne	Alea	Priorités	Regret	Cible
P1	5-80	510	766	1566	1146	1040	1052
P2	12-90	240	304	792	496	452	522
P3	15-80	180	216	597	353	335	384
P4	20-200	330	388	1056	639	555	655
P5	30-200	210	248	732	409	382	491

TAB. 3.8 - *Comparaisons du Cmax lorsque la période n'est pas réalisable*

Minimisation des coûts

La deuxième expérience monocritère porte sur la minimisation des coûts (cas où $\alpha = 0$). Tout comme dans l'expérience précédente, nous avons comparé les différentes heuristiques pour plusieurs types de problèmes. Le tableau 3.9 donne les résultats.

	Problème	Borne	Alea	Priorités	Regret	Cible
P1	5-80	40 348	63 547	51 947	47 706	43 344
P2	12-90	38 869	72 473	51 617	47 848	42 030
P3	15-80	32 924	62 694	45 068	39 854	35 473
P4	20-200	79 034	156 959	111 014	95 217	85 229
P5	30-200	75 918	157 895	111 794	92 688	81 962

TAB. 3.9 - *Comparaisons des coûts des solutions trouvées par les différentes heuristiques*

Il apparaît fortement que dans ce cas, l'heuristique cible est la plus appropriée, puisque c'est elle qui fournit le meilleur résultat pour toutes les instances étudiées.

Il est alors difficile de choisir, aux vues des optimisations monocritères quelle est la meilleure heuristique. En effet, l'heuristique "*Regrets*" semble meilleure pour la minimisation de la durée totale de production tandis que l'heuristique "*Cible*" est meilleure pour la minimisation des coûts.

Voyons quel est le comportement de chacune de ces heuristiques en environnement bicritère, c'est à dire lorsque l'on regarde à la fois les qualités des solutions en terme de durée maximale de production et de coûts de la solution.

3.6.4.3 Aspects bicritères

Dans cette partie, nous étudions le comportement des heuristiques en environnement bicritère. Pour cela, nous avons exécuté les heuristiques sur les mêmes problèmes en faisant varier la valeur de α.

Comme nous l'avons signalé lors de la présentation de notre approche bicritère, il est nécessaire, pour pouvoir comparer du temps et des coûts dans une même fonction de recherche, de normaliser ces deux composantes. Pour cela, nous avons divisé chacune des composantes par la borne inférieure, calculée comme expliquée dans la section 3.3, la concernant.

Puis, nous avons relevé, en faisant varier α entre 0 et 1 la longueur de l'ordonnancement trouvé et le coût de la solution. Les résultats de cette étude, pour des instances avec 30 processeurs et 200 tâches, sont reportés dans les tableaux 3.10 et 3.11. Les figures 3.7 et 3.8 montrent plus précisément l'évolution des solutions pour les différentes valeurs de α.

α	Priorités	Regret	Cible
1	509	479	1842
0,9	433	381	569
0,8	431	376	544
0,7	432	374	505
0,6	435	381	544
0,5	440	380	541
0,4	438	367	598
0,3	435	377	512
0,2	438	374	526
0,1	436	376	526
0	436	375	538

TAB. 3.10 - *Comparaisons des Cmax des solutions trouvées en faisant varier* α

Ces résultats nous montrent qu'une variation de α dans l'intervalle $]0.5, 1]$ n'a pas beaucoup d'influence sur la qualité de la solution. Par contre, pour $\alpha \leq 0.5$, une variation de α a de fortes incidences, en terme de temps et de coûts, sur la solution.

Ainsi, en faisant varier la valeur de α, il est possible d'influer sur le type de la solution obtenue. Cela nous permet donc d'appliquer le schéma itératif proposé pour la recherche de solutions. Nous pouvons nous demander quel est l'intérêt de faire varier α entre $0,5$ et 1, puisque l'influence sur la qualité des solutions est très faible. En fait, les solutions générées dans cette fourchette sont, il est vrai de

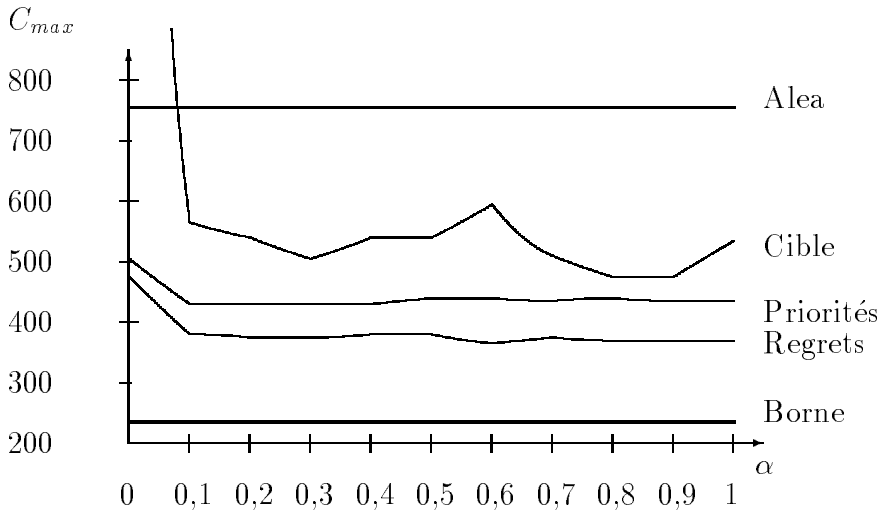


FIG. 3.7 - *Évolution de la valeur du cmax en fonction de* α

qualité semblables, mais sont différentes. Or n'oublions pas que les heuristiques ont comme objectif de proposer une première solution soumise ensuite à des améliorations. Ainsi, il est toujours intéressant de visiter des solutions différentes, pour permettre à la procédure d'amélioration d'avoir plusieurs points de départ, et ainsi plus de chances de trouver de bonnes solutions.

α	Priorités	Regret	Cible
1	112 197	93 311	82 760
0,9	132 135	113 125	105 521
0,8	136 511	120 811	116 672
0,7	137 966	124 311	120 991
0,6	138 515	127 700	124 785
0,5	137 601	129 225	126 110
0,4	139 098	128 561	127 611
0,3	139 484	129 213	128 028
0,2	140 099	130 118	128 121
0,1	139 638	129 827	128 121
0	139 638	130 205	129 485

TAB. 3.11 - *Comparaisons des coûts des solutions trouvées en faisant varier α*

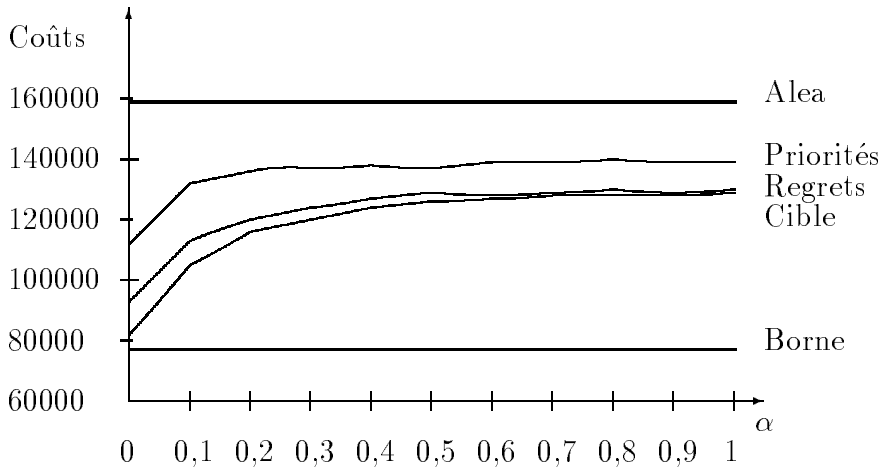


FIG. 3.8 - *Évolution de la valeur du coût en fonction de α*

3.7 Procédures d'amélioration

Les heuristiques présentées précédemment se basent sur un indicateur, constitué d'une combinaison linéaire de deux critères pour ordonnancer un ensemble de tâches sur un ensemble de processeurs parallèles. Cet indicateur peut, en fait, représenter différents critères (temps, coûts, combinaison temps et coûts) suivant les besoins de l'utilisateur. Elle sert à donner une direction de recherche à l'heuristique, mais ne sert en aucun cas à l'évaluation finale des solutions.

Ces méthodes sont heuristiques et leur point fort est leur rapidité. Mais l'inconvénient de l'utilisation de telles méthodes est que la qualité de leur solution n'est pas garantie et il est intéressant de chercher à améliorer ces solutions. Pour cela, nous proposons de nous inspirer de méthodes d'optimisation locale basées sur des échanges de tâches (méthodes r-opt, par exemple — voir 1.4.3.1 —). La méthode proposée est de type descente, car elle se dirige toujours vers une meilleure solution et n'autorise jamais la solution à se dégrader. Nous avons opté pour ce type de méthodes, car elles commencent par descendre très rapidemment. Bien sûr, ces méthodes restent bloquées dans un optimum local, mais elles permettent, lorsque le temps imparti est court, de trouver de bonnes solutions. Les autres méthodes de voisinage, plus complexes (recherche tabou, recuit simulé, algorithmes génétiques,...) sont plus efficaces pour la recherche de solutions globales, mais demandent beaucoup plus de temps, et ne progressent pas très rapidement au début.

C'est donc l'amélioration rapide qui nous a séduit dans les méthodes de descente, puisque dans notre résolution du problème par le schéma itératif faisant varier α, nous allons exécuter plusieurs fois cette procédure d'amélioration (à partir de solution de départ différentes).

Kedad et al. [83] distinguent, pour un problème d'ordonnancement sur machines parallèles semblable, deux types d'amélioration à étudier : les améliorations réalisées au sein même d'une séquence de tâches (c'est à dire sur un seul processeur) et les améliorations entre séquences par échange de tâches, et propose de les mettre en œuvre conjointement dans une procédure hybride. Nous nous sommes inspirés de cette façon de procéder pour proposer également une procédure hybride regroupant différents types d'amélioration.

Au cours de la recherche, la performance de la solution trouvée est calculée en fonction de la définition de l'optimalité des solutions (Définition 5). La meilleure solution (ou les meilleures, lorsque la période n'est pas respectée) est gardée en mémoire et mise à jour dès qu'une meilleure solution est trouvée.

3.7.1 Améliorations intra-séquences

Au sein d'une séquence, les gains potentiels se trouvent dans la minimisation des temps et coûts de changement, puisque tout ce qui est relatif à l'affectation aux machines (temps et coûts de production, coûts de distribution) ne peut être modifié par une modification de l'ordre d'exécution des tâches sur la machine. La séquence déterminée pour chaque processeur par l'heuristique de construction n'est pas forcément optimale et une simple réorganisation des tâches sur chaque processeur peut permettre de gagner du temps et de diminuer les coûts en ré-ordonnant mieux les tâches de façon à diminuer les temps et coûts relatifs aux changements.

Ce problème, note $1/S_{ij}/F$, avec F la fonction objectif visée a été très étudiée dans la littérature. Nous citons, comme référence, la thèse de Artigues [3] qui expose un état de l'art complet sur les problèmes à une machine avec temps de changement, ainsi qu'un article récent de Williams et Wirth [142] qui s'intéressent aux changement indépendant de la séquence.

Ce problème, comportant des temps de changement dépendants de la séquence, peut également être assimilé au Problème du Voyageur de Commerce Asymétrique [32]. Nous ne reportons pas ici de revue détaillée sur la littérature, tant celle-ci est vaste, mais citons néanmoins l'ouvrage de référence de Lawler et al. [93], l'article de Margot [98] qui propose une structure de donnée permettant d'exécuter une optimisation de type p-opt en $O(\log n)$ pour un problème à n noeuds, et l'article de Gendreau et al. [55] qui exposent leur méthode, combinant une procédure d'insertion et d'optimisation, GENIUS. Leur méthode d'insertion s'appuie sur le principe que l'insertion d'une ville dans une tournée ne se fait pas forcément entre deux villes adjacentes. Ils proposent deux types d'insertion. De même, pour l'optimisation, ils procèdent à des échanges en enlevant une ville d'un tour et en la réinsérant ailleurs en suivant leur principe d'insertion.

Nous proposons d'effectuer le ré-ordonnancement au sein des séquences de deux façons différentes en fonction du nombre de tâches composant la séquence. Soit le nombre de tâches est relativement faible (≤ 10), nous pouvons alors recourrir à une méthode exacte (PSE : Procédure de Séparation et Evaluation), soit le nombre de tâches est élevé, et nous proposons de procéder à des échanges en déplaçant une tâche d'une place à une autre.

La procédure d'échanges intra-séquence proposée est décrite ci-après et peut être appliquée à toutes les séquences.

Si $NbProd < N_{pse}$
  alors exécuter une procédure de séparation et évaluation
  sinon
    $NbIter = 0$
    Tque $NbIter < N_{max}$
      Prendre une tâche et la placer ailleurs (sur la même séquence)
      Si amélioration ou solution non dominée
        alors garder la solution dans les solutions candidates
          remplacer la séquence courante par celle trouvée
          $NbIter = 0$
        sinon $NbIter = NbIter + 1$
      FinSi
    FinTque
FinSi

Procédure d'échanges intra-séquences

	$NbProd$	:	Nombre de tâches de la séquence.
	N_{pse}	:	Nombre maximal de tâches pour lequel on fait une PSE.
avec :	$NbIter$	:	Nombre d'itérations depuis la dernière amélioration.
	N_{max}	:	Nombre maximal d'itérations autorisées, ne donnant pas d'améliorations.

3.7.2 Améliorations inter-séquences

Les affectations des tâches aux processeurs sont déterminées par une heuristique de construction qui s'appuie sur un indicateur. De cette affectation, dépendent les temps et coûts relatifs à la production et les coûts de la distribution. Pourtant il se peut que des tâches soient affectées à de mauvais processeurs. En effectuant des échanges de tâches entre processeurs, nous pouvons permettre à des tâches mal placées d'être affectées à de meilleurs processeurs.

Pour effectuer ces échanges inter-séquences nous procédons en trois étapes : tout d'abord, sélection des deux séquences de tâches concernées, puis choix du type d'échange et enfin détermination des positions concernées par l'échange.

Nous proposons de procéder à deux types d'échanges (voir figure 3.9) :

Tout d'abord, il est possible d'échanger deux tâches de deux séquences de processeurs différents. Les deux tâches sélectionnées échangent leur place respective. Leurs temps d'exécution ainsi que les temps de changement les précédant et les

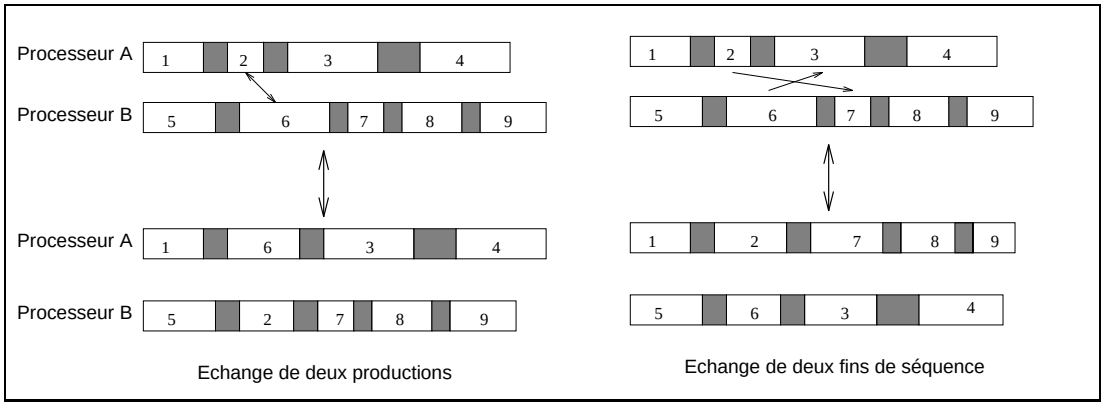


FIG. 3.9 - *Echanges inter-séquences*

succédant varient. Il est ainsi possible d'obtenir un meilleur ordonnancement pour l'un des deux processeurs, voire pour les deux.
La deuxième possibilité d'échange consiste à intervertir les fins de séquences de deux processeurs. De nouveau, les temps d'exécution et de changement, ainsi que les coûts diffèrent. Ce deuxième type d'échange a l'avantage de pouvoir faire varier le nombre de tâches par processeur et évite de rester dans un voisinage trop restreint.

La procédure d'échange inter-séquence proposée est la suivante :

$NbIter = 0$
Tque $NbIter < MaxIter$
    Sélectionner deux processeurs
    $NbEch = 0$
    Tque $NbEch < MaxEch$
        Procéder à des échanges des deux types
        Si amélioration ou solution non dominée
            alors considérer la solution comme solution candidate
                remplacer la solution par la nouvelle
        FinSi
        $NbEch = NbEch + 1$
    FinTque
    $NbIter = NbIter + 1$
FinTque

Procédure d'échanges inter-séquences

$NbIter$: Compteur du nombre d'itérations (sélections).
$MaxIter$: Nombre maximal de sélections à faire.
avec : $MaxEch$: Nombre maximal d'échanges à réaliser.
$NbEch$: Compteur du nombre d'échanges exécutés à partir du couple sélectionné.

3.7.3 Performances de la procédure d'amélioration

Dans cette partie, nous expérimentons la procédure d'amélioration appliquée aux solutions construites à l'aide des heuristiques. Cette expérimentation a deux objectifs : 1) évaluer l'amélioration apportée par l'application de la procédure d'échanges et le temps optimal de recherche, 2) déterminer quel est le meilleur couple heuristique de construction - procédure d'amélioration à utiliser dans notre Oracle.

Ainsi, nous avons appliqué sur plusieurs types de problèmes les heuristiques de contruction couplées à la procédure d'amélioration. Là encore, nous avons commencé par étudier le comportement en environnement monocritère.

Les deux figures suivantes 3.10 et 3.11 représentent, pour des problèmes à 30 processeurs et 200 tâches, respectivement, l'évolution de la durée totale de production de la solution courante dans un contexte de minimisation du C_{max} et l'évolution du coût de la solution dans un contexte de minimisation des coûts, en fonction du temps, par l'application de la procédure d'amélioration.

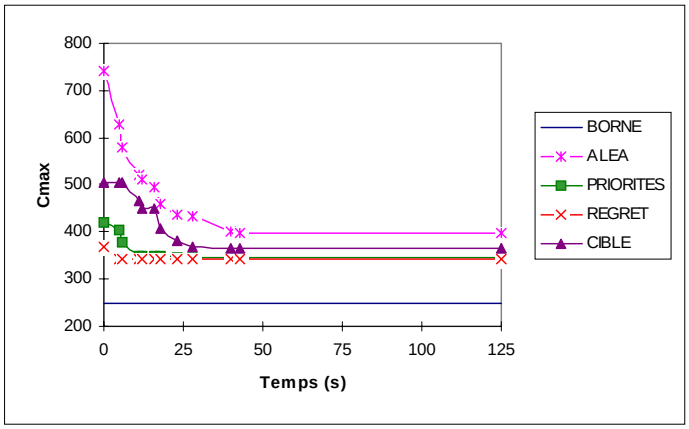


FIG. 3.10 - *Évolution du Cmax en fonction du temps (application de la procédure d'amélioration)*

La première remarque que nous inspire ces figures est que très rapidement la solution converge vers un optimum local. Il apparaît alors inutile d'exécuter la

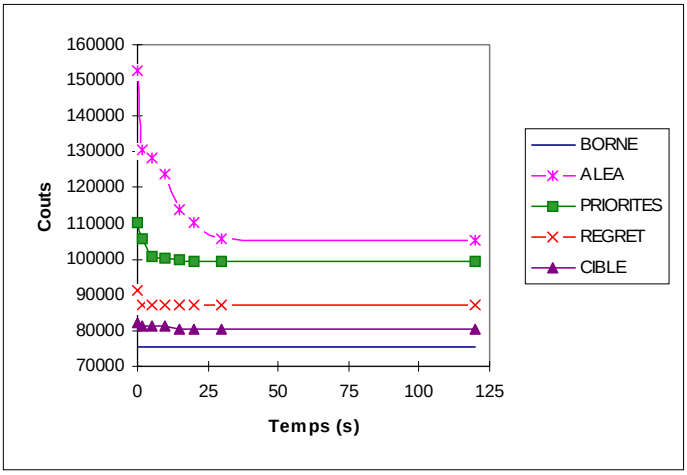


FIG. 3.11 - *Évolution du coût de la solution en fonction du temps (application de la procédure d'amélioration)*

procédure d'amélioration trop longtemps, puisque au bout d'un moment elle n'apporte plus d'amélioration.

La deuxième remarque est que lorsque la solution de départ est mauvaise (cas de la procédure *"Aléa"*, par exemple), la procédure d'amélioration met beaucoup de temps (lorsqu'elle y arrive) pour améliorer la solution et atteindre la qualité de départ des solutions données par les autres heuristiques. Il est donc important, de commencer la procédure d'amélioration avec la meilleure solution possible.

Enfin, la troisième remarque est que lors de l'amélioration, les solutions des différentes heuristiques restent dans le même ordre. Ainsi, si la solution donnée par l'heuristique des regrets est meilleure dans le cas de la minimisation de la durée totale de production, elle a de fortes chances d'être la meilleure après l'application de la procédure d'amélioration.

3.7.4 Conclusions pour le choix de l'Oracle

Ces différentes raisons font qu'il semble opportun de choisir comme point de départ de la procédure d'amélioration la meilleure solution fournit par les différentes heuristiques. Il est alors possible d'exécuter chacune des heuristiques et de sélectionner la meilleure solution, mais comme nous allons itérer ce processus plusieurs fois, il est préférable, pour gagner du temps et de la mémoire, d'exécuter une seule des heuristiques. Aux vues des résultats, nous proposons de choisir l'heuristique *"Regrets"* lorsque la durée totale de production devient l'aspect à privilégier, et l'heuristique *"Cible"* lorsque ce sont les coûts qui doivent être privilégiés.

Ainsi, notre Oracle de résolution à utiliser dans la procédure itérative sera com-

posé, suivant la situation, soit de la procédure des regrets, soit de la procédure cible, couplées à la procédure amélioratrice.

3.8 Résolution du problème initial

Toutes les expérimentations précédentes nous permettent maintenant de nous attaquer au problème étudié à savoir le problème d'ordonnancement de machines parallèles non liées avec temps de changement dépendant de la séquence entre les tâches, respect d'une période et minimisation de l'ensemble des coûts, que nous avons noté $R/s_{ij}, C_{max} \leq T/\sum(Coûts)$.

3.8.1 Méthode proposée

Nous proposons d'appliquer, pour résoudre ce problème, la procédure itérative exposée dans ce chapitre en utilisant pour l'Oracle la composition donnée dans la partie précédente. En fonction du résultat trouvé pour une solution, la procédure simule le comportement d'un décideur et détermine la réaction à avoir, c'est à dire la valeur à donner à α, de façon à espérer trouver une meilleure solution.
Nous rappelons ici la procédure itérative en détaillant la composition de l'Oracle.

Initialisations
  $\alpha = 0.5$, $Nbiter = 0$.
  Exécuter l'heuristique *"Regrets"*
  Améliorer la solution, sauver la ou les meilleures solutions
Affectations
  Tantque $Nbiter < N_{max}$
    Si $Cmax \leq T$
      alors $\alpha = \alpha/2$.
      sinon,
        Si $Cmax > T$
          alors $\alpha = (1 + \alpha)/2$.
        FinSi
    FinSi
    Si $\alpha \geq 0.5$ /* On favorise le temps */
      alors Exécuter l'heuristique *"Regrets"*
      sinon Exécuter l'heuristique *"Cible"*
    FinSi
    Améliorer la solution, sauver la ou les meilleures solutions
    $Nbiter = Nbiter + 1$
  FinTantque

Procédure itérative de résolution

où : $Cmax$: Durée totale de l'ordonnancement.
T : Longueur de la période.
Nbiter : Nombre d'itérations.
N_{max} : Nombre maximal d'itérations.

3.8.2 Expérimentation

Afin de valider la procédure itérative proposée, nous reportons ci-dessous les résultats obtenus pour la résolution d'un problème à 20 processeurs et 200 tâches. Différentes configurations sont étudiées : 1) cas où la période est facile à respecter, 2) cas où la période n'est pas respectable (la période est inférieure à la borne inférieure de la durée totale de production), 3) cas où la période est difficilement respectable, mais peut l'être. 30 secondes d'optimisation sont accordées pour chaque itération.

Les trois figures suivantes 3.12, 3.13 et 3.14 représentent, pour les trois cas cités ci-dessus, l'évolution du C_{max} et du coût de la solution courante, en fonction du temps, lors de la résolution du problème. Pour lire les valeurs réelle des coûts, il faut multiplier par 100 la valeur lue sur l'axe des ordonnées. Sur ces figures, les traits pointillés verticaux indiquent le moment où une nouvelle itération commence (c'est-à-dire, le moment où l'on relance l'heuristique avec une nouvelle valeur de α. Dans le cas où aucune solution réalisable n'est trouvée, plusieurs solutions candidates sont retenues (les solutions non dominées par rapport au temps et au coût). Nous avons représenté, sur les figures, par des flèches la (ou les) meilleure(s) solution(s) retenue(s) à la fin de l'exécution.

La figure 3.12 montre que lorsque la période est facile à respecter, la procédure favorise une diminution des coûts en diminuant la valeur de α et en acceptant ainsi d'augmenter la durée totale de production tout en respectant la période. Ainsi, une bonne solution en terme de coût (90 194 pour une borne inférieure de 80 509, soit une augmentation de 12%) peut être trouvée.

La figure 3.13 illustre le cas où la période n'est pas respectable. Dans ce cas, la procédure a beaucoup de mal à trouver une bonne solution. Les quatre flèches représentent les quatre solutions non dominées résultant de l'application de la procédure. Le rond indique la solution correspondant à l'ordonnancement le plus court (d'une longueur de 502, pour une borne inférieure de 387) pour un coût de 131 132 pour une borne inférieure de 76 862. Ainsi, la solution trouvée est 30% au dessus de la borne du C_{max} et à 70% au dessus de la borne sur les coûts. Ce cas est le plus difficile à résoudre.

Enfin, la figure 3.14 est certainement la plus intéressante. Elle montre comment une solution peut être chahutée de façon à parcourir un ensemble de solutions

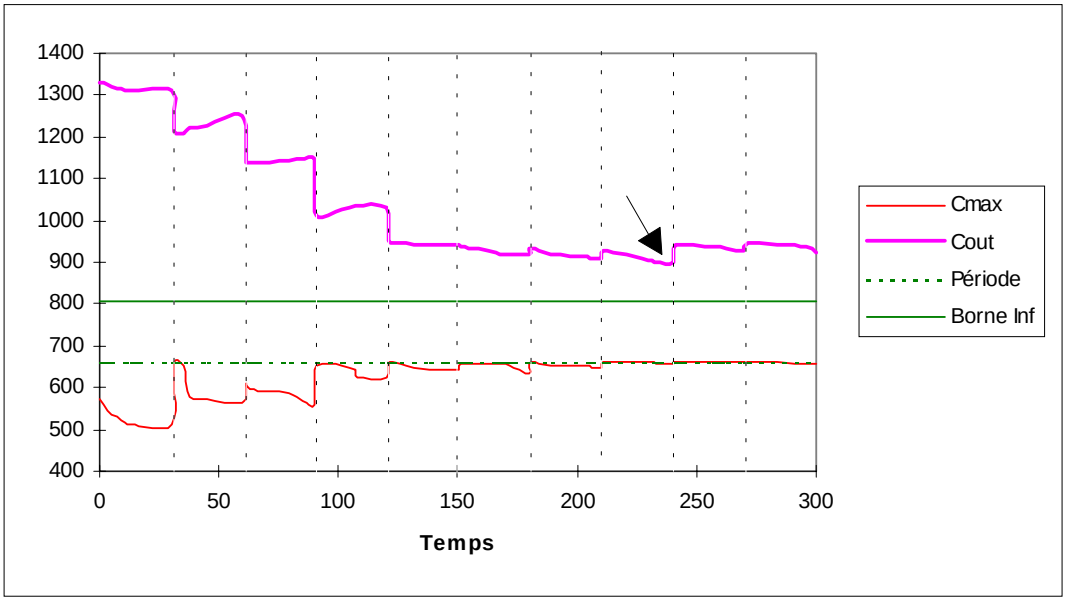


FIG. 3.12 - *Évolution du coût et du Cmax de la solution en fonction du temps : cas d'une période facile à respecter*

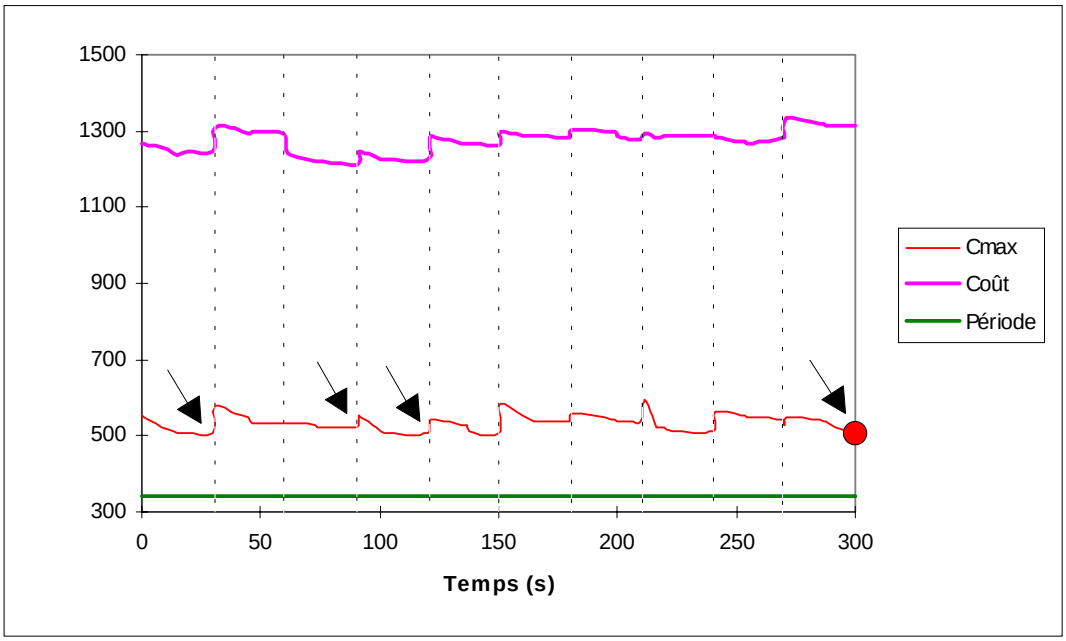


FIG. 3.13 - *Évolution du coût et du Cmax de la solution en fonction du temps : cas d'une période non respectable*

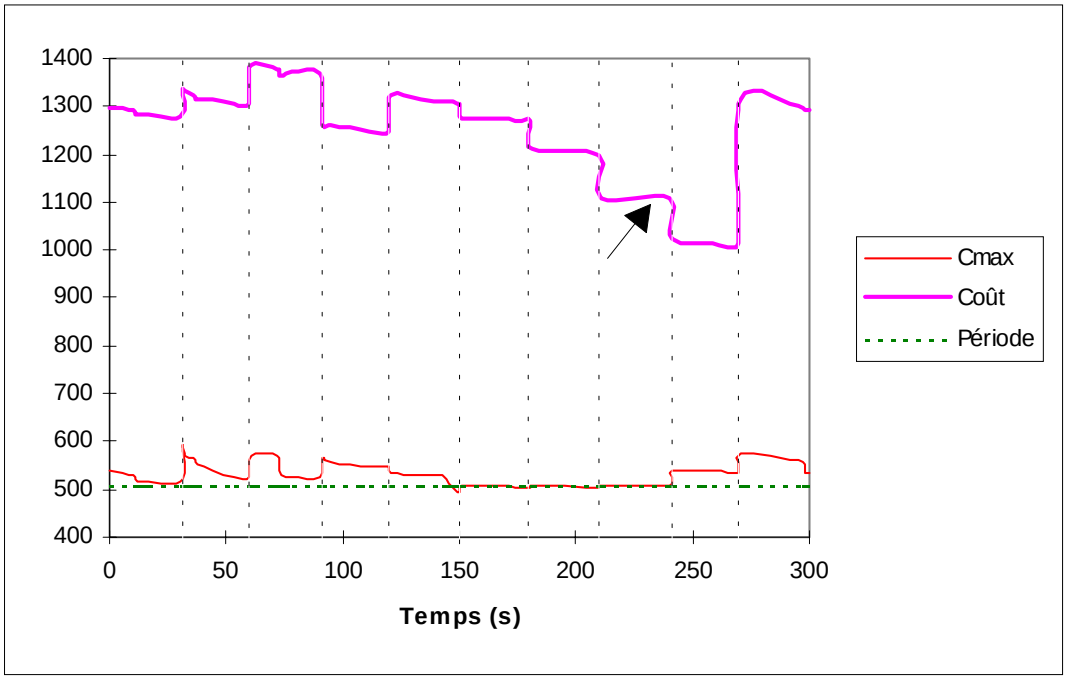


FIG. 3.14 - *Évolution du coût et du Cmax de la solution en fonction du temps : cas d'une période difficilement respectable*

intéressant. Finalement la meilleure solution trouvée fait exactement la longueur de la période (507) et son coût est de 110 611 pour une borne inférieure de 80 184, soit une augmentation de 38%.

3.8.3 Conclusions

D'après les expérimentations reportées ici, la méthode itérative proposée semble être efficace, surtout lorsque la période est respectable (facilement ou non). Dans ces deux cas, la procédure arrive, en faisant varier la valeur de α, à se rapprocher au maximum de la longueur de la période de façon à minimiser les coûts. La figure 3.15 illustre ce fait pour un exemple à 5 processeurs et 40 tâches. Nous avons choisi ce petit exemple pour pouvoir représenter tous les points. Nous avons représenté, pour les cinq premières itérations, menant à la solution optimale (temps 468, coût 23 461), les principaux points par lesquels passe l'heuristique.
Nous pouvons voir que l'heuristique profite de la marge qui existe, entre la solution la plus courte et une solution plus longue qui respecte la période, pour affecter les tâches de manière à diminuer les coûts. Lorsque la période est impossible à respecter, le comportement de l'heuristique est un peu moins probant, pourtant elle se dirige bien vers des solutions de plus en plus courtes puisque la meilleure solution, en terme de temps, est trouvée à la fin de la procédure.

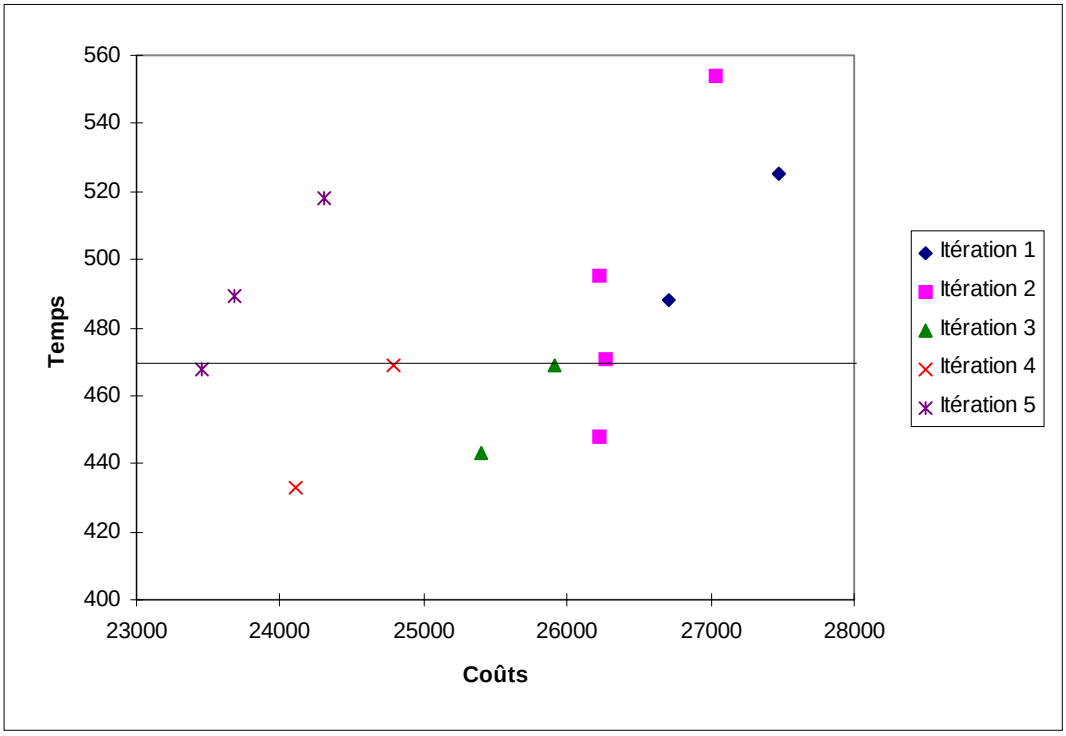


FIG. 3.15 - *Ensemble des points parcourrus par l'heuristique à chaque itération*

Les résultats de cette méthode systématique semblent satisfaisants et justifient le choix d'une telle méthode. Nous avons choisis de proposer, ici, une méthode utilisant un mécanisme de contrôle systématique pour faire varier la valeur du poids α. Souvent, pour aborder un problème multicritère, on préfère plutôt proposer des méthodes interactives qui font intervenir un décideur en lui demandant de réagir face aux solutions proposées. Dans notre cas, on pourrait lui demander de réévaluer, à chaque itération, l'importance qu'il donne au temps par rapport aux coûts aux vues des solutions précédentes, ce qui revient à donner une nouvelle valeur à α. Ici, nous avons, en fait, simulé un décideur qui modifierait les poids tel que nous l'avons proposé, pour permettre d'exécuter la procédure un grand nombre de fois. Transformer cette procédure systématique en une procédure itérative est immédiat et peut être fait si l'utilisateur en exprime la demande.

3.9 Conclusion

Nous avons vu, dans ce chapitre, différentes facettes d'un problème d'ordonnancement sur processeurs parallèles non liés avec temps de changement dépendant de la séquence entre les tâches. Dans un premier temps, nous avons exposé une formulation précise du problème, des bornes ainsi que des méthodes exactes. Mal-

heureusement ce type de méthode ne permet pas d'aborder des problèmes de taille raisonnable et nous avons donc étudié les possibilités de résolutions heuristiques. Dans ce type de résolution, l'inconvénient est la qualité dégradée des solutions obtenues. Ainsi, alors que le problème comporte au départ un problème de satisfaisabilité (respect de la période) et un problème d'optimisation (minimisation des coûts), nous avons choisi de l'aborder en relaxant la contrainte de satisfaisabilité et en étudiant un problème bicritère. Nous avons proposé différentes heuristiques et une procédure d'amélioration que nous avons intégrées dans une procédure itérative permettant de traiter le problème initial. Les résultats montrent que cette méthode itérative est un bon moyen pour parcourir différents lieux de l'espace des solutions et ainsi espérer obtenir de bonnes solutions.

PARTIE II

Un problème d'optimisation globale : prise en compte simultanée de la production et de la distribution

Chapitre 4

Gestion d'un système de production multisite

Comme nous avons commencé à le présenter dans l'introduction, la mondialisation des marchés a amené les grandes entreprises multinationales à avoir une vue globale tant de leurs activités liées au marketing que de leurs activités liées à la production au sein des différents sites. C'est ce deuxième aspect qui nous intéressent ici. En effet, ces entreprises sont maintenant confrontées à une gestion globale de leurs différents sites de production et aux difficultés liées à la coordination et à l'optimisation de ces différents sites. Cet aspect de coordination doit être traité avec beaucoup d'attention car les gains apportés par une bonne gestion coordonnée, peuvent être aussi importants que les pertes encourues par une mauvaise gestion.

De fait, les avantages d'une production multisite sont évidents : augmentation de la performance par une possible spécialisation des sites, diminution des coûts relatifs à la distribution par un rapprochement vers le client... Pourtant, d'importantes difficultés interviennent dans la gestion d'une production multisite où il est nécessaire, à la fois de gérer l'ensemble des sites de façon globale pour maximiser les gains, et d'étudier très fortement les liens existants entre l'activité de production et l'activité de distribution. Alors que, beaucoup de recherches ont porté soit sur la gestion des besoins matériels, soit sur l'optimisation d'ateliers de production isolés, comportant des structures de type jobshop, flowshop, machines parallèles ou de type hybride (cf 1.2.1) ou encore sur l'optimisation des problèmes de distribution (tournées de véhicules), peu ont concerné cet aspect de coordination nécessaire à une gestion multisite.

Pourtant, l'optimisation locale d'un seul atelier, par exemple, sans considérer les autres ateliers de l'entreprise, peut être en complète contradiction avec l'optimisation globale du système de production. Il devient donc nécessaire de se poser la question de la coordination entre le système de production et le système de

distribution, ainsi que de la coordination entre les différents sites de production.

Dans cette deuxième partie du document, nous nous intéressons non plus simplement à la partie liée à la production, mais nous considérons la problématique dans son ensemble avec tous les éléments qui la composent dans un objectif d'optimisation globale . Au cours de ce chapitre, nous exposons, dans un premier temps, des travaux récents effectués dans le domaine de la coordination entre la production et la distribution et le domaine des problèmes de production multisite. La deuxième section est consacrée au positionnement de notre problématique vis à vis de l'ensemble des problèmes de production multisite, à sa modélisation et à la détermination du cadre d'étude que nous considérons dans la suite du chapitre. Ensuite, nous présentons une approche centralisée permettant de traiter le problème étudié. La quatrième section présente une approche de résolution basée sur une décomposition spatiale. Nous terminons ce chapitre par une comparaison entre l'approche centralisée et l'approche par décomposition, pour la résolution du problème multisite, multiproduit considéré.

4.1 Littérature

Avant d'exposer notre approche développée pour traiter le problème multisite, avec coordination entre les aspects de production et de distribution, auquel nous sommes confrontés, nous commençons par présenter, dans cette section, des travaux récents, développés pour traiter des problèmes d'optimisation globale. Nous avons divisé ces travaux en deux catégories : les travaux portant sur des aspects de coordination entre production et distribution dans le cas de productions monosites, et les travaux portant sur des problèmes de productions multisites.

4.1.1 Coordination production et distribution

Dans ce paragraphe, nous nous intéressons aux travaux traitant de la coordination entre le système de production et le système de distribution pour le cas de productions monosites où les tailles des lots de production dépendent non seulement du système de production, mais également des aspects de distribution.

L'un des premiers à considérer conjointement la production et la distribution est J.F. Williams [143]. Il s'intéresse à trois types de problèmes : des problèmes d'assemblage dont la gamme est un réseau formant une anti-arborescence (chaque noeud a au plus un successeur), des problèmes de distribution représentés par un réseau formant une arborescence (chaque noeud a au plus un prédécesseur), et, des problèmes couplant les deux structures. L'objectif est la minimisation de la somme de la moyenne des coûts de stockage et de la moyenne des coûts de production, par période, en déterminant la taille des lots de production et de

distribution. Il teste sept heuristiques et discute leur performances.

Plus tard, Burns et al. [21] s'intéressent à un système composé d'un seul fournisseur (stock ou usine) distribuant différents produits à plusieurs clients répartis sur une région géographique. Ils comparent deux stratégies. La première, l'expédition directe, impose que chaque client soit livré directement et isolément. Les formules des coûts de stockage (pouvant être étendues aux coûts de production) et des coûts de transport sont développées pour cette stratégie. Le compromis de ces coûts donne la taille optimale des lots à livrer. La deuxième stratégie, le colportage, autorise un camion à livrer, lors d'une même tournée, plusieurs clients. Des sous-régions de distribution sont créées et les coûts associés au stockage et au transport sont étudiés. Il s'avère que dans ce cas, la taille optimale des lots à livrer revient à remplir les camions. Une comparaison entre ces deux approches montre que la deuxième est la meilleure et que son avantage, en terme de coûts, augmente avec la distance entre le fournisseur et les clients, la densité des clients, la valeur des produits et les coûts de transport.

Chandra et Fisher [22], quant à eux, considèrent le cas d'une usine produisant différents types de produits et maintenant un stock de produits finis dans l'usine. La demande est connue par période pour l'ensemble des clients répartis sur le territoire et la distribution se fait à l'aide d'une flotte de véhicules. L'objectif est de déterminer l'ordonnancement de la production de façon à minimiser les coûts de changement, de stockage et de transport, sachant qu'un client peut être livré en avance, mais pas en retard, et que les coûts de transport sont des coûts fixes par véhicules (qu'il est donc préférable de remplir). Deux approches sont comparées. La première traite la production et la distribution séparément, tandis que la deuxième intègre les deux aspects dans un même modèle. Pour résoudre ce problème de coordination, ils utilisent les mêmes méthodes que pour l'étude séparée, puis recherchent des changements amenant à des réductions de coûts. Ainsi, l'approche qu'ils qualifient de coordonnée, permet une réduction de l'ensemble des coûts de 5% à 10% sur les exemples traités.

Un système de production composé d'une usine, d'un stock de produits finis et d'un unique client est étudié par Pyke et Cohen [117]. Différents produits auxquels est associée une demande stochastique, sont fabriqués dans l'usine, stockés puis distribués. De plus, les stocks peuvent passer commandes des produits dont le niveau de stockage diminue dangereusement. Ils proposent une approximation du modèle complexe et une heuristique de résolution basée sur cette approximation.

Récemment, Barbarasoglu et Özgur [9] considèrent le problème de la détermination des tailles des lots pour différents produits fabriqués dans une même usine. Ces produits sont distribués de l'usine à un ensemble de dépôts et de ces dépôts

aux clients. La demande des clients est déterministe, connue pour chaque période, et la livraison doit se faire en juste-à-temps. Ils adoptent une approche basée sur une relaxation Lagrangienne pour découpler les décisions liées à la production et les décisions liées à la distribution. Puis une optimisation est implémentée pour coordonner les deux sous-problèmes de façon hiérarchique.

4.1.2 Production multisite

Dans ce paragraphe, nous nous intéressons aux problèmes de production multisite. Les travaux cités ci-dessous concernent des problématiques différentes ayant toutes comme point commun de considérer, plusieurs sites de production répartis sur un territoire.

Tout d'abord, Cohen et Lee [31] s'intéressent à toute une chaîne de production depuis les fournisseurs de matières premières jusqu'au client, en passant par les usines produisant les produits semi-finis, les usines des produits finis, les centres de distribution et les centres de stockage. Ils présentent un modèle pour coordonner les décisions et les performances tout au long de cette chaîne soumise à des demandes stockastiques. Le modèle montre les interactions existantes dans les systèmes de production et de distribution multiéchelon. A chaque étage des sous-modèles sont définis et des règles de contrôle utilisées. Une procédure d'optimisation est proposée et les résultats sont discutés à partir d'un exemple.

Garavelli et al. [53], posent le problème de la coordination des différents sites de production d'une multinationale. Ils présentent les avantages liés à la prise en compte globale, au niveau de la planification, de l'ensemble du système manufacturier. Pour le cas monoproduit, ils présentent un modèle de planification, développé dans un objectif de minimisation globale de l'ensemble des coûts (production et distribution) et utilisent un algorithme génétique pour la recherche de solutions. Ils présentent une petite application (trois usines, deux marchés et cinq niveaux de quantités par marché) qu'ils résolvent à l'optimum pour 80 % des cas en 80 secondes, en moyenne. Ils concluent sur l'extension possible de leur approche à des environnements plus complexes.

Metters [101], expose le problème de l'interaction entre le transport et le tri (équivalent à une phase de production) du courrier à l'United States Postal Service. Le mouvement journalier et le tri des lettres nécessite un équilibrage entre les coûts de transport et les coûts liés au tri. Ce problème peut être assimilé à un problème multisite, monoproduit et multipériode. Il traite le problème d'une division spécifique et le modélise en un programme linéaire mixte qu'il peut réduire en considérant la configuration propre du système étudié. Il peut ainsi utiliser un solveur de programmes linéaires pour résoudre son problème.

Le projet DISCO

A Toulouse, le projet ESPRIT DISCO (Distributed management and Coordination of Scheduling Systems in a Multisite production Environment), réalisé par un consortium composé de MAGNETI-MARELLI, AEG, BULL, FhG-IAO et PROMIP (Productique Midi-Pyrénées) a étudié la coordination et l'ordonnancement de productions multisites. Dans ce cadre, la thèse de C. Boronad-Thierry [17], a porté sur la planification et l'ordonnancement multisite en adoptant une approche par satisfaction de contraintes. Thierry et al. [134] ont alors proposé un modèle basé sur la théorie des flots pour des problèmes multisites, multiproduits et multipériodes. Ils proposent de reformuler ce problème en un problème de satisfaction de contraintes pour pouvoir le résoudre. Dans [135], Thierry et al. portent l'accent sur un début de réduction de complexité du problème, afin de pouvoir utiliser une gestion centralisée de l'ensemble de ce système multisite. Puis, Bel et al. [10] insistent sur l'importance et la difficulté de la coordination d'un système de production multisite, en vue d'améliorer sa productivité. Une modélisation de ce problème, considérant simultanément plusieurs niveaux de décision (correspondant aux niveaux d'agrégation des ressources), est présentée. Ils proposent une méthode de réduction basée sur la notion de produits critiques et de processus critiques. Ils définissent ainsi un ensemble de criticalités (technologique, managériale, économique, structurale) qui leur permettent de réduire la taille du problème. Leur heuristique de résolution est divisée en trois parties : la recherche rapide d'une bonne première solution, des modifications locales visant à réduire le coût global, la vérification de la satisfaction des contraintes. La première étape, la plus délicate, considère le problème multisite comme un problème de satisfaction de contraintes. La méthode utilisée, et détaillée dans [11], représente l'ensemble des variables de décisions dans un cube de décision, et plusieurs stratégies sont alors testées.

L'outil d'aide à la décision ARIANE

Un autre pôle de recherche sur la production multisite s'est créé depuis plus d'une vingtaine d'années, autour de l'outil d'aide à la décision ARIANE, développé pour la conception des plans de production de bouteilles de verre chez BSN/Emballage. La problématique industrielle concerne la planification de la production de bouteilles de verre (environ 2 000 articles emballés différemment) sur les différents sites de production (7 en France) et sur plusieurs périodes pour faire face à une demande saisonnière. L'outil, de production est formé de lignes de production polyvalentes, reparties sur les différents sites, et composées d'un distributeur (four) qui alimente plusieurs machines de façonnage en parallèle avec une matière première d'une certaine qualité (teinte).
Depuis la première version du logiciel, version 1977, (voir la Thèse de C. Proust

[115]), qui souffrait d'un manque d'interactivité et de convivialité causé par les moyens informatiques de l'époque, cet outil n'a cessé d'évoluer grâce aux collaborations entre BSN et l'université de Tours. Dans [116], Proust et Grünenberger présentent une nouvelle version d'Ariane, de meilleure convivialité, retransplantée sur micro-ordinateur. Ils détaillent également les caractéristiques du problème posé et se placent aux niveaux long terme et moyen terme de planification. Ils considèrent qu'à ce niveau de planification, les changements de fabrication sur les machines peuvent être ignorés car ils se traduisent par une diminution temporaire des cadences. Ils modélisent le problème en un programme mathématique qui n'autorise qu'une seule qualité par distributeur et par période. Ils constatent alors l'impossibilité technique (étant donnés la nonlinéarité du modèle, le nombre élevé de variables et de contraintes) de résoudre directement le problème, et montrent l'intérêt d'un logiciel interactif permettant à un décideur de participer à la recherche des solutions : ceci constitue l'esprit d'Ariane 2 000.

Le logiciel fonctionne en deux étapes [26]. Premièrement, la répartition des teintes sur les fours est optimisée en agrégeant les données relatives, aux demandes commerciales, aux teintes, aux régions géographiques et aux périodes, et en se rapportant à un Problème de Transport Simple. On obtient alors un ou plusieurs calendriers des teintes sur les fours. Deuxièmement, les produits sont affectés, teintes par teintes, sur les machines en résolvant, à chaque fois, un problème de type Transport Généralisé. Les coûts de transport sont considérés dans cette phase au même titre que les coûts de production. Dernièrement, la thèse de P. Richard [120], réalisée dans le cadre d'une convention CIFRE chez BSN, a porté en partie sur le développement de la version Europe d'Ariane 2 000.

Ariane a été développé essentiellement pour traiter les problèmes de planification de la production des bouteilles de verre à long et moyen terme. De récents travaux ont également porté sur la planification court terme [122], [121]. Celle-ci consiste, une fois le plan moyen terme établi, à affecter les modèles des produits aux machines. Le problème résolu ici est noté $R/.../blcg/\bar{M}$. Ce problème d'affectation sur machines parallèles non liées est traité suivant trois modes : avec splitting, avec préemption et sans préemption.

4.1.3 Tableau récapitulatif de la littérature

Nous représentons dans le tableau 4.1, un récapitulatif des différents problèmes traités dans la littérature. Pour mieux positionner notre problématique vis-à-vis des ces travaux, nous les avons classés en fonction de différentes caractéristiques : monosite/multisite, monoproduit/multiproduit, monopériode/multipériode.

	Monopériode	Multipériode
Monosite	Multiproduit J.F. Williams [143] Burns [21]	Multiproduit Chandra et Fisher [22] Pyke et Cohen [117] Barbarasoglu et Özgur [9]
Multisite	Monoproduit Gavarelli [53] Multiproduit Cohen et Lee [31]	Monoproduit Metters [101] Multiproduit DISCO [17] [134] [135] [10] ARIANE [115] [116] [26] [120] [122] [121]

TAB. 4.1 - *Tableau récapitulatif de la littérature*

4.2 Positionnement de la problématique

Comme nous l'avons indiqué dans la présentation générale de la problématique (au chapitre 0), toute une zone géographique est à considérer en tenant compte de la localisation des usines, des stocks et des clients ainsi que des distances séparant ces différents éléments. Nous avons donc à faire à un problème de production multisite. Nous allons, dans cette partie, caractériser de façon générale les problèmes de production multisite en positionnant, dans ce cadre, notre propre problématique. Puis nous proposerons une modélisation du problème global, intégrant l'ensemble des éléments et définirons de manière précise les éléments pris en compte dans l'approche que nous proposons.

4.2.1 Les problèmes de production multisite

Les problèmes de production multisite se posent dans le cadre d'entreprises qui comprennent plusieurs sites de production distincts géographiquement et qui ont la possibilité de fabriquer certains produits ou composants dans plusieurs sites. Il peut donc exister des transports de marchandises, sur de grandes distances, entre les sites de production eux-mêmes, ainsi qu'entre les sites de production et les clients [17]. Cette répartition des moyens de production sur tout un territoire, a bien des avantages au niveau des coûts : réduction des coûts de production en spécialisant les sites de production en une sous famille de produits, réduction des coûts de transport en se rapprochant des clients.

Bel et al. [10] distinguent deux types de situations : les *productions alternatives*, lorsqu'une entreprise possède plusieurs sites de production capables de réaliser les mêmes produits finis, et les *productions séquentielles*, lorsque certains sites de production produisent les composants utilisés ensuite par d'autres sites de l'en-

treprise. Notre problématique se situe dans le cadre des productions alternatives, où un produit est entièrement manufacturé sur un seul site, et plus précisément sur une seule ligne de production, qui est sélectionnée parmi l'ensemble des lignes de production du système.

Nous rappelons que les trois particularités principales de notre problématique initiale sont a) des lignes de production de caractéristiques (cadences, coûts...) différentes, b) des temps de changement entre produits pouvant être très pénalisants en terme de temps et coûts et c) des coûts de distribution élevés (du même ordre que les coûts de fabrication) pour des produits de faible valeur ajoutée. Ces trois éléments imposent d'analyser le problème dans son ensemble et en particulier le lien existant entre les aspects production et les aspects distribution, puisque les composantes production et distribution, ont ici toutes deux un poids très important en terme de coûts. Or les décisions prises de façon à diminuer les coûts associés à la production peuvent être désastreuses par rapport à l'optimisation de la distribution et vice versa. En effet, du choix de l'affectation d'un produit à une ligne va dépendre la distribution, et en particulier, les coûts de distribution associés à la livraison de ce produit au client. Pourtant, dans un système multisite, la gestion de production est très souvent propre aux différents sites, sans coordination, à cause de la difficulté de leur gestion coordonnée.

Or, d'après Giard, "la gestion de production a pour objet la recherche d'une organisation efficace de la production de biens et de services" [56]. Plus précisément, d'après Dupont, "la gestion de production consiste à produire en temps voulu, les quantités demandées par les clients dans des conditions de coût de revient et de quantité déterminés, en optimisant les ressources de l'entreprise de façon à assurer sa pérénité, son développement et sa compétitivité" [40].

Ainsi, dans cet objectif d'optimisation des ressources et de recherche de compétitivité, il est nécessaire, en environnement multisite, d'avoir un système de gestion de production intégrant l'ensemble des sites. Le système de gestion de production doit prendre des décisions concernant :

- le long terme : décisions prises au plus haut niveau de l'entreprise (politiques commerciales, objectifs généraux, investissement...),
- le moyen terme (planification) : décisions permettant de fournir les ressources physiques et informationnelles nécessaires à la production (ressources en hommes, équipements, matières premières...),
- le court terme (ordonnancement) : décisions déterminant la séquence des produits à réaliser et leur date de lancement.

Très souvent, l'aspect multisite n'est considéré que dans les décisions à long terme et éventuellement à moyen terme sur des données agrégées, à cause de la taille des

problèmes à résoudre et du nombre de données à traiter lorsque l'on s'intéresse au court terme et donc à des données détaillées.

Dans notre problématique, un compromis est à trouver entre 1) l'affectation des produits sur des lignes de bons rendements pour eux de façon à limiter les coûts de production, 2) le regroupement de produits semblables sur des mêmes lignes pour minimiser les temps et coûts associés aux changements et 3) la fabrication des produits dans les usines proches des clients les ayant commandés de manière à limiter les coûts de distribution. Si l'affectation a été décidée à un niveau de décision moyen terme, par exemple, en se basant exclusivement sur les coûts de production et de distribution (sans tenir compte des changements de production), il peut être très difficile pour le court terme, non seulement d'optimiser les temps et coûts relatifs à ces changements, mais également de trouver un ordonnancement réalisable. Nous pensons donc que ces trois éléments doivent être considérés simultanément.

Ce souci de coordination entre les aspects planification et ordonnancement, permettant de s'assurer de la réalisabilité du plan de production donné par la planification est considéré par C. Boronad-Thierry dans sa thèse [17]. Elle propose d'adapter, aux problèmes multisites, multiproduits (mais sans temps de changement dépendant de la séquence) et multipériodes, l'approche cohérente intégrée planification-ordonnancement, présentée par Lasserre [89] et Dauzère-Peres [34] pour des problèmes de type jobshop. Cette approche consiste à résoudre alternativement les problèmes de planification et d'ordonnancement, pour assurer leur cohérence. Pour les problèmes multisites, C. Boronad-Thierry utilise les langages de programmation sous contraintes et en particulier les mécanismes de démons, pour déclencher la recherche d'ordonnancement dès qu'une décision est prise au niveau de la planification. Ainsi, chaque fois qu'une quantité de produit est planifiée pour une période sur une unité de production, elle est affectée à une ressource et la réalisabilité de l'ordonnancement est vérifiée. En fonction du résultat de cette vérification, certaines décisions seront remises en cause, de façon à obtenir un ordonnancement réalisable.

C. Boronad-Thierry [17] indique qu'il n'est pas possible d'envisager une approche intégrée pour l'ensemble du problème, en particulier pour des raisons de complexité, mais également pour des raisons de validité des données pour les périodes futures. Notre approche est différente. Nous recherchons, au contraire, à considérer dès le plus haut niveau les aspects de séquencement (à cause des changements très pénalisants entre produits). La méthode de résolution proposée étant rapide, cela permet au décideur de la relancer très souvent, et en particulier lorsque les données changent. Ainsi, la question de validité des données n'est pas pour nous, un élément génant pour la recherche de solution.

Pour cela nous proposons une approche non classique qui consiste en une prise de décision court terme (au niveau de l'ordonnancement) pour l'ensemble des lignes de production des différents sites.

4.2.2 Modélisation du problème global

Dans cette partie, nous exposons une formulation mathématique de la problématique qui permet une meilleure compréhension du problème et de ses contraintes. La taille de ce programme linéaire mixte augmente très rapidement lorsque l'on considère plusieurs produits et plusieurs périodes. De plus, certaines variables de décision sont bivalentes ce qui rend l'obtention de solutions optimales ou même simplement réalisables difficile. C'est pourquoi, ce programme linéaire ne pourra être exploité tel quel pour la résolution du problème, mais permet néanmoins d'en préciser les contraintes et de comprendre l'influence des aspects de production sur les aspects de distribution et vice-versa.

4.2.2.1 Les données du problème

Les données du problème concernent la production, le stockage et les clients. Les indices i et j sont relatifs aux produits, l'indice m aux machines (ou lignes de production), l'indice s aux stocks, l'indice c aux clients et l'exposant t aux périodes.

Production

Tp_{im}, Cp_{im} : Temps et coût de production de i sur m (en heures par millier).
Tc_{ijm}, Cc_{ijm} : Temps et coût de changement lors du passage de i à j sur m.
B_m^t : Capacité du processeur m pour la période t (en heures).

Stockage

Ct_{ms} : Coût de transport de m à s (par unité de volume).
K_s : Capacité de stockage du stock s (en volume).
V_i : Volume du produit i (pour un millier de produit).
Cs_{is} : Coût de stockage de i dans le stock s (par millier). Détails :
Ce_{is} : Coût d'entrée en stock.
Cd_{is} : Coût de sortie.
Ci_{is} : Coût d'immobilisation.

Clients

Nous considérons que les clients passent leurs commandes pour une période donnée, sans précision supplémentaire sur la date de livraison.

D_{ic}^t : Demande en produit i du client c pour la période t (par millier).
Cl_{sc} : Coût de livraison de s à c (par unité de volume).
Ca_{mc} : Coût d'acheminement de m à c (par unité de volume).

4.2.2.2 Les variables de décision

Les variables nécessaires à la modélisation du problème concernent les décisions à prendre au niveau de la production, du stockage et du transport. Les variables concernant l'affectation et l'ordonnancement des productions sur les lignes sont de même type que les variables utilisées dans le chapitre 3 pour le problème d'ordonnancement sur machines parallèles. La différence est qu'il s'agit d'affecter la production d'un type de produit dont la quantité reste à déterminer par d'autres variables, tandis que pour le problème d'ordonnancement sur machines parallèles, il s'agissait d'affecter l'exécution complète d'une tâche (tout ou rien).

Production

q_{im}^t : Quantité de i fabriquée sur m pendant t (en millier).
y_{ilm}^t : $= 1$ si i est fabriqué en position l sur m pendant t,
: $= 0$ sinon.
x_{ijm}^t : $= 1$ si j suit immédiatement i sur m pendant t,
: $= 0$ sinon.

Stockage

t_{ims}^t : Quantité de i transportée depuis m jusqu'à s pendant t (en millier).
s_{is}^t : Quantité de i dans le stock s à la fin de t (en millier).

Livraison

a_{imc}^t : Quantité de i acheminée directement de m à c pendant t (en millier).
l_{imc}^t : Quantité de i livrée depuis s jusqu'à c pendant t (en millier).

4.2.2.3 Les contraintes

De même les contraintes permettant de modéliser le problème se situent aux différents niveaux de la problématique. Au niveau de la production, il faut vérifier le respect des capacités de production, le bon enchaînement des produits et la prise en compte des temps de changement entre produits différents. Au niveau du stockage, les contraintes concernent le respect des capacités de stockage, et l'équilibrage entre les entrées et sorties. Finalement, au niveau des clients, les contraintes assurent chaque client d'obtenir sa demande. Ces contraintes sont exposées ci-dessous.

Production

Tout d'abord, les inéquations (4.1) permettent de s'assurer que pour chacune des lignes, le temps passé à la production et aux changements, pendant une période, ne dépasse pas la capacité horaire de la ligne pour cette période.

$$\sum_i q_{im}^t \times Tp_{im} + \sum_i \sum_j x_{ijm}^t \times Tc_{ijm}^t \leq B_m^t \qquad \forall m, \forall t \qquad (4.1)$$

Les contraintes suivantes (4.2 - 4.6) permettent de définir l'enchaînement des produits et de comptabiliser les temps de changement. Ainsi, les inégalités (4.2) affectent à chaque produit, dont la quantité fabriquée sur une ligne pendant une période donnée est non nulle, une place dans l'ordonnancement ($\mathcal{HV}$ représente une grande valeur). L'inéquation suivante (4.3) s'assure qu'une place de l'ordonnancement n'est utilisée que par la fabrication d'un seul produit, tandis que l'inéquation (4.4) vérifie qu'un produit n'est fabriqué qu'une seule fois sur une machine donnée pendant une période (cohérence avec la minimisation des temps et coûts de changements). Ensuite, l'inéquation (4.5) assure la continuité de l'ordonnancement en interdisant à une production d'occuper la place $l+1$ si aucune production n'occupe la place l. Enfin, les contraintes (4.6) positionnent les variables de succession x_{ijm}^t à 1 lorsque les produits i et j se succèdent sur la machine m pendant la période t, ce qui permet de considérer les temps et coûts de changement entre produits successifs.

$$q_{im}^t - \mathcal{HV} \times \sum_l y_{ilm}^t \leq 0 \qquad \forall i, \forall m, \forall t \qquad (4.2)$$

$$\sum_i y_{ilm}^t \leq 1 \qquad \forall l, \forall m, \forall t \qquad (4.3)$$

$$\sum_l y_{ilm}^t \leq 1 \qquad \forall i, \forall m, \forall t \qquad (4.4)$$

$$\sum_i y_{i(l+1)m}^t - \sum_i y_{ilm}^t \leq 0 \qquad \forall l, \forall m, \forall t \qquad (4.5)$$

$$y_{j(l+1)m}^t + y_{ilm}^t - x_{ijm}^t \leq 1 \qquad \forall i, \forall j, \forall l, \forall m, \forall t \qquad (4.6)$$

De même que pour l'enchaînement des produits au sein d'une période, l'enchaînement des produits entre périodes doit être considéré. Nous prenons la convention de toujours terminer une période avec une production (ou éventuellement un temps mort) et de commencer la période suivante avec un changement de production qui peut être nul si la fabrication d'un même produit termine une période et commence la suivante. Ainsi, nous interdisons aux changements de production d'intervenir au moment des changements de période (cas très difficiles à modéliser). Dans ce contexte, les contraintes (4.7) positionnent les indicateurs de

changement (variables x_{ijm}^{t+1}), entre le dernier produit i de la période t et le premier produit j de la période $t+1$, à 1. Le dernier produit de la période t est celui qui n'a pas de successeur c'est-à-dire celui pour lequel, la somme $\sum_j \sum_{l'>l} y_{jl'm}^t$ est nulle.

$$y_{j1m}^{t+1} + y_{ilm}^t - \sum_j \sum_{l'>l} y_{jl'm}^t - x_{ijm}^{t+1} \leq 1 \qquad \forall i, \forall j, \forall l, \forall m, \forall t \tag{4.7}$$

Stockage

Un niveau du stockage, un premier type de contraintes est nécessaire pour permettre la conservation du flot de produits passant par les stocks. Ainsi, l'équation (4.8) indique que la quantité de produit i stockée en s à la fin de la période t est égale à la quantité transportée jusqu'à ce stock pendant la période t plus la quantité déjà présente en fin de période $t-1$ moins la quantité livrée aux clients depuis le stock pendant t. Les contraintes (4.9), quant à elles, assurent le respect des capacités de stockage.

$$\sum_m t_{ims}^t + s_{is}^{t-1} - \sum_c l_{isc}^t - s_{is}^t = 0 \qquad \forall i, \forall s, \forall t \tag{4.8}$$

$$\sum_i s_{is}^t \times V_i \leq K_s \qquad \forall s, \forall t \tag{4.9}$$

Livraison

Les deux ensembles de contraintes qui suivent, gèrent le transport des marchandises. Tout d'abord, les contraintes (4.10) s'assurent que chaque client reçoit bien la quantité de produit demandée, au bon moment. Puis les contraintes (4.11) expriment que la quantité de produit i acheminée directement aux clients depuis une ligne de production plus la quantité transportée jusqu'aux stocks depuis cette ligne est égale à la quantité produite sur la ligne.

$$\sum_m a_{imc}^t + \sum_s l_{isc}^t = D_{ic}^t \qquad \forall c, \forall i, \forall t \tag{4.10}$$

$$q_{im}^t - \sum_c a_{imc}^t - \sum_s t_{ims}^t = 0 \qquad \forall i, \forall m, \forall t \tag{4.11}$$

Intégralité

Les contraintes suivantes déterminent le domaine de validité de chacune des variables.

$$y_{ilm}^t, x_{ijm}^t \in \{0,1\} \qquad \forall i, \forall j, \forall l, \forall m, \forall t \tag{4.12}$$

$$q_{im}^t, t_{ims}^t, s_{is}^t, a_{imc}^t, l_{isc}^t \geq 0 \qquad \forall i, \forall m, \forall s, \forall c, \forall t \tag{4.13}$$

En fait, il s'avère que les contraintes d'intégralité sont redondantes et que seule l'intégralité sur les variables y^t_{ilm} est nécessaire. En effet, les contraintes (4.6) et (4.7) suffisent, dans le cas d'une minimisation, pour positionner les variables x^t_{ijm} à 0 ou à 1, sans qu'il soit nécessaire de le préciser. Ainsi, les domaines de validité des variables peuvent être décrits par :

$$y^t_{ilm} \in \{0,1\} \qquad \forall i, \forall l, \forall m, \forall t \tag{4.12'}$$

$$x^t_{ijm}, q^t_{im}, t^t_{ims}, s^t_{is}, a^t_{imc}, l^t_{isc} \geq 0 \qquad \forall i, \forall j, \forall m, \forall s, \forall c, \forall t \tag{4.13'}$$

4.2.2.4 La fonction objectif

L'objectif de l'étude de ce problème est la minimisation de l'ensemble des coûts (production, changement, transport, stockage), ce qui peut être modélisé par la fonction suivante.

$$\sum_t \sum_i [\sum_m (q^t_{im} \times Cp_{im} + \sum_j x^t_{ijm} \times Cc_{ijm} + \sum_s t^t_{ims} \times V_i \times Ct_{ms} + \sum_c a^t_{imc} \times V_i \times Ca_{mc}) + \sum_s (\sum_c l^t_{isc} \times V_i \times Cl_{sc} + s^t_{is} \times Cs_{is})] \tag{4.14}$$

Le premier terme correspond aux coûts de production, le deuxième aux coûts relatifs aux changements. Les trois terme suivants correspondent aux différents types de transport : transports lignes-stocks, acheminements lignes-clients et livraisons stocks-clients. Enfin le dernier terme est relatif aux frais de stockage, considérés de façon globale.

4.2.3 Description du problème étudié

Ayant présenté précisément le problème tel qu'il a été exposé initialement, nous proposons maintenant de définir quels sont les éléments considérés ici. En fait, dans la suite du chapitre, nous allons étudier la problématique avec ses aspects :

- multisite (plusieurs sites répartis sur un territoire),
- multiproduit (avec des lignes de production de comportement différents vis à vis des produits et des temps de changements dépendants de la séquence entre les produits),
- monopériode (les aspects de stockage n'interviennent pas).

Comme nous l'avons exposé plus haut, en vue de minimiser l'ensemble des coûts, un compromis est à trouver entre trois décisions possibles et souvent contradictoires :

- affecter les produits sur leur(s) meilleure(s) ligne(s) pour minimiser les coûts de production,

- regrouper les produits semblables sur des mêmes lignes pour diminuer les coûts de changement,
- fabriquer les produits dans les usines proches des clients pour diminuer les coûts de distribution.

Ces trois éléments sont très importants et aucun ne peut être négligé. En particulier, il est nécessaire, à tout niveau, de considérer les temps de changement car ceux-ci peuvent durer jusqu'à six jours et diminuent fortement les capacités de production. Ainsi, notre étude porte, non seulement sur la recherche d'une affectation des produits aux lignes mais, également sur la recherche des séquencements des produits sur les lignes qui permettent d'évaluer les temps et coûts associés aux changements et donc de contrôler le respect des capacités de production des lignes. Nous nous intéressons alors à la recherche, pour une période donnée, de l'affectation et de l'ordonnancement d'un ensemble de produits sur un ensemble de lignes de production, de caractéristiques différentes, réparties sur une zone géographique, de façon à minimiser les coûts de production, de changement et de distribution aux clients. Le stockage n'étant pas considéré dans ce problème monopériode.

Pour traiter ce problème très général, nous modifions, comme dans la première partie du document, la notion de produit. Dans la problématique, les clients commandent une certaine quantité de produits. Nous appelons une tâche, la fabrication d'une demande. De cette façon, la notion de destination d'un produit est intégrée dans la caractérisation de la tâche à réaliser et les coûts de distribution ne dépendent alors que de la tâche et de la ligne de fabrication. Si une demande est trop importante (quantité trop importante pour être produite par une seule machine), elle est découpée en plusieurs tâches.

Ainsi, les différents éléments du système de production à considérer sont :

$F = \{1 \dots |F|\}$ ensemble des fabriques (usines).
$M = \{1 \dots |M|\}$ ensemble des lignes de production (non liées).
$J = \{1 \dots |J|\}$ ensemble des tâches à réaliser.

Les données sont :

Tp_{im} : temps de production de la tâche i sur la ligne m.
Cp_{im} : coût de production de la tâche i sur la ligne m.
Tc_{ijm} : temps de changement entre i et j sur la ligne m.
Cc_{ijm} : coût de changement entre i et j sur la ligne m.
B_m : capacité de la ligne m.
Ct_{im} : coût de distribution de la tâche i depuis la ligne m (rappelons que la notion de destination est incluse dans la notion de tâche).

4.3 Approche centralisée

La façon optimale de considérer ce problème, serait de centraliser toute les demandes en un même centre de décision qui procéderait, en ayant une vue globale de l'ensemble du système de production, à l'ordonnancement des productions sur l'ensemble des lignes de la division industrielle.

Lorsque l'on considère une gestion centralisée de ce système, le problème consiste en un problème d'ordonnancement sur machines parallèles non liées, dispersées sur une zone géographique, avec temps de changement entre produits dépendant de la séquence. Il peut donc être modélisé à l'aide du programme linéaire **PL1** présenté section 3.2, lors de l'étude du problème d'ordonnancement exposé dans la première partie du document, avec les paramètres adéquats :

$$\mathbf{PL1}(J,\, M,\, Tp_{im},\, (Cp_{im} + Ct_{im}),\, Tc_{ijm},\, Cc_{ijm},\, B_m)$$

Comme nous l'avons vu au chapitre 3, ce problème est difficile à résoudre et les méthodes exactes ne sont utilisables que pour des problèmes de petites tailles. Dans le cas d'une gestion centralisée, le problème est de grande taille et seule des heuristiques, comme celles présentées au chapitre 3, permettent de trouver des solutions.

Pour cela, dans la partie suivante, nous proposons une décomposition spatiale profitant de la répartition géographique des centres de production pour diviser le problème global en sous-problèmes de tailles plus petites de façon à utiliser de meilleures méthodes de résolution [47].

4.4 Décomposition spatiale hiérarchique

Lors de l'étude de problèmes complexes faisant intervenir de nombreux paramètres, une méthode couramment utilisée est la décomposition du problème global en sous-problèmes de tailles plus petites. Afin de tirer parti, au mieux, de l'aspect multisite du problème, nous proposons d'adopter une décomposition spatiale [49].

M.C. Portmann indique qu'au niveau de l'atelier, "la décomposition spatiale cherche à constituer des sous-ateliers les plus indépendants possibles les uns des autres" [113]. Deux types de découpage sont mis en évidence : le découpage par fonction qui regroupe les machines qui correspondent à une même phase de fabrication et le découpage par famille de produits, où une famille de produits est (presque) entièrement fabriquée au sein d'un groupe. Une méthode de décomposition par famille de produits, inspirée d'une méthode de résolution des problèmes de technologies de groupe, est proposée pour la résolution de problèmes d'ordonnancement.

Pour une meilleure efficacité des méthodes de décomposition, l'accent est porté sur la nécessité d'obtenir des groupes indépendants, donnant des problèmes pouvant alors être résolus séparément et plus facilement que le problème global [113]. C'est dans cette optique que nous proposons une décomposition basée sur des considérations spatiales. Nous rappelons ici que la particularité de notre étude est de se placer au niveau court terme pour la recherche de l'ordonnancement des lignes de production des différents sites. Ainsi, la décomposition spatiale proposée divise le problème global en plusieurs sous-problèmes, liés par une structure hiérarchique à plusieurs niveaux, chaque niveau ayant le même horizon de planification.

Pour décrire la décomposition spatiale et la structure hiérarchique nécessaire à la coordination des sous-problèmes, nous présentons, tout d'abord, le schéma global de la structure. Puis nous détaillons les différents éléments de cette structure avec leurs données et résultats. Enfin, nous terminons par décrire les échanges de données nécessaires entre ces éléments.

4.4.1 Schéma global

Le système est décomposé en quatre niveaux (voir figure 4.1). Le premier niveau (composé de la division industrielle toute entière) est le plus général. Le niveau quatre (composé des lignes de production) est le plus précis.

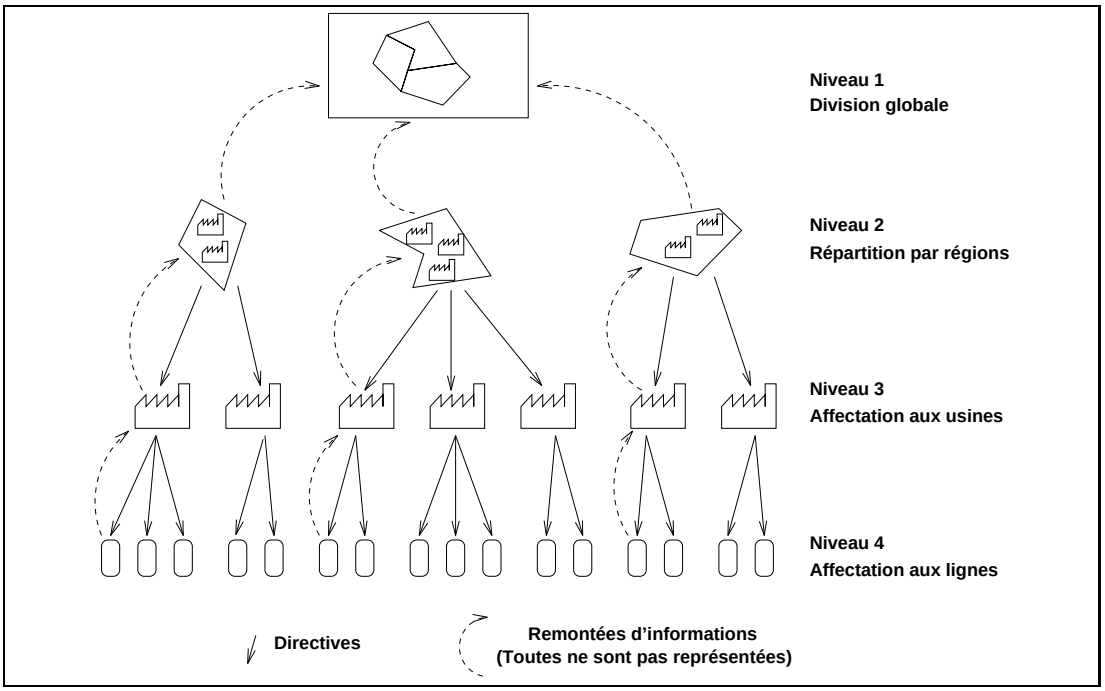


FIG. 4.1 - *Décomposition spatiale*

Étant donné l'importance des coûts de distribution (du même ordre que les coûts de fabrication), la localisation des clients par rapport aux usines est un facteur important. C'est pourquoi, nous proposons une décomposition spatiale qui regroupe les usines en régions géographiques (comme dans Proust et Grünenberger [116]). Ces régions sont construites en fonction de la localisation des clients et des caractéristiques des lignes de production des usines. Les clients sont ensuite affectés, en fonction de leur position géographique essentiellement, à une région et adressent leurs demandes à cette région qui doit ensuite les répartir sur l'ensemble de ses usines.

4.4.2 Niveau 1 : Le niveau central - C

Le rôle du niveau central est de répartir les usines en régions en fonction de la dispersion des clients et du type de leur demande. Pour cela il se base sur les données concernant plusieurs années de fonctionnement afin de pouvoir équilibrer les demandes et les capacités de production. Il prend des décisions à long terme qui peuvent de temps en temps modifier la composition en régions en déplaçant une usine d'une région à une autre.

Également, le niveau central prend les décisions concernant l'anticipation des périodes de fortes demandes et peut imposer à certaines régions de produire des demandes supplémentaires de façon à remplir les stocks et pouvoir répondre à la demande des clients en périodes de sous-capacités de production. En effet, les demandes en boîtes de boissons varient d'une période à l'autre en fonction de la saison. Au printemps, les demandes augmentent très fortement pour diminuer dès la fin de l'été. Il est alors nécessaire de prévoir ces variations et de produire en avance. Ainsi, pendant les périodes de faible demande, une demande visant à remplir les stocks (D_{is}) est ajoutée à la demande des clients, alors que pendant les périodes de fortes demandes, des produits issus des stocks sont utilisés pour satisfaire les clients (correspond alors à des D_{is} négatifs). A ce niveau, les données et résultats sont les suivants :

Données :
B_r : Capacité de production de la région r.
D_{il} : Demande des clients (pour chaque période).

Résultats :
Décision de planification à long terme :
- première répartition des usines en régions,
- vérification de l'équilibrage entre les régions,
- anticipation des périodes de fortes demandes, prévisions de stockage (D_{is}).

Les décisions à prendre à ce niveau relèvent de compétences très spécialisées et également d'études poussées sur les habitudes de commande des clients, sur

les capacités de production des lignes, etc. Sans données réelles concernant ces aspects, il n'est pas possible de proposer une quelconque répartition. Ainsi, dans notre étude, nous prendrons comme données de départ, les décisions prises au niveau central, c'est-à-dire une répartition des usines en régions et un ensemble de clients affectés à chaque région.

4.4.3 Niveau 2 : Les régions - R_r

A ce niveau, les usines sont réparties en régions. Un client passe sa commande directement à une région qui doit la satisfaire. Ainsi, étant donné un ensemble de demandes, une région doit affecter chacune de ces demandes à l'une de ses usines, en tenant compte de la production, des changements et de la distribution, de façon à minimiser l'ensemble des coûts. Des données agrégées (capacités de production, temps de production, coûts...) sont utilisées pour caractériser chacune des usines, permettant ainsi de diminuer le nombre d'informations à traiter à ce niveau. "Le principe d'agrégation peut se définir comme une forme d'abstraction par laquelle un ensemble de données est remplacé par un ensemble moins important de données tout en conservant une partie significative de l'information contenue dans l'ensemble initial" [40]. Nous définirons plus loin, lors de la présentation de l'expérimentation, le type d'agrégation utilisé.
Les demandes des clients (D_{il}) et les prévisions de stockage (D_{is}), déterminées par le niveau central, sont transformées en tâches auxquelles nous associons des temps et des coûts. Les données et les résultats pour chaque région sont donnés ci-dessous :

Données :
J_r : Ensemble de tâches à réaliser par la région r (D_{il} et D_{is}).
F_r : Ensemble de fabriques (usines) de la région r $(1 \ldots |F_r|)$.
Tp_{if} : Temps de production de la tâche i dans l'usine f.
Cp_{if} : Coût de production de la tâche i dans l'usine f.
Ct_{if} : Coût de transport de la tâche i depuis l'usine f.
Tc_{ijf} : Temps de changement de la tâche i à la tâche j dans l'usine f.
Cc_{ijf} : Coût de changement de la tâche i à la tâche j dans l'usine f.
B_f : Capacité de production de l'usine f.

Résultats :
J_f : Affectation de la demande aux usines : détermination de l'ensemble des tâches à réaliser par l'usine f.

Le problème, à ce niveau, consiste en un problème d'affectation et d'ordonnancement d'un ensemble de productions sur un ensemble d'usines de caractéristiques différentes avec des temps de changement entre produits. Comme ces temps peuvent être longs et coûteux, il est important de les considérer pour s'assurer

du respect des capacités de production des usines et ne pas engendrer de solutions trop coûteuses. Il n'est donc pas possible, à ce niveau, de ne regarder que l'affectation des tâches aux usines, mais il est nécessaire de proposer un séquencement de ces tâches permettant d'assurer le respect des capacités de production. Ainsi, si l'on considère les usines comme des machines, le problème est équivalent au problème d'ordonnancement sur machines parallèles non liées avec temps de changement dépendant de la séquence, présenté au chapitre 3 et peut alors être décrit à l'aide du programme linéaire **PL1** en utilisant les substitutions suivantes.

$$\begin{array}{l|l|l} J \to J_r & Tp_{im} \to Tp_{if} & Tc_{ijm} \to Tc_{ijf} \\ P \to F_r & Cp_{im} \to Cp_{if} + Ct_{if} & Cc_{ijm} \to Cc_{ijf} \end{array}$$

Nous obtenons alors le programme linéaire

$$\mathbf{PL1}(J_r,\ F_r,\ Tp_{if},\ (Cp_{if} + Ct_{if}),\ Tc_{ijf},\ Cc_{ijf},\ B_f)$$

4.4.4 Niveau 3 : Les fabriques - F_f

Ce niveau est composé des fabriques (ou usines), comportant un ensemble de lignes de production en parallèle. Chaque usine doit déterminer l'ordonnancement de ses lignes de production en fonction de l'ensemble des tâches à réaliser. Les données manipulées sont alors :

Données :

J_f	: Ensemble des tâches à réaliser par l'usine f $(1 \ldots \lvert J_f \rvert)$.
M_f	: Ensemble des lignes de l'usine f $(1 \ldots \lvert M_f \rvert)$.
Cp_{im}	: Coût de production de la tâche i sur la ligne m.
Tp_{im}	: Temps de production de la tâche i sur la ligne m.
Cc_{ijm}	: Coût de changement de la tâche i à la tâche j sur la ligne m.
Tc_{ijm}	: Temps de changement de la tâche i à la tâche j sur la ligne m.
B_m	: Capacité de la ligne m.

Résultats :
O_m : Ordonnancement de la ligne m de l'usine.

Le problème à résoudre dans chacune des usines est clairement un problème de machines parallèles non liées avec temps de changement dépendant de la séquence. Il peut donc également être modélisé par le programme linéaire **PL1** en effectuant les substitutions suivantes.

$$\begin{array}{l|l|l} J \to J_f & Tp_{im} \to Tp_{im} & Tc_{ijm} \to Tc_{ijm} \\ P \to M_f & Cp_{im} \to Cp_{im} & Cc_{ijm} \to Cc_{ijm} \end{array}$$

Le problème est ainsi décrit par

$$\mathbf{PL1}(J_f,\, M_f,\, Tp_{im},\, Cp_{im},\, Tc_{ijm},\, Cc_{ijm},\, B_m)$$

4.4.5 Niveau 4 : Les lignes de production - M_m

Les éléments de ce niveau sont les lignes de production. Chaque ligne est différentes des autres de par ses caractéristiques techniques et ses coûts associés. Chaque ligne doit effectuer l'ordonnancement donné par le niveau immédiatement supérieur (l'usine). Aucune décision n'est prise à ce niveau.

Données :
O_m : Ordonnancement pour la ligne m.

Résultats :
Exécution de l'ordonnancement.

4.4.6 Communications entre les niveaux

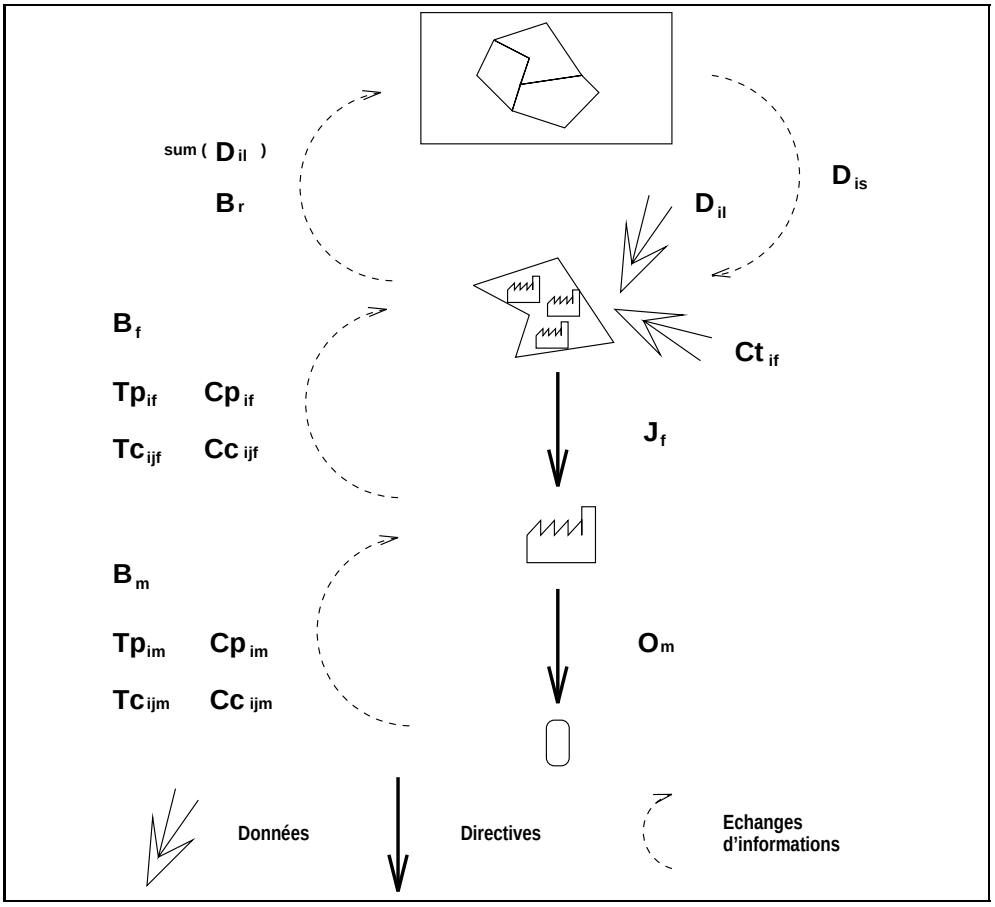


FIG. 4.2 - *Communications entre cellules*

Ayant précisé l'ensemble des données et résultats intervenant aux différents niveaux de la structure hiérarchique, il est maintenant nécessaire de définir les liens entre ces différents niveaux. Ces liens sont décrits par la figure 4.2. Il apparaît que chaque cellule a besoin d'informations concernant les cellules dépendant d'elle.

4.4.7 Commentaires sur cette décomposition

Nous venons de voir quels sont les problème à résoudre à chacun des niveaux de la structure hiérarchique et avons mis en évidence la similitude entre le problème à résoudre au niveau des régions et le problème au niveau des usines. En effet, à la nature des données près, ces problèmes peuvent se modéliser à l'aide du programme linéaire **PL1**, tout comme le problème à résoudre par l'approche centralisée. Ainsi, ordonnancer de façon centralisée un ensemble de productions sur un ensemble de lignes distantes est, par nature, le même problème que celui de l'affectation (en tenant compte du séquencement) au sein des régions d'un ensemble de productions sur différentes usines ou encore le même problème que celui de l'ordonnancement des lignes de production au sein des usines.

Ces trois problèmes peuvent donc être résolus de la même façon. La seule différence est la taille des problèmes à résoudre qui permet ou ne permet pas d'utiliser des méthodes exactes. Nous allons donc, pour résoudre ces différents problèmes, utiliser les méthodes présentées dans le chapitre 3. Lorsque le nombre d'éléments à considérer sera assez petit, nous utiliserons la procédure de séparation et évaluation. Au contraire, si le nombre d'éléments est trop important (cas de la résolution centralisée, par exemple), nous utiliserons la procédure itérative de l'Oracle.

Nous allons, dans la section suivante, comparer la résolution du problème par l'approche centralisée avec l'approche hiérarchisée.

4.5 Comparaisons des deux approches

Dans cette partie, nous comparons, pour la résolution de problèmes multisites et multiproduits, deux approches possibles : l'approche centralisée résolvant le problème pour toutes les lignes de production en même temps et l'approche par décomposition. Le premier élément de comparaison concerne la taille des problèmes à résoudre dans les différents cas. Puis nous donnerons quelques indications sur la mise en œuvre de chacune des méthodes et détaillerons les algorithmes développés. Enfin, nous terminerons par une comparaison des coûts des solutions trouvées par les deux approches.

4.5.1 Indications sur la taille des problèmes

Pour discuter de l'intérêt de la décomposition spatiale d'un problème d'optimisation globale d'une production multisite et multiproduit, nous décrivons le problème d'une division industrielle particulière. Cette division est composée de 10 usines, pour un total de 16 lignes de production. Le nombre de commandes à traiter par période de 15 jours est de l'ordre de 80. A chaque commande, nous associons une tâche définie par le type de produit à réaliser, sa quantité et le lieu de livraison.

Nous avons vu précédemment que le problème vu de façon centralisée pouvait être modélisé de la même façon que le problème existant au niveau des régions ou encore que le problème existant au niveau des usines. Nous allons donc comparer, pour chacun de ces problèmes la taille de leur modèle associé. Pour chacun d'eux, nous évaluons le nombre de variables (borné par $|J|^2 \times |M|$) et de contraintes (de l'ordre de $|J|^3$) nécessaires à leur description.

Le problème global

Le tableau 4.2 donne la taille du programme linéaire décrivant le problème vu dans son ensemble (approche centralisée). Il apparaît clairement que le nombre de variables et de contraintes est trop élevé pour espérer utiliser une méthode de résolution exacte. Nous sommes alors contraints d'utiliser des heuristiques telles que celles exposées au chapitre 3.

Centre	Nb lignes	Nb tâches	Variables	Contraintes
C1	16	80	102 400	512 000

TAB. 4.2 - *Le problème global*

Décomposition en régions

Une décomposition spatiale des usines en cinq régions, inspirée d'une répartition géographique réelle rencontrée chez Pechiney, est décrite dans le tableau 4.3. Le nombre d'usines composant chacune des régions et le nombre de tâches à considérer est réduit de façon significative, ce qui donne des programmes linéaires de plus petites tailles. Le nombre d'usines par région est relativement fixe et est déterminé par le niveau central, qui peut décider occasionnellement de déplacer une usine d'une région à une autre de façon à rééquilibrer la charge de travail des régions. Par contre, le nombre de tâches à réaliser dans chacune des régions peut varier d'une période à une autre en fonction des demandes. Ce tableau donne donc un ordre de grandeur sur les tailles des problèmes à résoudre au niveau des régions.

Régions	Nb usines	Nb tâches	Variables	Contraintes
R1	3	25	1 875	46 875
R2	2	20	800	8 000
R3	2	15	450	3 375
R4	2	10	200	1 000
R5	1	10	100	500

TAB. 4.3 - *Niveau 2: Les régions*

Au niveau des usines

Le tableau 4.4 montre les différents problèmes à résoudre dans chacune des dix usines de la division industrielle. L'usine numéro 1.2 représente la deuxième usine de la région 1. Tout comme dans le tableau 4.3, le nombre de tâches à considérer par usine n'est pas fixe et dépend, entre-autre, du résultat obtenu au niveau de la région dont fait partie l'usine.

Usines	Nb lignes	Nb tâches	Variables	Contraintes
U 1.1	2	10	200	1 000
U 1.2	2	9	162	729
U 1.3	1	6	36	216
U 2.1	1	7	49	343
U 2.2	3	13	507	2 197
U 3.1	1	6	36	216
U 3.2	2	9	162	729
U 4.1	1	4	16	64
U 4.2	1	6	36	216
U 5.1	2	10	200	1 000

TAB. 4.4 - *Niveau 3: Les usines*

De cette décomposition spatiale résulte plusieurs sous-problèmes qui sont chacun de taille raisonnable et plus facile à résoudre que le problème global. En effet, le chapitre 3 nous a montré qu'il est possible, en utilisant une procédure de séparation et évaluation adaptée au problème, de trouver des ordonnancements optimaux pour des problèmes à trois machines, dix tâches ou encore, quatre machines, dix tâches en moins de une minute (voire la plupart du temps en moins de trente secondes).

4.5.2 Mise en œuvre des deux approches

Comme nous l'avons présenté dans la partie précédente, la résolution centralisée du problème peut se faire en utilisant la procédure itérative de l'Oracle présenté au chapitre 3. Pour la résolution hiérarchisée, les différents sous-problèmes seront résolus soit par la procédure de séparation et évaluation, soit par l'Oracle, en fonction du nombre d'éléments en jeux.

Afin de comparer les deux approches de façon équitable, nous les avons mises en œuvre de manière à leur autoriser à peu près le même temps de recherche. Ainsi, lors de l'exécution de l'approche hiérarchisée (par décomposition), nous mémorisons le nombre de fois où l'Oracle est appelé ($NbOracle$). Puis l'exécution centralisée lance $NbOracle$ fois l'Oracle (ou au moins une fois) avec des valeurs de α très légèrement différentes de façon à visiter le plus grand nombre de solutions possible.

L'algorithme de l'approche par décomposition est exposé ci-dessous :

Pour chaque région faire
    // Détermination des productions à réaliser par chaque usine
    // (données agrégées)
    Agrégation des données au niveau de la région
    Si le nombre de composantes est trop élevé (i.e $M \times M \times N > 250$)
        alors
            Exécuter l'Oracle
            Incrémenter $NbOracle$
        sinon
            Exécuter une Procédure de Séparation et Évaluation
    FinSi
    Pour chaque usine faire
        // Détermination des ordonnancements de chaque ligne
        // (données non agrégées)
        Si le nombre de composantes est trop élevé (i.e $M \times M \times N > 250$)
            alors
                Exécuter l'Oracle
                Incrémenter $NbOracle$
            sinon
                Exécuter une Procédure de Séparation et Évaluation
        FinSi
    FinPour
FinPour

Schéma de l'approche par décomposition

L'approche centralisée consiste à appeler plusieurs fois l'Oracle avec des valeurs de α légèrement modifiées. Le schéma utilisé est le suivant.

$Nbiter = 0$, $\alpha = 0,5$
Tantque $Nbiter < NbOracle$
    $\alpha = \alpha + (Nbiter/NbOracle)/10$;
    Exécuter l'Oracle, gardez la meilleure solution
    Nbiter++
FinTantque

Schéma de l'approche centralisée

4.5.3 Expérimentations

Nous reportons ci-dessous (Tableau 4.5) les résultats de comparaisons entre le coût des solutions données par la procédure centralisée et celles données par l'approche par décomposition. L'expérimentation a été réalisée sur un Pentium 133MHz. Nous avons généré aléatoirement des problèmes de différentes tailles. Les problèmes sont dénotés de la façon suivante : R indique le nombre de régions, U, le nombre total d'usines, L le nombre total de lignes de production et P le nombre total de productions à réaliser. Les nombres indiqués sont en fait les moyennes des dix exécutions réalisées pour chaque type de problème. La répartition des usines par régions est faite de façon homogène (c'est-à-dire en répartissant à peu près autant d'usines dans chaque région). La répartition des lignes de production en usines est faite de telle manière que les usines comportent entre une à trois lignes de production (cas réel).
Afin de générer des problèmes ayant de fortes chances d'être réalisables, nous les avons construits de la manière suivante : lecture d'un fichier donnant la structure de la division industrielle (c'est-à-dire la répartition des usines en régions et le nombre de lignes par usine). Puis, pour chaque région, en tenant compte de sa capacité, nous déterminons un nombre de productions à réaliser. Enfin, pour l'ensemble des productions nous générons de façon aléatoire les temps et coûts de production, les temps et coûts de changement et les coûts de distribution. Ces derniers sont calculés en fonction de la région de commande des productions (pour se rapprocher au mieux de la réalité).
A la lecture du tableau 4.5, il apparaît que l'approche par décomposition donne en moyenne de meilleures solutions que l'approche centralisée. En effet, le surcoût moyen de l'approche centralisée, sur les exemples testés est de 6%. Une étude plus précise nous permet de voir que lorsque le nombre moyen de productions à réaliser par chacune des lignes augmente, la supériorité de l'approche par décomposition augmente également. Ceci peut être expliqué par le fait que, dans la

Problèmes *R-U-L-P*	Approche par décomposition		Approche centralisée	
	Coût de la solution	Temps de de résolution	Coût de la solution	Temps de de résolution
6-14-24-109	1194	178	1215	161
6-14-24-133	1609	159	1805	143
9-21-35-160	1905	210	1968	272
9-21-35-195	2237	285	2561	619
12-30-44-170	1751	175	1834	270
12-30-44-205	2225	307	2355	547

TAB. 4.5 - *Comparaisons des coûts des solutions données par les deux approches et des temps de résolution utilisés*

phase d'amélioration de l'approche centralisée, il y a plus d'échanges possibles à réaliser et également plus d'optima locaux dans lesquels la méthode peut rester bloquée.

Tout semble donc être en faveur de l'approche par décomposition. Il est tout de même nécessaire d'émettre quelques réserves concernant les difficultés de mises en œuvre de cette approche. En effet, au préalable, un certain nombre de paramètres sont à ajuster. Ces paramètres sont la répartition des usines et des demandes en régions. Il est nécessaire, pour que la méthode fonctionne correctement, que les charges de travail et les capacités de production soient bien équilibrées. Pour certains jeux de tests (pour lesquels les capacités de production du système global permettaient tout juste de répondre aux demandes), l'approche par décomposition n'a pas été capable de trouver une solution réalisable, à cause de la préaffectation des demandes aux régions. Par contre, l'approche centralisée a réussi, en affectant certaines productions à des sites plus éloignés, à respecter les capacités de production. Cela montre donc l'importance du rôle du niveau central qui doit gérer la répartition des outils de production à partir des données à long terme.

Dans le cadre de l'ordonnancement de lignes de production distantes, il pourrait être intéressant de développer une méthode hybride qui couplerait les avantages de la méthode centralisée et de la méthode par décomposition. Cette méthode aurait le fonctionnement suivant : au niveau central, la méthode centralisée calculerait un ordonnancement réalisable pour l'ensemble des lignes. Puis en s'appuyant sur cet ordonnancement, les productions seraient réparties dans les différentes régions. Enfin, connaissant la répartition des productions et des usines par régions, il serait possible d'exécuter la méthode par décomposition. Cela permettrait de s'assurer de l'existence d'une solution réalisable avec la répartition donnée par la méthode centralisée et de profiter des performances de la méthode par décomposition.

4.6 Conclusion

L'objectif de ce chapitre était de mettre en évidence l'importance de la cohérence entre les aspects de productions et les aspects de distribution dans le cas d'une entreprise multisite.

La particularité de notre étude est de considérer au plus haut niveau (au niveau des différents sites) non seulement les aspects concernant l'affectation, mais également les aspects d'ordonnancement. En effet les temps de changement existant entre la fabrication de deux produits différents pouvant être tellement longs qu'il n'est pas possible, à notre avis, de les ignorer lors de la détermination des productions à réaliser par les différents sites. Ceci aurait pour conséquence d'imposer aux sites des plans de production non réalisables à cause de la capacité finie des lignes. C'est pourquoi, afin de s'assurer de la réalisabilité d'un plan de production, celui-ci doit être défini en tenant compte du séquencement possible entre les productions.

Pour traiter ce problème d'ordonnancement sur plusieurs lignes de production distantes, nous avons étudié deux approches.

La première, la résolution centralisée, considère l'ensemble de la division industrielle comme un tout et procède en un seul temps à l'ordonnancement des toutes les lignes, sans tenir compte de leur répartition géographique. Bien sûr, les coûts de distribution sont pris en compte dans l'évaluation de la solution, mais ne sont pas réellement utilisés pour guider la recherche d'une meilleure solution.

La deuxième approche étudiée est une approche par décomposition que nous proposons pour traiter des problèmes multisites avec de forts coûts de distribution. Cette approche décompose le problème global en plusieurs problèmes de tailles plus petites. Au niveau des régions, des données agrégées sont utilisées pour pouvoir considérer chaque usine comme une seule ligne de production.

Les résultats montrent que cette deuxième approche donne, en général, de meilleurs résultats que l'approche centralisée. Il est donc possible de conclure que la perte de précision induite par l'utilisation, pour certains sous-problèmes, de données agrégées est compensée par l'utilisation de méthodes plus précises pour la résolution. Nous rappelons tout de même, qu'il est nécessaire, avant d'utiliser la méthode par décomposition, de procéder avec soin à la répartition des outils de production. Nous suggérons l'utilisation d'une méthode hybride liant les deux approches.

Ce travail a été mené dans un cadre monopériode et il conviendrait de l'étendre au cas multipériode. Cet aspect est déjà légèrement pris en compte dans l'approche par décomposition, puisque le rôle du niveau haut de la structure hiérarchique est

justement d'avoir une vue globale du système tant au niveau spatial que temporel. C'est en effet ce niveau qui détermine, en fonction des demandes actuelles et des prévisions de demandes, les productions à réaliser en avance, afin d'anticiper les périodes de fortes demandes.

Au niveau de l'ordonnancement, une façon assez simple d'introduire les aspects multipériodes, serait de donner, à chaque production, une date d'échéance qui indiquerait la fin de la période pour laquelle la production est demandée. Puis il nous faudrait introduire des coûts de stockage pour les productions réalisées en avance et des pénalités de retard lorsque la production est réalisée en retard. Si aucun retard n'est accepté, ces pénalités doivent être prohibitives, de façon à privilégier le respect des délais. La méthode de décomposition proposée étant rapide, elle permet de résoudre de plus gros problèmes, et il est donc tout à fait réalisable de l'utiliser, moyennant l'introduction de délais, de coûts de stockage et de pénalités de retard, pour des problèmes multipériodes.

De plus, la rapidité d'exécution de l'approche par décomposition permet également de l'utiliser assez fréquemment, et pourquoi pas dès qu'un changement survient. Ainsi, en environnement dynamique et réactif, lorsqu'une perturbation se produit, cette méthode a l'avantage de pouvoir être réutilisée, soit entièrement, ce qui remet en cause toutes les décisions concernant l'ensemble du système de production, soit à un niveau plus local de façon à absorber la perturbation en ne modifiant qu'un petit nombre d'ordonnancements.

Chapitre 5

Étude du cas Europe

Dans le cadre de nos contacts et collaborations avec les chercheurs du Centre de Recherche de Voreppe (CRV) de la société Pechiney, nous nous sommes intéressés à un cas particulier de la problématique générale exposée au chapitre 0 : l'étude de la division industrielle européenne multisite de fabrication de boîtes de boisson (en aluminium et en acier). Un groupe de travail, comprenant deux représentants du CRV et quelques personnes de notre laboratoire, a réfléchi sur la modélisation de ce cas particulier. Notre travail a consisté en la validation du modèle, et l'exécution de simulations de façon à mettre en évidence les possibilités et limites de cette modélisation [43].

La modélisation et la résolution du problème Europe ont été rendues possible grâce aux recherches, menées par les chercheurs de Pechiney, visant à réduire intelligemment la taille du problème. Les différentes restrictions proposées nous ont permis de modéliser les différents flux de produits sous forme d'un graphe. Ce graphe a facilité le dialogue avec les personnes de Pechiney, et a permis une meilleure compréhension du problème. Puis, dans un deuxième temps, nous avons proposé une modélisation formelle, sous forme d'équations, basée sur le graphe. Enfin, des résolutions de petits problèmes ont permis de valider la modélisation proposée.

Nous exposons dans ce chapitre les différentes phases suivies pour la résolution du cas Europe. Ainsi, nous commençons par détailler les hypothèses et les restrictions de départ. Puis nous présentons la modélisation en exposant, dans un premier temps, celle correspondant au problème monoperiode, et dans un deuxième temps, son extension au problème multipériode. Enfin, l'expérimentation menée donne des indications sur les tailles des problèmes abordables par une telle modélisation et sur l'influence de différents paramètres de résolution.

5.1 Hypothèses et restrictions

Les chercheurs de Pechiney ayant un besoin urgent d'optimiser la division industrielle européenne fabriquant les boîtes de boisson, ont étudié de près les données du problème et les habitudes de production. Suite à ce très important travail de récolte des données, ils ont formulé des hypothèses et des restrictions.

Nous exposons dans les paragraphes qui suivent, les différentes composantes du problème, les restrictions sur les polyvalences des lignes et sur les changements de fabrication, ainsi que les conséquences de ces restrictions sur les états possibles des lignes.

5.1.1 Composantes du problème

Tout d'abord, les différentes composantes du système ont été dénombrées. L'énumération de ces composantes, exposée ci-après, ne décrit pas précisément la division industrielle européenne, dont les données restent confidentielles, mais respecte la taille du problème et les ordres de grandeur.

10 usines composées de une à trois lignes de production et dispersées sur l'ensemble de la zone géographique considérée, à savoir l'Europe (cf fig 5.1).

16 lignes de production réparties dans les usines.

16 produits caractérisés en fonction de leur métal (acier, aluminium), leur format (25cl, 33cl, 44cl, 50cl) et leur col de fermeture (deux types de cols). Les produits excentriques (27.5cl, 35cl, 37.5cl, 56.8cl), représentant infiniment peu de volume, ont été éliminés du problème. Les temps de changement, nécessaires pour passer d'un produit à un autre, peuvent durer jusqu'à six jours pour certains changements de format. Les changements de fermetures (cols) sont beaucoup moins pénalisants (quelques heures) et ne sont donc pas considérés. Nous appellerons, *produit hors-col*, un produit sans spécification de col. Il existe donc 8 types de produits hors-col (2 métaux $\times$ 4 formats).

12 périodes d'un mois qui permettent de modéliser une année entière et de tenir compte de la saisonalité des demandes.

50 stocks dont ceux des usines : soit 10 stocks en usine et 40 stocks répartis sur le territoire.

300 lieux de livraison produit-client. Les lieux sont caractérisés par un client géographique et un produit. Si un client commande plusieurs produits, il est compté comme plusieurs lieux de livraison produit-client.

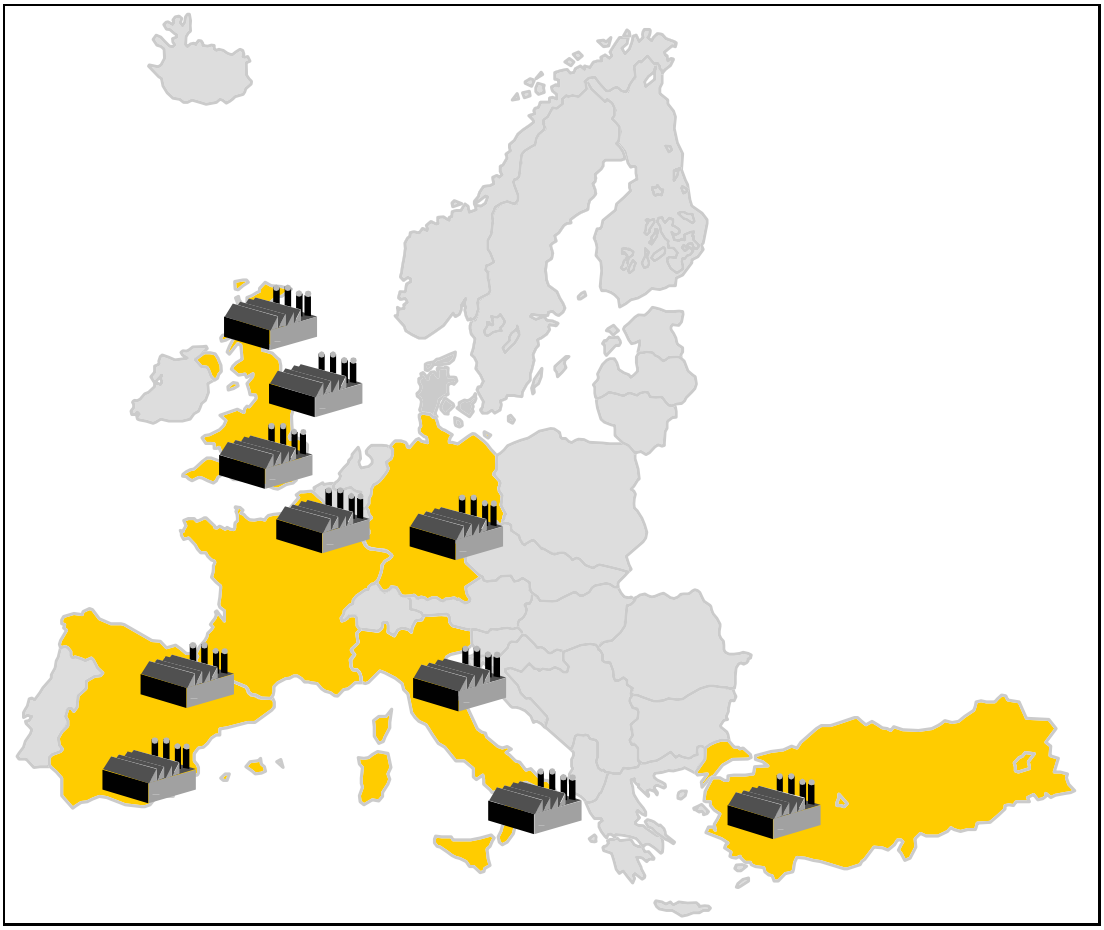


FIG. 5.1 - *Division industrielle européenne*

5.1.2 Polyvalence des lignes

Dans la problématique telle qu'elle avait été posée initialement, chacune des lignes pouvait faire tous les produits avec une cadence dépendant de la ligne et du produit. En fait, d'après les contraintes technologiques et les habitudes de production, il s'avère qu'en pratique la polyvalence des lignes est fortement réduite. En effet, une ligne fabrique, en réalité, au plus trois produits hors-col différents. Dans l'ensemble du système de production, il a été dénombré les lignes suivantes :

8 lignes ne faisant qu'un produit hors-col (Lignes de type $L1$).

6 lignes faisant jusqu'à deux produits hors-col (Lignes de type $L2$).

2 lignes faisant jusqu'à trois produits hors-col (Lignes de type $L3$).

Les interlocuteurs de Pechiney ont formulé le souhait de garder, pour les lignes, les polyvalences observées au cours du travail de caractérisation du système de production.

5.1.3 Changements de production

Lors du passage de la production d'un produit à un autre, un temps de changement est à considérer. Ces temps de changement pénalisent fortement la production, puisqu'ils réduisent le temps disponible pour la fabrication des produits. Pour limiter ces pénalisations, pour répartir les changements sur l'ensemble des lignes, et aux vues des habitudes de production, certaines restrictions sont prises sur les possibilités de changement de production. On autorise, en fait, au plus un changement par ligne et par période, à condition que ce changement ne soit pas situé au moment du changement de période. Ainsi, une ligne peut fabriquer pendant une période au plus deux produits hors-col différents (même si la ligne est de type $L3$). De plus, en interdisant les changements de produit au moment du changement de période, on impose qu'une ligne, en début de période, fabrique le même produit qu'à la fin de la période précédente.

5.1.4 Conséquences sur les séquences des lignes

Enumération des séquences admissibles

Les différentes restrictions prises sur le nombre de produits par ligne et le nombre de changements autorisés par période limitent très fortement le nombre de séquences possibles pour chaque période pour une ligne, et en permettent l'énumération complète donnée dans le tableau 5.1.
Remarquons que A, B et C représentent des produits hors-col, et doivent donc être vus comme des produits définis par leur format et leur métal.

Type de ligne	Produits possibles	Séquences admissibles par période
$L1$	A	A
$L2$	A, B	A, B, AB, BA
$L3$	A, B, C	A, B, C, AB, AC, BA, BC, CA, CB

TAB. 5.1 - *Séquences admissibles par période en fonction du type de ligne*

Enchaînement des séquences

Rappelons que les contraintes sur les changements de production imposent à une ligne de commencer une période avec la même production que celle fabriquée à la fin de la période précédente. Ainsi, des liens sont créés entre les séquences réalisables des périodes adjacentes d'une même ligne, pour garantir le respect de ces contraintes d'adjacence.

Séquence	Prédécesseurs compatibles	Successeurs compatibles
A	A, BA	A, AB
B	B, AB	B, BA
AB	A, BA	B, BA
BA	B, AB	A, AB

TAB. 5.2 - *Compatibilités d'adjacence pour une ligne de type $L2$*

Prenons l'exemple d'une ligne de type $L2$ ayant suivi la séquence AB pendant la période t. Les différents choix possibles pour la période $t+1$, compte tenue des séquences et des changements autorisés, sont toutes les séquences commençant par B, soient B ou BA. Le tableau 5.2 récapitule l'ensemble des prédécesseurs et successeurs compatibles avec les séquences d'une ligne de type $L2$.

Ainsi, il n'est plus nécessaire de s'intéresser aux produits façonnables sur une ligne et aux enchaînements réalisables, mais il est possible de ne considérer que les différentes séquences admissibles sur chaque ligne. Le temps de changement induit par la réalisation, sur une ligne, d'une séquence comportant plusieurs produits, pendant une période, est alors directement soustrait à la capacité totale de production de la ligne pour cette période. Il reste ensuite à répartir la durée de production restante entre les produits de la séquence.

5.2 Modélisation mono-période sous forme de flots

La modélisation est basée sur le suivi des produits depuis leur fabrication jusqu'à leur livraison. En fait, la demande des clients "tire" le flux des produits venant des stocks et plus en amont, des lignes. Cette modélisation fait appel aux différentes entités qui représentent les trois étages principaux de la problématique : les lignes de production, indexées par m, les stocks, indexés par w et les clients, indexés par c. Dans cette partie, nous exposons un modèle mono-période. Nous donnerons, dans la partie suivante, les ajouts à réaliser pour obtenir un modèle multi-période [50].

Pour la modélisation, nous divisons les lignes de production en deux entités (les états des lignes de production qui déterminent quelles sont les séquences en cours de réalisation sur les lignes et les lignes elles mêmes), ce qui donne une modélisation à quatre niveaux (cf figure 5.2) : les états des lignes de production, les lignes, les stocks et les clients. Nous décrivons ci-dessous, les paramètres, variables et contraintes des trois jonctions inter-étages : le réseau états des lignes de production-lignes (réseau PL), le réseau lignes-stocks (réseau LS) et le réseau stocks-clients (réseau SC).

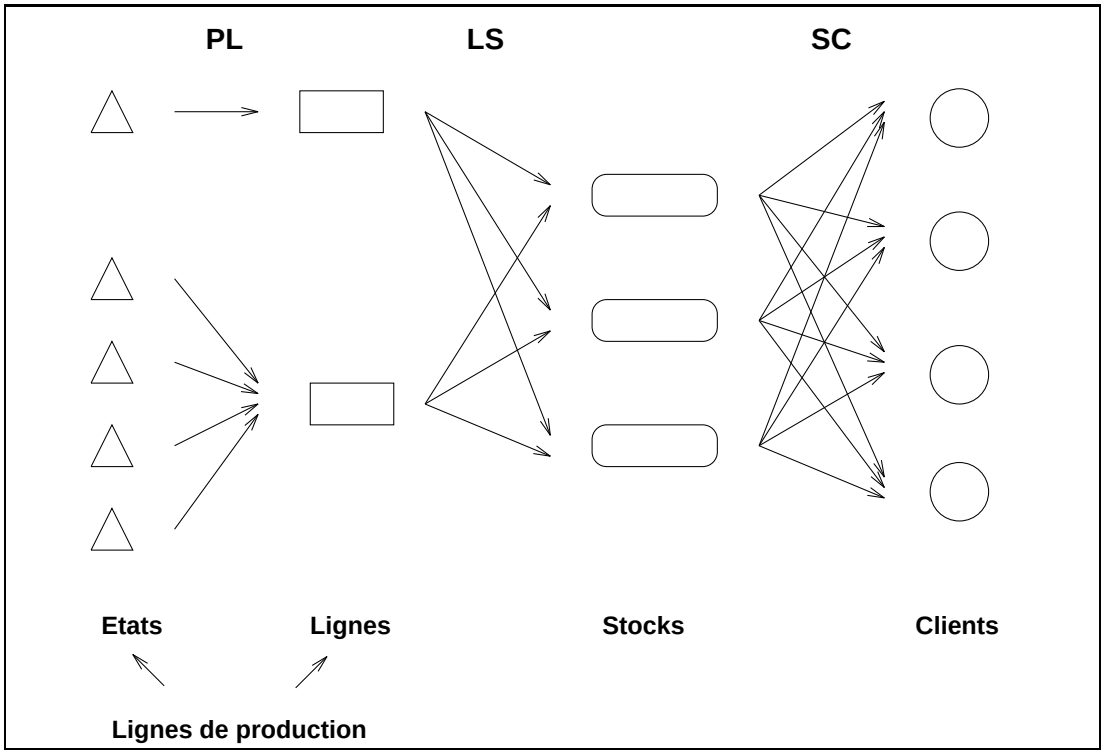


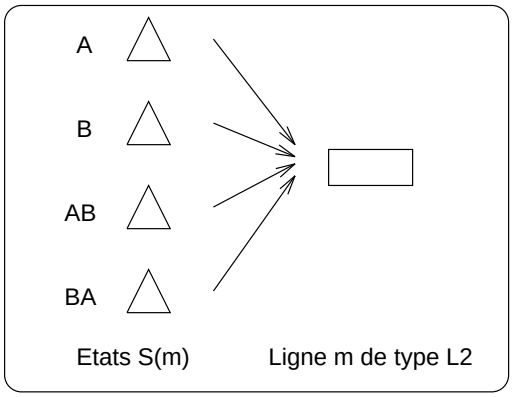
FIG. 5.2 - *Modélisation des flux de produits*

5.2.1 Le réseau PL

Le réseau PL est le lien entre l'état des lignes de production et les lignes elles-mêmes.

Modélisation graphique

Comme nous l'avons vu dans la section précédente (5.1.4), une ligne de production a un ensemble fini de séquences admissibles (ou états), qui sont énumérables. Une ligne de production est décomposée en ses états possibles, représentés par des triangles, et en la ligne elle-même, représentée par un rectangle.



Représentation d'une ligne de type $L2$: Les quatre triangles représentent les quatre séquences admissibles (A, B, AB, BA) de la ligne. Une seule de ces séquences est choisie par ligne et par période. Les produits fabriqués pendant une période, sur une ligne, sont regroupés à la fin de la ligne, moyennant des coûts de production.

Les différents éléments représentant les lignes de production sont :

États : Cette entité représente les différentes séquences possibles pour une ligne. Ainsi, une ligne de type $L1$ ne sera représentée que par un triangle, une ligne de type $L2$ par quatre triangles et une ligne de type $L3$, quant à elle, sera représentée par neuf triangles. Soit un total de $(8 \times 1) + (6 \times 4) + (2 \times 9) = 50$ nœuds, pour l'exemple étudié.

Lignes : Cette entité représente une ligne de production physique. Elle regroupe, en un seul nœud, tous les états possibles de la ligne, et permet d'avoir en sortie du réseau PL un seul nœud par ligne. Ceci a pour effet de limiter le nombre de liens dans le réseau LS, et donc le nombre de variables.

Modélisation formelle

Nous notons $S(m)$ l'ensemble des séquences admissibles pour la ligne m, les séquences étant indexées par s.
À cet étage, les données concernent les capacités de production des lignes (B_m), les temps (TP_{im}) et coûts (CP_{im}) de production des produits sur les lignes et les temps (TC_{sm}) et coûts (CC_{sm}) relatifs au changement associé à la séquence s de la ligne m.

De plus, deux variables sont utiles pour, d'une part définir la séquence choisie pour chacune des lignes et, d'autre part déterminer la quantité de produits fabriquée.

y_{ms} $= 1$ si la séquence s est choisie pour la ligne m,
$= 0$ sinon.
p_{im} : quantité de produit i fabriquée sur la ligne m.

Trois classes de contraintes sont associées à ce réseau. Tout d'abord, au niveau de la production, les contraintes (5.1) s'assurent qu'une et une seule séquence est choisie pour chacune des lignes. Puis, les contraintes (5.2) expriment qu'un produit i peut être fabriqué sur la ligne m seulement s'il appartient à la séquence choisie. Dans ces contraintes, $\mathcal{HV}$ représente une grande valeur (High Value). Nous pourrons spécifier pour chaque équation la valeur de $\mathcal{HV}$ en prenant, par exemple, $\mathcal{HV} = (B_m - TC_{sm})/TP_{im}$. Finalement, les contraintes (5.3) spécifient que le temps utilisé à la production plus le temps utile à l'éventuel changement de la séquence sur une ligne, ne doit pas dépasser la capacité de cette ligne.

$$\sum_{s \in S(m)} y_{ms} = 1 \qquad \forall m \qquad (5.1)$$

$$p_{im} - \sum_{s \in S(m)/i \in s} y_{ms} \times \mathcal{HV} \leq 0 \qquad \forall i, \forall m \qquad (5.2)$$

$$\sum_{i} p_{im} \times TP_{im} + \sum_{s \in S(m)} y_{ms} \times TC_{sm} \leq B_m \qquad \forall m \qquad (5.3)$$

$$p_{im} \geq 0 \qquad \forall i, \forall m \qquad (5.4)$$

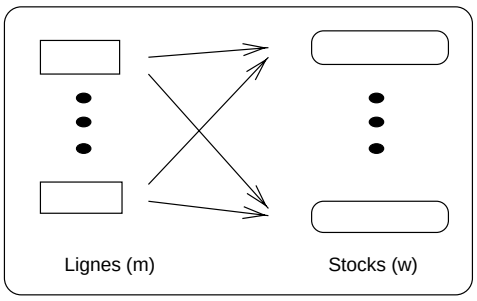
$$y_{ms} \in \{0, 1\} \qquad \forall m, \forall s \qquad (5.5)$$

5.2.2 Le réseau LS

Le réseau LS est le lien entre l'organe de production (les lignes de production) et l'organe de stockage.

Modélisation graphique

A cet étage, les produits sont envoyés depuis les lignes de production vers les stocks. Les stocks sont soit situés au sein d'une usine, soit en dehors des usines. Nous dénombrons, au total, 50 stocks.



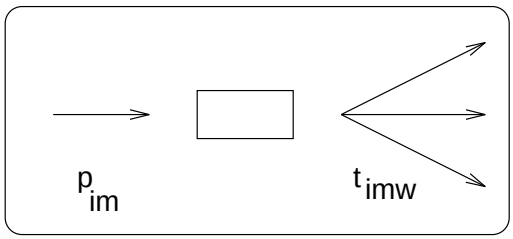
Les stocks sont représentés par des rectangles arrondis et les produits circulant le long des arcs (lignes-stocks) induisent des coûts de transport. Lorsque un stock est situé au sein d'une usine, le coût de transport entre une ligne de cette usine et ce stock est nul.

Modélisation formelle

Soit :
t_{imw} : quantité de produit i transportée depuis la ligne m jusqu'au stock w.
s_{iw} : quantité de produit i traversant le stock w.

Dans le modèle mono-période, les stocks ne sont pas utilisés en temps que tel, puisqu'il n'est pas possible d'anticiper la demande. Ils sont donc des points de transit pour les produits.

Les contraintes (5.6) expriment que pour tout produit i, la quantité transportée depuis une ligne m jusqu'à l'ensemble des stocks, est égale à la quantité de produit i fabriquée sur m.



Les deuxièmes contraintes sont utilisées pour s'assurer du respect des capacités des stocks. La capacité du stock w (en volume) est notée K_w, tandis que le volume du produit i est noté V_i. Ainsi, la contrainte (5.7) s'assure que le volume des produits traversant un stock ne dépasse pas la capacité de ce stock et la contrainte (5.8) positionne le niveau des stocks à tout ce qui circule par eux.

$$p_{im} - \sum_{w} t_{imw} = 0 \qquad \forall m, \forall i \qquad (5.6)$$

$$\sum_{i} s_{iw} \times V_i \leq K_w \qquad \forall w \qquad (5.7)$$

$$s_{iw} - \sum_{m} t_{imw} = 0 \qquad \forall w, \forall i \qquad (5.8)$$

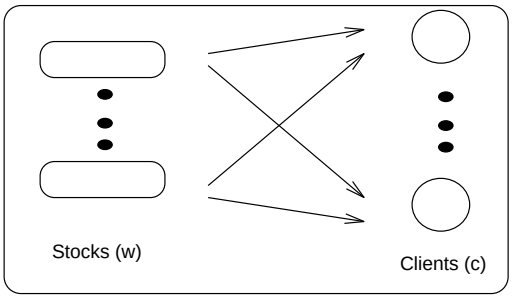
$$t_{imw} \geq 0 \qquad \forall i, \forall m, \forall w \qquad (5.9)$$

5.2.3 Le réseau SC

Le réseau SC représente la livraison des produits aux clients depuis les stocks.

Modélisation graphique

Un client est associé à la localisation d'un lieu de livraison, tel qu'il a été défini au début du chapitre. Ainsi, à un client ne correspond qu'un seul type de produit.



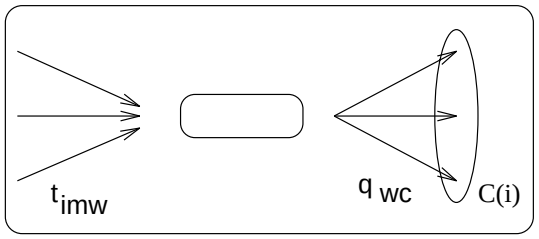
La demande des clients (représentés par des cercles) doit être satisfaite. Des coûts de distribution sont associés aux flux des produits le long des arcs Stocks-Clients.

Modélisation formelle

Nous définissons les variables suivantes :
q_{wc} : quantité de produit transportée depuis le stock w jusqu'au client c.

Nous remarquons qu'il n'est pas nécessaire de spécifier le produit transporté jusqu'au client, puisque chaque client ne commande qu'un produit, qui est donc connu.

Les contraintes (5.10) expriment que pour tout produit i, la quantité de produit livrée, depuis un stock w, à l'ensemble des clients ayant commandé i (noté $C(i)$) est égale à la quantité de ce produit transportée jusqu'à ce stock.



Ce réseau garantie également que chaque client reçoit la quantité de produit commandée, grâce aux contraintes (5.11).

$$\sum_m t_{imw} - \sum_{c \in C(i)} q_{wc} = 0 \qquad \forall w, \forall i \qquad (5.10)$$

$$\sum_w q_{wc} = D_c \qquad \forall c \qquad (5.11)$$

$$q_{wc} \geq 0 \qquad \forall w, \forall c \qquad (5.12)$$

5.2.4 Fonction objectif mono-periode

La fonction objectif concerne la minimisation de l'ensemble des coûts. Dans le problème mono-période, les coûts sont composés des coûts de production, de changement de production et de transport (lignes-stocks et stocks-clients). La fonction objectif est exprimée par l'équation (5.13)

$$\begin{aligned} min \quad & \sum_{m} \left(\sum_{i} CP_{im} \times p_{im} + \sum_{s \in S(m)} CC_{sm} \times y_{ms}\right) \\ & + \sum_{w} \left(\sum_{i} \sum_{m} CT_{mw} \times t_{imw} + \sum_{c} CT_{wc} \times q_{wc}\right) \end{aligned} \tag{5.13}$$

Le modèle proposé pour le problème mono-période est le programme linéaire mixte $\mathcal{P}$ consistant en la minimisation de la fonction objectif (5.13) sous les contraintes (5.1-5.12).

5.3 Modélisation multi-période

Le modèle $\mathcal{P}$ exposé dans la section précédente représente le système pendant une seule période. Comme nous sommes dans un environnement où la demande varie fortement d'une saison à l'autre, il est nécessaire d'étendre ce modèle de façon à pouvoir considérer plusieurs périodes. Pour cela, nous réutilisons le programme linéaire mixte $\mathcal{P}$, autant de fois que le nombre de périodes souhaitées. Les périodes sont indexées par t et les différentes copies de $\mathcal{P}$ sont distinguées par un exposant qui indique la période à laquelle elles font référence. De la même manière, tous les paramètres et variables liés à la période t sont marqués de l'exposant t (exemple: t^t_{imw} = quantité de produit i transportée depuis la ligne m jusqu'au stock w pendant la période t).

Il est maintenant nécessaire de lier correctement l'ensemble de ces problèmes $\mathcal{P}^t$. Deux types de liens existent entre deux périodes adjacentes. Ils concernent le stockage des produits et le choix des séquences des lignes de production.

La figure 5.3 présente le modèle du problème multi-période. Sur cette figure, les liens entre les stocks de différentes périodes sont représentés car ils correspondent physiquement à des flux de produits, tandis que les liens existant entre séquences de périodes adjacentes ne le sont pas, car ils ne représentent rien physiquement, et ne sont que des liens logiques. Ils seront exprimés plus loin sous forme d'équations.

5.3.1 Le stockage

De façon à anticiper les périodes de fortes demandes, il est nécessaire de produire en avance certains produits puis de les stocker.

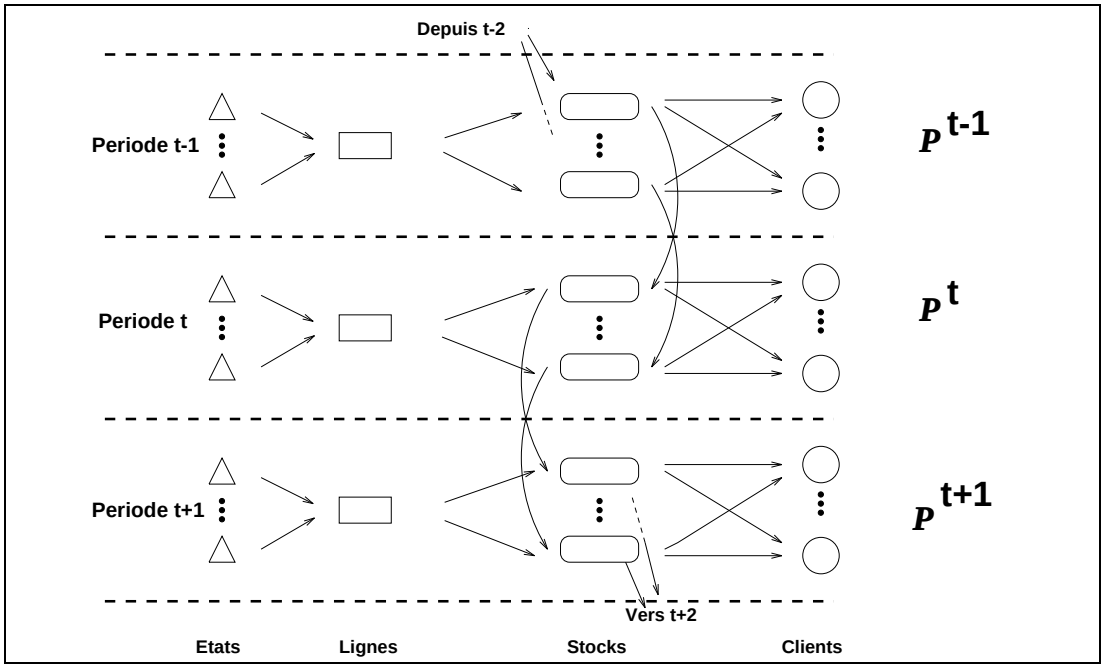
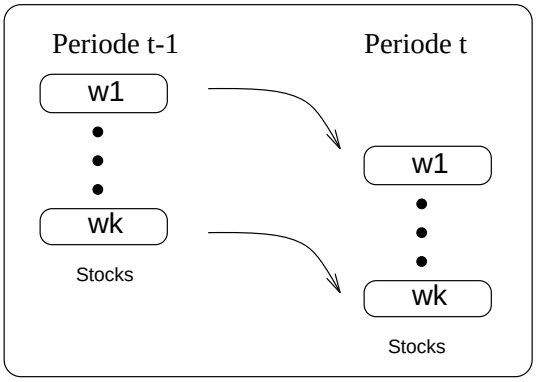


FIG. 5.3 - *Le modèle multi-période*

Modélisation graphique



Des arcs sont nécessaires pour relier un même stock entre deux périodes adjacentes. Les produits circulant le long de ces arcs représentent les produits restant en stock, d'une période à l'autre. Des coûts de stockage sont associés à ces flux de produits.

Modélisation formelle

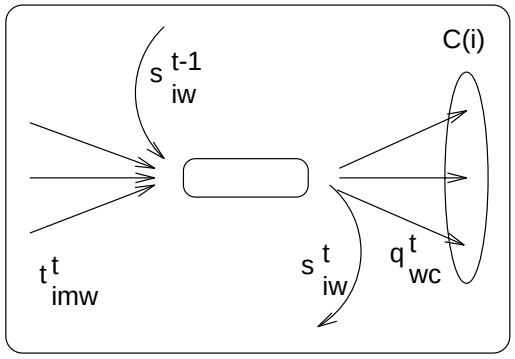
Lors de la présentation du modèle monopériode, nous avons proposé les contraintes 5.7 pour veiller au respect des capacités de stockage. Ces contraintes, en environnement multipériode, vont procéder à une vérification de ces capacités en fin de chaque période (étant données les définitions des différentes variables). Cela peut donc apparaître comme une simplification du modèle qui devrait vérifier à tout instant que la capacitée est respectée. En fait cela peut être justifié par le fait que les commandes des clients sont passées pour certaines périodes. Ainsi, dès qu'une commande, qui doit être livrée dans la période courante est prête, elle est envoyée au client. De cette façon, en cours de période, le volume de produits stockés ne

dépasse pas celui de la fin de la période qui correspond aux commandes produites en avance pour les périodes futures.

D'autre part, les autres contraintes à satisfaire visent la conservation des flux de produits.

Les contraintes (5.8') veillent à l'équilibrage des flux de produits. En fait, les contraintes 5.8 et 5.10 du modèle mono-période ne sont pas complètes pour le problème multi-période et doivent être remplacées par la contrainte (5.8').

Les flux de produits arrivant à un stock sont composés des produits transportés depuis l'ensemble des lignes vers ce stock, plus les produits déjà en stock à la fin de la période précédente. De la même manière, les flux de produits sortant d'un stock sont composés des produits livrés aux clients et des produits restant en stock pour la période suivante.



$$\sum_m t^t_{imw} + s^{t-1}_{iw} - \sum_{c \in C(i)} q^t_{wc} - s^t_{iw} = 0 \qquad \forall t, \forall w, \forall i \qquad (5.8')$$

$$s^t_{iw} \geq 0 \qquad \forall t, \forall w, \forall i \qquad (5.14)$$

5.3.2 Choix des séquences

Le second type de liens existant entre les périodes concerne le choix des séquences pour chacune des lignes de production. Ces liens ne sont pas représentés graphiquement car ils ne reflètent pas de flux de produits.

Modélisation formelle

Dans la présentation du problème, nous avons indiqué que les changements de productions étaient interdits aux moment des changements de périodes. Il en découle, qu'une ligne de production doit commencer une période en fabriquant le même produit qu'à la fin de la période précédente. Ainsi, a chaque période, la séquence choisie doit être compatible avec les séquences choisies pour la période précédente et la période suivante. Nous définissons les ensembles de prédécesseurs $PR(s)$ et successeurs $SU(s)$ compatibles avec chaque séquence s d'une ligne donnée. Par exemple, pour une ligne de type $L2$, fabriquant les produits A et B, $PR(AB) = \{A, BA\}$ and $SU(AB) = \{B, BA\}$).

Il existe plusieurs façons d'exprimer les compatibilités et incompatibilités entre les séquences d'une même ligne pour des périodes adjacentes. Chacune de ces façons mène à un type de contraintes. Nous en proposons plusieurs ci-dessous.

Contraintes de succession ou de précédence

Tout d'abord, les contraintes de succession (*Succ.*) (5.15), indiquent que pour chaque ligne de production, une séquence est susceptible d'être choisie pour la période $t+1$ seulement si la séquence choisie pour cette même ligne pendant la période t appartient à l'ensemble de ses prédécesseurs.

$$y_{ms}^{t+1} - \sum_{s' \in PR(s)} y_{ms'}^{t} \leq 0 \qquad \forall t, \forall m, \forall s \in S(m) \tag{5.15}$$

De la même façon, il est possible de définir des contraintes de précédence (*Prec.*) (5.16), qui lient la période $t-1$ à la période t et indiquent qu'une séquence a pu être choisie pendant la période $t-1$ seulement si la séquence choisie pour la même ligne pendant la période t appartient à ses successeurs.

$$y_{ms}^{t-1} - \sum_{s' \in SU(s)} y_{ms'}^{t} \leq 0 \qquad \forall t, \forall m, \forall s \in S(m) \tag{5.16}$$

Contraintes d'incompatibilité

Il est également possible d'exprimer des incompatibilités entre des séquences. Nous proposons pour cela trois types de contraintes d'exclusion. Les premières (5.17), dénommées Exclusion1 (*Excl1.*), éliminent la succession de séquences non compatibles en indiquant qu'il est impossible d'avoir, sur une même ligne, une séquence s pendant la période t et une séquence s' pendant la période $t+1$ si s' n'appartient pas à l'ensemble des successeurs de s.

$$y_{ms}^{t} + y_{ms'}^{t+1} \leq 1 \qquad \forall t, \forall m, \forall s \in S(m), \forall s' \in S(m)/s' \notin Su(s) \tag{5.17}$$

Le deuxième type de contraintes d'incompatibilité (5.18), nommées Exclusion2 (*Excl2.*), interdit pour la période $t+1$ toutes les séquences n'appartenant pas à l'ensemble des successeurs de la séquence choisie pour la période t.

$$y_{ms}^{t} + \sum_{s' \notin SU(s)} y_{ms'}^{t+1} \leq 1 \qquad \forall t, \forall m, \forall s \in S(m) \tag{5.18}$$

Symétriquement, les contraintes Exclusion3 (*Excl3.*) éliminent le choix d'une séquence s pour la période t si la séquence choisie pour la période $t-1$ n'appartient pas à ses prédécesseurs.

$$y^t_{ms} + \sum_{s' \notin PR(s)} y^{t-1}_{ms'} \leq 1 \qquad \forall t, \forall m, \forall s \in S(m) \qquad (5.19)$$

Contraintes liant trois périodes

Finalement, le nombre de produits réalisables par une ligne par période étant limité à deux, il est possible de lier les périodes $t-1$, t et $t+1$ et d'exprimer que pour une séquence donnée pour la période $t-1$ et une séquence donnée pour la période $t+1$ une seule séquence est possible pour la période t (*3Per.*). Ceci est décrit par les contraintes (5.20).

$$y^{t-1}_{ms'} + y^{t+1}_{ms''} \leq 1 + y^t_{ms} \qquad \forall t, \forall m, \forall s' \in S(m), \forall s'' \in S(m), \quad \forall s \in S(m)/s \in PR(s'') \text{ et } s \in SU(s') \qquad (5.20)$$

Chacune de ces contraintes est suffisante, mais certaines d'entre elles peuvent être plus efficaces que d'autres (en terme de temps nécessaire à l'obtention d'une solution ou de taille de l'arbre parcourru). Ainsi, il est impératif de choisir le (ou les) type(s) de contraintes à utiliser. Pour cela, une partie de l'expérimentation de la section suivante est dédiée à la comparaison de l'efficacité de ces contraintes.

5.3.3 Fonction objectif multi-période

La fonction objectif globale vise, tout comme la fonction objective mono-période, à minimiser l'ensemble des coûts. pour cela, il faut additionner les coûts de chaque période et intégrer les frais de stockage. La fonction objectif multi-période est décrite par l'équation (5.11').

$$min \sum_t \Big(\sum_m \Big(\sum_i CP_{im} \times p^t_{im} + \sum_{s \in S(m)} CC_{sm} \times y^t_{ms} \Big) \qquad (5.13')$$

$$+ \sum_w \sum_i \Big(\sum_m CT_{mw} \times t^t_{imw} + \sum_c CT_{wc} \times q^t_{iwc} + CS_w \times s^t_{wi} \times V_i \Big) \Big)$$

Ainsi, le problème multi-période consiste à minimiser la nouvelle fonction objectif (5.12') sous les contraintes (5.1-5.12) mises à jour de façon à considérer les aspects multi-périodes, les contraintes (5.7-5.14) pour considérer les aspects de stockage et une ou plusieurs contraintes parmi (5.15-5.20) pour vérifier les contraintes d'adjacence entre séquences d'une même ligne.

5.4 Expérimentations

Afin de valider la modélisation, des exemples générés aléatoirement ont été résolus sur un Pentium 133MHz, en utilisant un programme écrit en C pour la génération

des problèmes et la mise en forme du programme linéaire, et CPLEX 5.0 pour la résolution. Nous avons ainsi résolus des exemples de différents types pour mettre en évidence l'influence de certains paramètres et les limites d'utilisation d'un tel modèle [38].

Pour des facilités d'écriture, nous nommons de façon abrégée un problème en donnant dans cet ordre, le nombre de lignes, de produits, de clients, de stocks et de périodes (M, N, C, W, T).
Notre plan d'expérience vise à étudier l'influence des éléments suivants :

- le choix des contraintes utilisées pour modéliser les contraintes d'adjacence des séquences,
- la densité du graphe de distribution,
- la taille des différentes composantes du système.

Cette partie, présente en premier lieu la taille du modèle, puis chacun des trois points d'expérimentation donnés ci-dessus.

5.4.1 Taille du modèle

Pour déterminer la taille du modèle, nous donnons le nombre de variables et de contraintes du modèle pour l'exemple de la division européenne de Pechiney.

5.4.1.1 Nombre de variables

Un premier décompte du nombre de variables montre que l'étage demandant le plus de variables est la distribution aux clients. Pour chaque période, 15 000 liens sont nécessaires pour relier les 50 stocks aux 300 clients. Ce qui donne 180 000 variables pour les 12 périodes. En fait, à cause des forts coûts de distribution, tous ces liens ne sont pas utilisés et il semble intéressant de ne considérer qu'une partie de ces liens (20%, 40%, ...), en ne conservant que les moins coûteux. Ainsi, pour 20%, le nombre de liens entre les stocks et les clients serait donc réduit à 36 000. Voici le nombre total de variables nécessaires.

p^t_{im} : 312 variables réelles $(= 12 \times (8 + 6 \times 2 + 2 \times 3))$
t^t_{imw} : 15 600 variables réelles $(= 12 \times (8 + 6 \times 2 + 2 \times 3) \times 50)$
q^t_{iwc} : 36 000 variables réelles $(= 12 \times 3000$, pour 20%$)$
s^t_{iw} : 9 600 variables réelles $(= 12 \times 16 \times 50)$

y^t_{ms} : 504 variables binaires $(= 12 \times (6 \times 4 + 2 \times 9))$

Total : 61 512 variables réelles et 504 variables binaires (0-1).

L'utilisation de variables réelles aux différents niveaux du modèle est justifiée par les quantités manipulées par le programme linéaire. En effet, les quantités

commandés sont en dizaines, voire centaines de milliers de produits et il semble naturel d'utiliser le millier comme unité.

5.4.1.2 Nombre de contraintes

Le décompte des contraintes est réalisé pour toutes les contraintes excepté les contraintes de non négativité et d'intégralité.

Contraintes (5.1)	:	192 contraintes	$(= 12 \times 16)$
Contraintes (5.2)	:	312 contraintes	$(= 12 \times (8 + 6 \times 2 + 2 \times 3))$
Contraintes (5.3)	:	192 contraintes	$(= 12 \times 16)$
Contraintes (5.6)	:	312 contraintes	$(= 12 \times (8 + 6 \times 2 + 2 \times 3))$
Contraintes (5.10)	:	4 800 contraintes	$(= 12 \times 50 \times 8)$
Contraintes (5.11)	:	3 600 contraintes	$(= 12 \times 300)$
Contraintes (5.7)	:	600 contraintes	$(= 12 \times 50)$

Total : 10 008 contraintes, sans compter les contraintes d'adjacence, dont au moins un type est nécessaire à la validité du modèle.

5.4.2 Choix des contraintes d'adjacence

Comme nous l'avons vu précédemment, plusieurs types de contraintes sont possibles pour modéliser les restrictions au niveau de la succession des séquences choisies pour chacune des lignes. Il est intéressant de rechercher quel est le groupe de contraintes à utiliser pour une résolution efficace.

Pour cela, nous avons mené deux expérimentations. La première compare chacun des types de contraintes pris séparément, tandis que la deuxième cherche à regrouper plusieurs types de contraintes dans l'objectif de réduire la taille de l'arbre de recherche et d'obtenir une solution plus rapidement.
Dans cette expérimentation, nous nous intéressons à plusieurs types de configurations de l'organe de production. Ainsi, pour un même ensemble (ou classe) de problèmes (M, N, W, C, T), le nombre de lignes de type $L1$, $L2$ et $L3$ peut légèrement varier d'un exemple à l'autre, ce qui nous permet d'observer, de façon plus générale, l'influence des contraintes d'adjacence utilisées.
Dans ce cadre, les comparaisons entre résultats sont à réaliser avec précaution. Il est possible de comparer pour un même ensemble de problèmes (comparaisons horizontales), l'efficacité des contraintes utilisées. Par contre, les comparaisons entre ensembles de problèmes (comparaisons verticales) n'ont pas de sens, car il se peut que chaque classe ne comporte pas les mêmes types de problèmes.

5.4.2.1 Comparaison de contraintes isolées

Pour différentes tailles de problèmes, nous relevons le temps nécessaire à la recherche de la solution optimale en fonction des contraintes utilisées pour modéliser

Problème (M, N, W, C, T)	Succ.	Prec.	Excl1.	Excl2.	Excl3.	3Per.
P1 : 4-4-5-15-12	69	**62**	201	93	94	200
P2 : 8-8-5-15-9	62	52	82	**45**	60	190
P3 : 12-4-5-15-12	24	**23**	45	25	25	111
P4 : 12-8-5-15-6	**55**	73	125	88	76	202

TAB. 5.3 - *Temps de recherche (sec) en fonction des contraintes utilisées*

les compatibilités entre les séquences de périodes adjacentes. Le tableau 5.3 reporte les moyennes des temps d'exécution pour 10 problèmes de chaque taille. Le meilleur temps de recherche est souligné pour chaque instance.

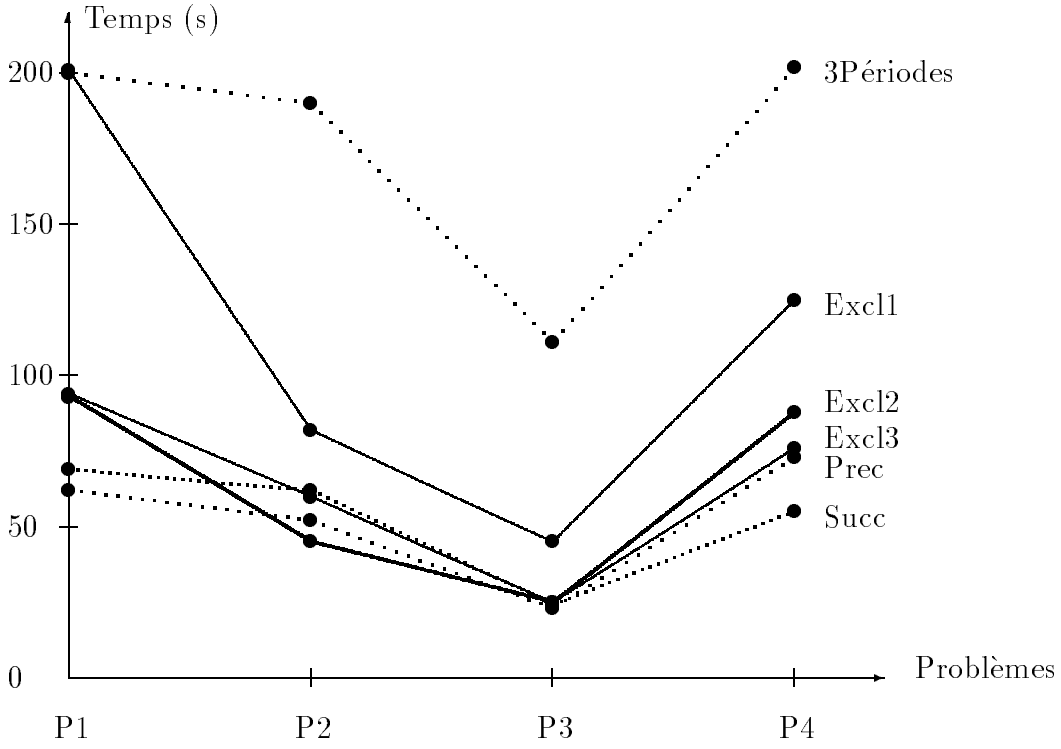


FIG. 5.4 - *Comparaison de l'efficacité des contraintes*

Comme nous pouvons le voir sur la figure 5.4, représentant le tableau 5.3, deux types de contraintes se font remarquer par leur non efficacité. Il s'agit des contraintes reliant trois périodes (3per.) et du premier type de contraintes d'exclusion (Excl1.). Ces deux types de contraintes ont la particularité de lier peu de variables entre elles. L'inconvénient est qu'il faut beaucoup de contraintes de ce type pour énu-

mérer toutes les possibilités, d'où une augmentation du temps nécessaire à la résolution. Les quatre autres types de contraintes (Succ., Prec., Excl2. et Excl3.) ont un comportement équivalent pour l'ensemble des problèmes. Nous choisirons, pour la suite des expérimentations, les contraintes de précédence comme référence, car elles semblent avoir un comportement assez régulier.

5.4.2.2 Comparaison de groupes de contraintes

Nous cherchons maintenant à combiner différents types de contraintes en vue d'augmenter l'efficacité de la modélisation. De même que précédemment, différentes expériences ont été menées pour des problèmes de tailles variées.

Problème (M, N, W, C, T)	6	5	4	P, S	P, E2	P, E3	P
P5: 8-8-5-15-9	230	162	164	151	159	146	**128**
P6: 12-4-5-15-12	188	102	100	59	97	69	**56**
P7: 16-6-5-15-6	163	105	105	**69**	82	75	71
P8: 10-6-5-15-12	401	334	174	**136**	199	140	142

TAB. 5.4 - *Temps de recherche (sec) en fonction du groupe de contraintes utilisé*

Le tableau 5.4 donne le temps nécessaire à la recherche de solutions en fonction du groupe de contraintes utilisé. La première colonne, notée 6, représente le cas où tous les types de contraintes sont utilisés (soit les six différents types). La deuxième colonne, indexée par 5, fait référence au cas où tous les types de contraintes, sauf 3-périodes, sont utilisés, quant à la colonne suivante, elle élimine également Exclusion1 du lot. Les colonnes suivantes combinent les contraintes deux à deux : Précédence et Succession, Précédence et Exclusion2, Précédence et Exclusion3. Quant à la dernière colonne, elle représente le point de référence car elle se rapporte à l'utilisation seule des contraintes de précédence.

La figure 5.5, représentant le tableau 5.4, montre qu'il est possible de distinguer trois groupes de courbes. Le premier groupe, composé des courbes 6 et 5, regroupe les cas les moins efficaces. Cela peut s'expliquer simplement par l'inefficacité intrinsèque des contraintes 3Per et Excl1. Le deuxième groupe, comportant 4 et P-E2, a un comportement assez homogène relativement efficace. Mais le groupe comportant les associations les plus efficaces, est composé de Prec, Prec-Succ et Prec-Excl3. Ainsi, les contraintes de précédence seules, sont très efficaces. Il ne semble pas utile de coupler différents types de contraintes en vue de diminuer le temps d'exécution. Nous remarquons tout de même que les couplages les plus performants sont ceux liants des contraintes symétriques, comme c'est le cas pour Prec-Succ et Prec-Excl3 qui associent Prec (relation entre $t-1$ et somme sur t)

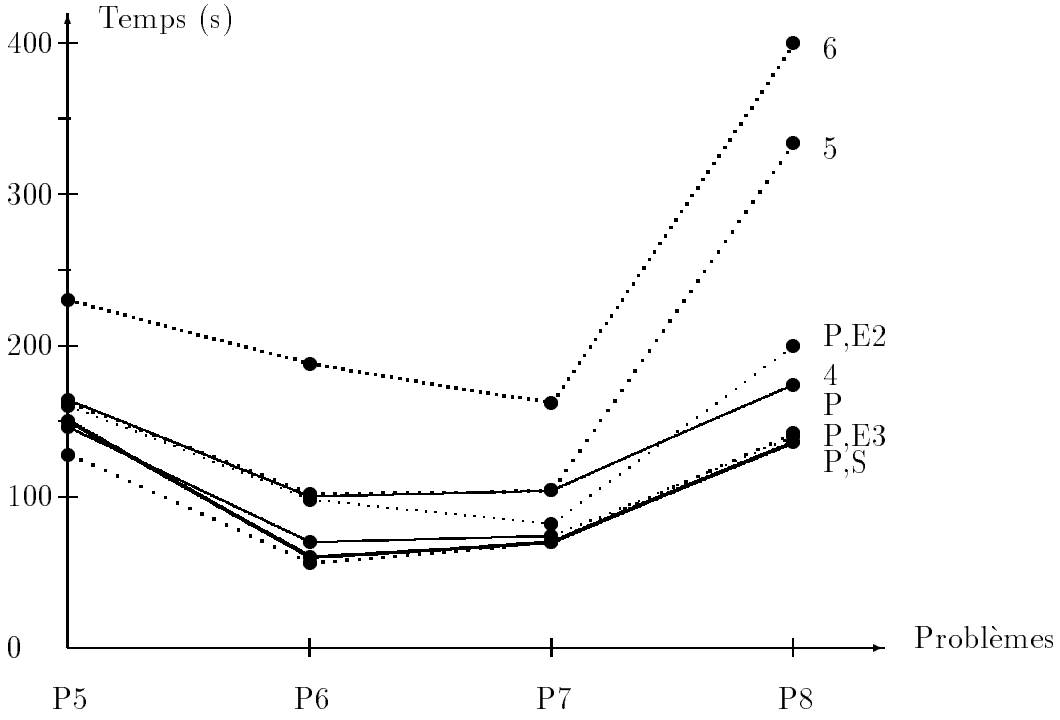


FIG. 5.5 - *Comparaison de l'efficacité des contraintes*

à Succ ou Excl3 (relation entre t et somme sur $t-1$.

Un autre critère pourrait entrer en ligne de compte pour le choix des contraintes à utiliser : celui de la taille de l'arbre de recherche, qu'il est parfois bon de réduire. Sur les exemples étudiés, nous avons relevé une légère diminution de la taille de l'arbre de recherche lorsque l'on couple, par exemple, Prec et Excl3 par rapport à l'utilisation seule de Prec. Mais cette diminution est assez faible (de l'ordre de 8%, en moyenne) et surtout, n'est pas systématique. Il ne semble donc pas que ce critère soit déterminant quant à l'utilisation de tel groupe de contraintes pour la résolution.

Ainsi, d'après ces conclusions et dans un souci de garder le programme linéaire le plus simple possible, nous choisissons d'utiliser, pour la suite de l'expérimentation les contraintes de précédence seules, non couplées à d'autres contraintes.

5.4.3 Densité du graphe de distribution

L'étude de l'influence de la densité du graphe de distribution nous parait intéressante car il semble qu'un graphe très peu dense limite très fortement la combinatoire en imposant très rapidement des choix. Au contraire, un graphe très dense

offre beaucoup de possibilités quant à l'acheminement des produits, et entraîne une combinatoire importante.

Pour étudier l'influence de la densité du graphe de distribution, nous avons résolu les mêmes problèmes, problèmes comportant un nombre de stocks et de clients suffisamment grand, avec des densités de distribution différentes. Tout comme pour l'expérimentation sur les contraintes d'adjacence, pour chaque classe de problèmes, le nombre de lignes de type $L1$, $L2$ et $L3$ peut varier légèrement d'un problème à l'autre. Les comparaisons entre classes de problèmes ne correspondent donc à rien. Seules les comparaisons au sein d'une classe, entre densités différentes, ont un sens ici. Une densité de 20 % correspond à ne considérer que les 20 % meilleurs liens pour chacun des clients (un client est relié aux 20 % stocks les plus proches). Pour ces exemples, nous avons relevé à la fois le temps nécessaire à l'obtention de la solution, pour les différentes densités et la valeur de la fonction objectif. En effet, il se peut, lorsque l'on restreint trop fortement le réseau de distribution, que l'on s'écarte de la solution optimale du problème. C'est ce que nous avons voulu étudier.

Le tableau 5.5 donne pour différents types de problèmes les moyennes pour 10 exécutions des temps de recherche de la solution, en fonction de la densité retenue pour le réseau de distribution utilisée.

Problème (M, N, W, C, T)	20 %	40 %	60 %	80 %	100 %
12-6-10-50-6	133	139	81	63	**33**
12-6-10-100-6	3893	1587	502	213	**36**
12-6-20-100-6	64	29	**28**	29	34
12-6-30-150-6	**66**	68	105	113	156
12-6-30-200-6	**32**	37	55	81	111
12-6-40-200-6	**38**	39	70	98	121

TAB. 5.5 - *Temps nécessaire (sec) à l'obtention de la solution optimale en fonction de la densité*

La figure 5.6, traduisant le tableau 5.5 à l'exception du cas $12-6-10-100-6$ qui sortirait du graphique, montre que l'intérêt de la limitation du réseau de distribution dépend fortement de la taille de ce réseau de distribution. Pour des réseaux trop petits, et en particulier, pour un petit nombre de stocks, la limitation du réseau de distribution augmente considérablement le temps nécessaire à la recherche de la solution optimale. Cela s'explique par la difficulté du problème engendrée par le peu de possibilités qu'il existe pour fournir un client. Par contre,

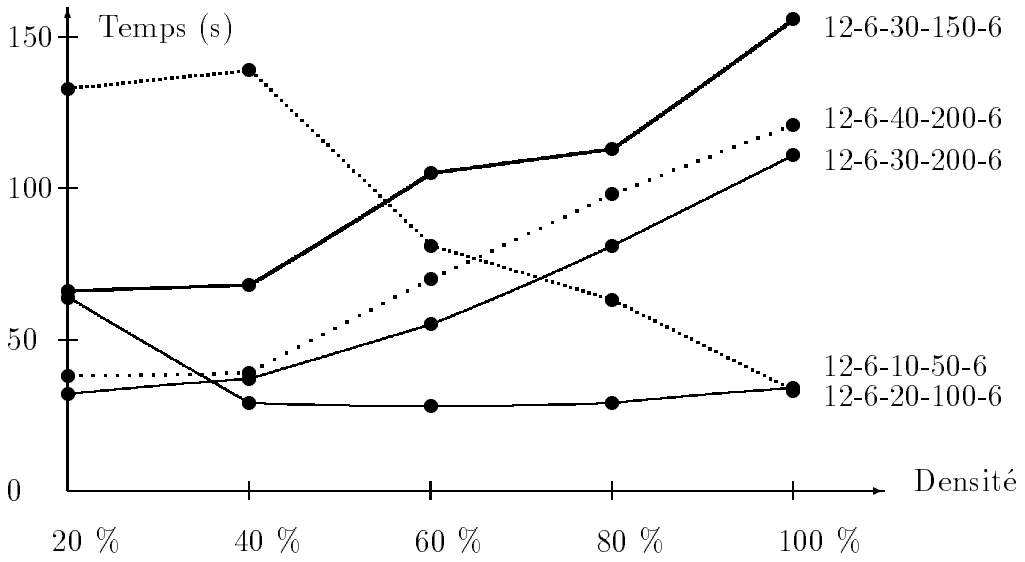


FIG. 5.6 - *Temps d'exécution en fonction de la densité*

dès que le nombre de stocks augmente (la limite semble se situer autour de 20 dans notre expérimentation), limiter la densité du réseau de distribution a des effets bénéfiques sur les temps de recherche.

Un autre critère important qu'il est nécessaire de surveiller est la valeur de la fonction objectif. Le tableau 5.7 donne, pour le même problèmes que dans le tableau 5.5, les moyennes des valeurs des fonctions objectifs.

Problème (M, N, W, C, T)	20 %	40 %	60 %	80 %	100 %
12-6-10-50-6	4.19	3.52	3.26	3.14	**3.06**
12-6-10-100-6	4.08	3.43	3.17	3.01	**2.91**
12-6-20-100-6	3.10	**2.79**	**2.79**	**2.79**	**2.79**
12-6-30-150-6	**2.58**	**2.58**	**2.58**	**2.58**	**2.58**
12-6-30-200-6	**2.67**	**2.67**	**2.67**	**2.67**	**2.67**
12-6-40-200-6	**2.85**	**2.85**	**2.85**	**2.85**	**2.85**

TAB. 5.6 - *Valeur des fonctions objectifs (10^7) en fonction de la densité*

Nous remarquons, d'après le tableau 5.6 et la figure 5.7 que pour les petits cas (peu de stocks), la solution trouvée avec un réseau de distribution incomplet n'est pas optimale. Par contre, pour les problèmes comportant un nombre élevé

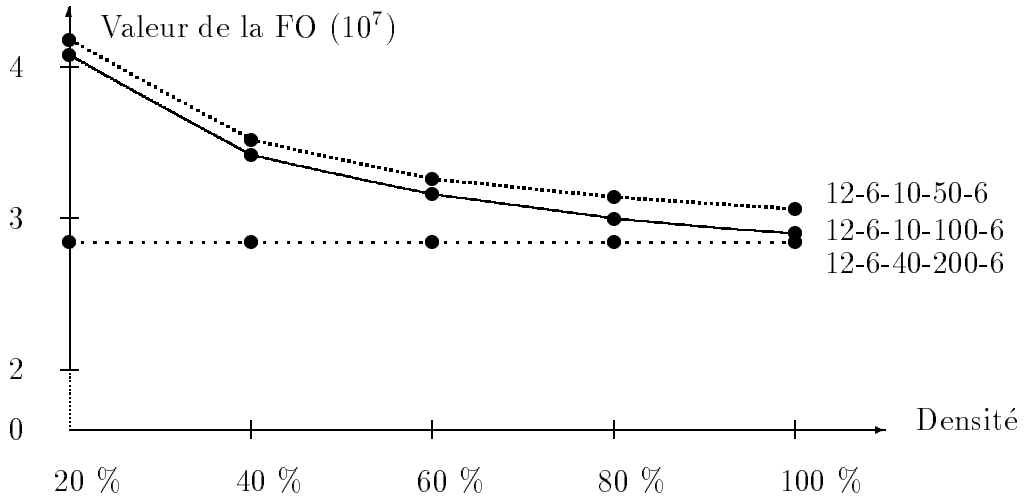


FIG. 5.7 - *Valeur de F.O (10^7) en fonction de la densité*

de stocks, la solution optimale est trouvée même lorsque l'on considère un réseau de distribution peu dense.
Nos conclusions, quant à l'intérêt de la réduction du réseau de distribution sont les suivantes : pour des problèmes avec peu de stocks (moins de 20 pour nos types de problèmes), il faut utiliser le réseau complet de façon à obtenir la solution optimale le plus rapidemment possible. Pour des problèmes plus gros (à partir de 20 stocks), nous conseillons d'utiliser un réseau à 40 % de manière à obtenir en un temps raisonnable et une solution qui a de grandes chances d'être optimale. C'est donc ce que nous ferons dans la suite de l'expérimentation.

5.4.4 Limites du modèle

Dans cette partie, nous essayons de déterminer les limites d'utilisation d'un tel modèle. Le critère limitant considéré est, une nouvelle fois, le temps mis pour l'obtention de la solution. En effet, c'est ce qui permet de discuter de la possibilité d'utilisation de ce modèle comme outil d'aide à la décision. Nous rappelons que nous utilisons les contraintes de précédence comme contraintes d'adjacence et une densité du réseau de distribution de 40%.

Pour mener cette expérience, nous avons résolu des problèmes de tailles variées et nous donnons ci-dessous les temps moyens d'exécution. Nous donnons également dans les colonnes qui suivent le pourcentage de problèmes pour lesquels la solution a été obtenue avant le temps limite indiqué (60s, 180s, 600s, 3600s), car pour une même classe de problèmes, le temps nécessaire à l'obtention de la solution optimale peut varier assez fortement (de 1 à 10).

Contrairement aux expérimentations précédentes, les configurations des exemples traités maintenant sont toutes identiques et sont similaires à la configuration rencontrée chez Pechiney, à savoir : 1 ligne sur 2 de type $L1$, 3 lignes sur 8 de type $L2$ et 1 ligne sur 8 de type $L3$. Ainsi, les comparaisons entre classes de problèmes sont possibles puisque chaque classe comporte des problèmes de même type de configuration. Le dernier problème étudié (noté Eur) est de la taille du problème Européen présenté au début du chapitre.

	Problème (M, N, W, C, T)	Temps	60s	180s	600s	1800s	3600s
P1	12-6-40-200-6	72 s	50%	100%	100%	100%	100%
P2	12-6-40-200-12	417 s	0%	0%	73,3%	100%	100%
P3	16-6-40-200-6	104 s	36,4%	90,9%	100%	100%	100%
P4	16-6-40-200-12	373 s	0%	6,3%	87,5%	100%	100%
P5	16-8-40-200-12	1704 s	0%	0%	28,6%	71,4%	78,6%
Eur	16-8-50-300-12	1392 s	0%	0%	35,3%	70,5%	94,1%

TAB. 5.7 - *Limites d'utilisation de la modélisation*

Une première constatation concerne l'influence du nombre de périodes. Nous pouvons remarquer, en comparant P1 avec P2 ou P3 avec P4 que doubler le nombre de périodes multiplie le temps de résolution par un facteur supérieur à 3. Cette augmentation est donc loin d'être linéaire.

De même, l'augmentation du nombre de produits entre P4 et P5 entraine une forte augmentation du temps d'exécution. Ceci s'explique, d'une part, par une forte augmentation du nombre de variables. D'autre part, à nombre de lignes égales, si un problème a plus de produits différents, cela signifie que pour chaque produit il existe moins de lignes candidates à sa fabrication. Le problème est donc plus difficile à satisfaire.

L'augmentation du nombre de lignes de production entre P1 et P3 ou P2 et P4 ne semble pas avoir de trop forte influence sur le temps d'exécution.

Finalement, la réduction du temps de résolution entre les problèmes P5 et Eur, peut être expliquée par l'augmentation des clients et la génération des données. En effet, de manière à générer des problèmes réalisables, la demande des clients est déterminée en fonction du temps global de production des lignes. Pour un même nombre de lignes et un nombre de clients plus élevé, la demande des clients sera diminuée. Ainsi, dans Eur, la demande de chaque client est légèrement réduite par rapport à P5, ce qui peut créer des problèmes un peu plus faciles à résoudre.

L'expérimentation ci-dessus montre également que pour des problèmes réels (de la taille de la division européenne), la résolution du programme linéaire associé à la modélisation prend, dans la plupart des cas, moins d'une heure.

5.5 Conclusion

La modélisation proposée pour l'étude de cas de la division européenne de boîtes de boisson a été rendue possible grâce au travail des chercheurs de chez Pechiney qui ont su poser le véritable problème et épurer les données en mettant en évidence les contraintes fortes du problème réel. Ainsi, l'aspect général de la problématique a été diminué et un problème réel, beaucoup moins complexe que le problème initial, a été étudié. Cependant, l'aspect générique de ce problème réel existe toujours, puisque les caractéristiques suivantes ont été considérées : production multisite multiproduit et multipériode avec des machines non liées, des temps de changement dépendant de la séquence et des demandes saisonnières. Ainsi, de très nombreux problèmes industriels pourraient être modélisés de la même façon, à condition qu'un même travail d'étude des données soit réalisé en amont.

L'expérimentation montre que des problèmes de taille réelle peuvent être résolus par la modélisation proposée dans des temps raisonnables. En effet, nous nous rendons compte que la plupart des problèmes sur 12 périodes, correspondants à une planification pour une année entière, sont résolus en moins d'une heure (ou environ une heure). Ce qui permet à cette modélisation de s'inclure dans un processus global d'aide à la décision, où le décideur peut, en faisant varier quelques paramètres (en particulier sur les commandes des clients), rechercher plusieurs solutions et choisir la plus intéressante à ses yeux. Cette solution n'est pas forcément celle de coût minimal, mais peut être une solution de coût légèrement plus élevé ayant d'autres caractéristiques intéressantes que seul le décideur peut évaluer.

Conclusion

Le travail exposé dans cette thèse s'intéresse à l'étude d'une problématique industrielle très générale concernant l'ordonnancement d'une entreprise multisite. Dans cette problématique, les aspects de distribution ont un impact aussi important que les aspects de production en terme de coûts.

Nous avons, au cours de ce mémoire, présenté rapidemment les problèmes d'ordonnancement et en particulier les problèmes d'ordonnancement sur machines parallèles, afin de montrer que certains de ces problèmes sont déjà difficiles pour des problématiques monosites. Puis nous avons étudié le problème global, réduit à une seule période en redéfinissant la notion de produit et en agrégeant les coûts de distribution et les coûts de production. Ce problème, qui peut être vu, une fois la modification réalisée, comme un problème monosite, se traduit par un problème d'ordonnancement sur machines parallèles non liées, avec temps de changement dépendant de la séquence. C'est un problème $\mathcal{NP}$-difficile, pour lequel, nous avons proposé des bornes, des méthodes exactes et des heuristiques de résolution. Il apparaît clairement, que sans hypothèses supplémentaires, ce problème est difficile à résoudre et que seules les heuristiques permettent d'appréhender les problèmes de tailles réelles.

Dans la deuxième partie du document, nous avons essayé d'étudier plus précisément les caractéristiques du problème de façon à tirer parti de la répartition géographique des sites de production. Nous avons proposé, dans un premier temps, une décomposition spatiale découplant le problème global en plusieurs problèmes, organisés de façon hiérarchique, dont les tailles permettent, la plupart du temps, l'utilisation de méthodes exactes. L'expérience nous montre que cette approche hiérarchique permet d'obtenir de meilleures solutions qu'une approche centralisée. Ainsi, la perte de précision induite par l'agrégation nécessaire à la décomposition est compensée par l'utilisation de méthodes plus efficaces.

Le dernier chapitre de la thèse s'intéresse à un cas particulier. Il nous montre

que lorsque l'on a une connaissance beaucoup plus précise du problème réel, il est possible de le résoudre beaucoup plus efficacement, voire optimalement. En effet, le travail d'épuration des données réalisé au sein de l'entreprise permet de se poser le problème réel et non un problème général, voire générique. En définissant plus précisément le problème, on obtient de meilleurs modèles, plus proches de la réalité et surtout de tailles plus petites. De meilleures résolutions sont alors possibles et dans notre cas, nous présentons une modélisation permettant une résolution optimale d'un problème de grande taille. Le problème de départ est pourtant de taille importante, puisqu'il s'agit de déterminer les plans de production, pour l'année à venir, des dix usines européennes de la société Pechiney.

Bien sûr la résolution proposée au chapitre 5 ne peut pas s'appliquer à tout type de problématique multiproduit, multisite et multipériode et reste l'étude d'un cas particulier.

Aussi, une extension directe du travail présenté dans cette thèse serait d'étendre les méthodes présentées au cours des chapitres 3 et 4 de façon à intégrer les aspects multipériodes. En effet, ces méthodes s'adressent à des problématiques plus générales que la résolution par programmation linéaire mixte du chapitre 5 et sont plus facilement transposables à d'autres problématiques. Etendre ces méthodes à des problématiques multipériodes permettrait d'anticiper les fluctuations de la demande, ce qui est très important pour des entreprises dont la demande est saisonnière.

Une deuxième perspective intéressante serait de prendre en compte plus finement l'aspect temporel. En effet, nous avons, dans ce mémoire, considéré que les délais étaient fixés par période, et n'avons pas considéré de délais au sein de la période. En fonction de la manière dont les commandes sont passées avec les clients, il peut être génant de ne considérer le temps que par période et ainsi de risquer de livrer en fin de période un client qui avait passé sa commande pour le début de période.

Enfin, une troisième perspective intéressante serait la prise en compte de l'environnement incertain dans lequel évoluent les entreprises. En effet, dans le monde réel, bien rares sont les situations pour lesquelles il est possible de déterminer de façon certaines toutes les données du problème. D'un point de vue externe, des perturbations peuvent arriver au niveau des commandes prévues, par exemple. En effet, les demandes sont directement liées aux prévisions de vente des clients, qui elles-mêmes ne peuvent pas être connues avec précision. Il est donc tout à fait possible que ces prévisions ne soient pas vérifiées et que le client revoit à la baisse ou à la hausse sa demande. Au niveau interne, des machines peuvent tomber en panne et les perturbations ainsi engendrées doivent être absorbées. Ainsi, à tout niveau et à tout moment, des perturbations risquent de venir géner

le bon déroulement du plan de production prévu, et il est nécessaire de remettre en cause un certain nombre de décisions. La question est de savoir quelles décisions remettre en cause. Jusqu'où la perturbation va-t-elle se propager? En ce qui concerne les perturbations internes, un premier élément de réponse a été donné au chapitre 4. Ces perturbations arrivent au niveau de la production et il semble intéressant de les traiter le plus localement possible. Pour cela, la structure hiérarchique proposée semble être appropriée puisqu'elle décompose l'ensemble de la division industrielle en sous-groupes. Chacun de ces groupes a une autonomie de décision qu'il est intéressant d'exploiter pour absorber les perturbations internes et locales. Pour les perturbations externes, si le traitement local n'est pas réalisable, il est nécessaire de revoir les décisions concernant l'ensemble du système de production, ce qui peut être très génant vis-à-vis des différentes usines. Il peut donc être utile de réfléchir à une méthode remettant en cause le moins de décisions possible, mais tout aussi efficace, qui utilisent la coopération entre régions par exemple.

Bibliographie

[1] Aguilera, L., Binder, Z., Hanada, F., and Guimares Ramos, J. Non identical parallel machines scheduling with sequence-dependent changeover costs in an industrial application. In *CESA'96 IMACS Multiconference, Computational Engineering in Systems Applications, Symposium on Discrete Events and Manufacturing Systems* (1996), Lille, pp. 559–564.

[2] Aguilera, L. M. *Ordonnancement de production avec coûts de changements dépendant de la séquence.* PhD thesis, L.A.G - I.N.P.Grenoble, 1993.

[3] Artigues, C. *Prise en compte des temps de préparation des ressources dans les problèmes d'ordonnancement d'atelier en temps réel.* PhD thesis, Université Paul Sabatier, Toulouse, 1997.

[4] Artigues, C., and Roubellat, F. An operation insertion procedure in a cumulative multi-resource schedule based on dominance rules. In *International Conference on Industrial Engineering and Production Management (IEPM97)* (1997), vol. 1, Lyon, pp. 304–371.

[5] Artigues, C., and Roubellat, F. Re-insertion principles for multi-resource shop scheduling with sequence-dependent setup times. In *Conference on Management and Control of Production and Logistics (MCPL97)* (1997), IFAC-IFIP, Campinas - Brasil, pp. 365–371.

[6] Assad, A., and Ball, M. Sleeping beauties : Scheduling the production of mattresses with sequence dependent set-ups. In *Symposium of Emerging Technology and factory Automation* (1995), vol. 3, INIRA/IEEE, pp. 229–239.

[7] Baker, K. *Introduction to Sequencing and Scheduling.* Wiley, New-York, 1974.

[8] Balas, E., and Toth, P. *The Traveling Salesman Problem.* E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan and D.B. Schmoys, Wiley - Chichester, 1985, ch. Branch and Bound methods, pp. 361–401.

[9] Barbarasoglu, G., and Özgür, D. A Lagrangean relaxation approach to an integrated production and 2-echelon distribution system. In *International Conference on Industrial Engineering and Production Management (IEPM'97), Lyon, France* (1997), vol. I, pp. 480–496.

[10] Bel, G., Perona, M., Sianesi, A., and Thierry, C. The multi-site production management problem : mathematical formalization and prototype developments. In *Tenth International Conference on Computer-Aided Production Engineering* (1994), Palermo, pp. 598–609.

[11] Bel, G., and Thierry, C. A constraint-based system for multi-site coordination and scheduling. In *Workshop on knowledge based production planning scheduling control* (1993), IJCAI 93, p. 10 pages.

[12] Bellman, R. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

[13] Belton, V., and Elder, M. Exploring a multicriteria approach to production scheduling. *Journal of the Operational Research Society 47* (1996), 162–174.

[14] Billaut, J., T'Kindt, V., Richard, P., and Proust, C. Three exact methods and an efficient heuristic for solving a bicriteria flowshop scheduling problem. In *International Conference on Computational Engineering on Systems Applications (CESA'98)* (1998), Hammamet (Tunisie), pp. 371–377.

[15] Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., and Weglarz, J. *Scheduling Computer and Manufacturing Processes.* Springer-Verlag, Berlin, Heidelberg, 1996.

[16] Bodin, L., and Golden, B. Classification in vehicle routing and scheduling. *Networks 11* (1981), 97–108.

[17] Boronad-Thierry, C. *Planification et ordonnancement multisite : une approche par satisfaction de contraintes.* PhD thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1994.

[18] Bourgade, V., Aguilera, L., Penz, B., and Binder, Z. Problème industriel d'ordonnancement bicritère sur machine unique : modélisation et aide à la décision. *APII 29*, 3 (1995), 331–341.

[19] Brucker, P. *Scheduling algorithms.* Springer-Verlag, Berlin, Heidelberg, 1995.

[20] Bruno, J., Coffman, E., and Sethi, R. Scheduling independent tasks to reduce mean finishing time. *Communications of the A.C.M. 17*, 7 (1974), 382–387.

[21] Burns, L., Hall, W., Blumenfeld, D., and Dazango, C. Distribution strategies that minimize transportation and inventory costs. *Operations Research 31* (1985), 361–380.

[22] Chandra, P., and Fisher, M. Coordination of production and distribution planning. *European Journal of Operational Research 72* (1994), 503–517.

[23] Cheng, T., and Chen, Z. Parallel machine scheduling with batch setup times. *Operations Research 24* (1995), 1171–1174.

[24] Cheng, T., and Sin, C. A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research 47* (1990), 271–292.

[25] Chevalier, A. *La programmation dynamique.* DUNOD, 1977.

[26] Chevalier, G., Barrier, J., and Richard, P. Production planning in the glass industry. In *Workshop on Production Planning and Control* (1996), Mons, Begium, pp. 282–285.

[27] Christofides, N. The vehicle routing problem. *R.A.I.R.O Recherche Opérationnelle 10*, 2 (1976), 55–70.

[28] Clark, G., and Wright, J. Scheduling vehicles from a central depot to a number of delivery points. *Operations Research 12* (1964), 568.

[29] Coffman, E., Garey, M., and Johnson, D. An application of bin-packing to multi-processor scheduling. *SIAM Journal of Computing 7* (1978), 1–16.

[30] Coffman, E., and Graham, R. Optimal scheduling for two processors systems. *Acta Informatica 1* (1972), 200–213.

[31] Cohen, M., and Lee, H. Strategic analysis of integrated production-distribution systems : Models and methods. *Operations Research 36* (1988), 216–228.

[32] Conway, R., Maxwell, W., and Miller, L. *Theory of Scheduling.* Addison Wesley, Massachussets, 1967.

[33] Cook, S. The complexity of theorem-proving procedures. In *Third ACM Symposium on Theory of computing* (1971), pp. 151–158.

[34] Dauzère-Peres, S. *Planification et ordonnancement de la production: une approche intégrée cohérente.* PhD thesis, Université Paul Sabatier, Toulouse, 1992.

[35] Davis, E., and Jaffe, J. Algorithms for scheduling tasks on unrelated processors. Report MIT-LCS-TM-137, MIT, 1979.

[36] De, P., Ghosh, J., and Wells, C. On the minimization of completion time variance with bicriteria extension. *Operations Research 40*, 6 (1992), 1148–1155.

[37] Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. Time constrained routing and scheduling. Tech. Rep. g-92-42, Les cahiers du GERAD, 1993.

[38] Dhaenens-Flipo, C., and G. Finke. An integrated model for an industrial production-distribution problem. Research Report 1006-I, LEIBNIZ-IMAG, 1998.

[39] Dietrich, B. A 2-phase heuristic for scheduling parallel unrelated machines with set-ups. Report RC 14330, IBM US Research Center, 1989.

[40] Dupont, L. *Gestion industrielle.* Hermès, Paris, 1998.

[41] Elmaghraby, S., Guinet, A., and Schellenberger, K. Scheduling jobs on unrelated parallel machines to minimze makespan. In *ORSA/ TIMS* (1992), San Fransisco.

[42] Emmons, H. A note on a scheduling problem with dual criteria. *Naval Res. Logis. Quaterly 22* (1975), 615–616.

[43] Finke, G., and Flipo, C. Integrated production-distribution model in manufacturing. *Management science and operations research in an emerging region - INFORMS* - Tel Aviv, 1998.

[44] Flipo, C. A fast heuristic for scheduling unrelated parallel machines with changeover times. *Workshop on Czech-French Cooperation in CIM Education and Research (WCFCIM96)* - Prague, 1996.

[45] Flipo, C. Management of a production and distribution system. *Workshop on Czech-French Cooperation in CIM Education and Research (WCFCIM96)* - Prague, 1996.

[46] Flipo, C. Scheduling unrelated parallel machines with changeover times. *Symposium of Combinatorial Optimization (co96)* - London, 1996.

[47] Flipo, C. A hierarchical scheme for a multi-facility production and distribution problem. In *International Conference on Industrial Engineering and Production Management (IEPM97)* (1997), vol. II, Lyon, pp. 396–406.

[48] Flipo, C. Scheduling unrelated parallel machines in the context of an industrial production-distribution problem. In *IFAC-IFIP Conference on Management and Control of Production and Logistics (MCPL97)* (1997), vol. I, Campinas - Brasil, pp. 249–254.

[49] Flipo, C. Spatial decomposition for a multi-facility production and distribution problem. *Journal of Intelligent Manufacturing, to appear* (1998).

[50] Flipo, C., and Finke, G. Modélisation intégrant production et distribution. *1er Congrès de la Société Française de Recherche Opérationnelle et Aide à la décision* - Paris, 1998.

[51] França, P., Gendreau, M., Laporte, G., and Müller, F. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics 43* (1996), 79–89.

[52] Gal, T. On efficient sets in vector maximum problems : A brief survey. *European Journal of Operational Research 24*, 2 (1986), 253–264.

[53] Garavelli, A., Okogbaa, O., and Violante, N. Global manufacturing systems : a model supported by genetic algorithms to optimize production planning. *Computers ind. Engng 31*, 1 (1996), 193–196.

[54] Garey, M., and Johnson, D. *Computers and Intractability : A Guide to the Theory of NP-Completeness.* Freeman and Company, New York, 1979.

[55] Gendreau, M., Hertz, A., and Laporte, G. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research 40*, 6 (1992), 1086–1094.

[56] Giard, V. *Gestion de production.* Economica, Paris, 1988.

[57] Glass, C. A., Potts, C., and Shade, P. Unrelated parallel machines scheduling using local search. *Mathl. Comput. Modelling 20*, 2 (1994), 41–52.

[58] Glover, F. Heuristic for integer programming using surrogate constraints. *Decision Sciences 8* (1977), 156–166.

[59] Glover, F. Future paths for integer programming and links to artificial intelligence. *Computers and O.R. 13*, 5 (1986), 533–549.

[60] Glover, F. Tabu search - part i. *ORSA Journal on Computing 1* (1989), 190–206.

[61] Glover, F. Tabu search - part ii. *ORSA Journal on Computing 2* (1990), 4–62.

[62] Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, Reading, USA, 1989.

[63] Golden, B., and Stewart, W. *The Traveling Salesman Problem.* E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan and D.B. Schmoys, Wiley - Chichester, 1985, ch. Empirical analysis of heuristics, pp. 207–249.

[64] Gonthier, A. *De l'optimisation des Coûts de Changement de Fabrication vers l'Atelier Logiciel Décentralisé d'Ordonnancement.* PhD thesis, LAG, Institut National Polytechnique Grenoble, 1990.

[65] GOTHa. Les problèmes d'ordonnancement. *R.A.I.R.O. - O.R. 27*, 1 (1993), 77–150.

[66] Graham, R. Bounds for certain multiprocessing anomalies. *Bell Systems Tech. J.* (1966), 263–269.

[67] Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. Optimization and approximation in deterministic sequencing and scheduling theory : a survey. *Annals of Discrete Mathematics 5* (1979), 287–326.

[68] Guinet, A. Textile production systems : a succession of non-identical parallel processor shops. *Journal of Operational Research Society 42*, 8 (1991), 655–671.

[69] Guinet, A. Scheduling sequence dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of production Research 31*, 7 (1993), 1579–1594.

[70] Guinet, A., Echalier, F., and Dussauchoy, A. Scheduling jobs on parallel machines : a survey. In *Joined International Conference on Operational Research and Management Science* (1992), EURO XII, TIMS 31, Helsinki, Finland.

[71] Hoffman, A., and Wolfe, P. *The Traveling Salesman Problem.* E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan and D.B. Schmoys, Wiley - Chichester, 1985, ch. History, pp. 1–15.

[72] Hoogeveen, J., and van de Velde, S. A new lower bound approach for single-machine multicriteria scheduling. *Operations Research Letters 11* (1992), 39–44.

[73] Hoogeveen, J., and van de Velde, S. Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. *Operations Research Letters 17* (1995), 205–208.

[74] Horn, W. Minimizing average flow time with parallel machines. *Operations Research 21* (1973), 846–847.

[75] Horowitz, E., and Sahni, S. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of Assoc. Comput. Machinery 23* (1976), 317–327.

[76] Horvath, E., Lam, S., and Sethi, R. A level algorithm for preemptive scheduling. *J. Assoc. Comput. Mach. 25* (1977), 92–101.

[77] Hu, T. Parallel sequencing and assembly line problems. *Operations Research 9* (1961), 841–848.

[78] Ibarra, O., and Kim, C. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. Assoc. Comput. Mach. 24*, 2 (1977), 280–289.

[79] ILOG. *Using the CPLEX 5.0 Callable Library.* ILOG Inc. CPLEX Division, Incline Village, 1997.

[80] Johnson, S. Optimal two- and three-stage production schedules with setup times included. *Naval Research logistic Quaterly 1* (1954), 61–68.

[81] Karp, R. *Complexity of computer Computations.* R.E. Miller, J.W. Thatcher (eds.), Plenum Press, New-York, 1972.

[82] Kedad, S. *Résolution de problèmes de partitionnement généralisé par des méthodes d'optimisation globale à base de déplacements stochastiques : application à l'ordonnancement à machines parallèles.* PhD thesis, Ecole Centrale de Paris, 1997.

[83] Kedad, S., Lecomte, C., and Dejax, P. Une heuristique en deux phases pour la résolution d'un problème d'ordonnancement à machines parallèles. In *GI5 - 5e congrès international de Génie industriel* (1996), vol. 1, pp. 97–103.

[84] Kirkpatrick, S., Gelatt, C., and Vecchi, M. Optimization by simulated annealing. *Science 220* (1983), 671–680.

[85] Labetoulle, J., Lawler, E., Lenstra, J., and Rinnooy Kan, A. *Progress in Combinatorial Optimization.* W.R. Pulley-blank (eds.), Academic press, New York, 1984, ch. Preemptive Scheduling of uniform processors subject to release dates, pp. 245–261.

[86] Laporte, G. The vehicle routing problem : An overview of exact and approximate algorithms. *European Journal of Operational Research 59* (1992), 345–358.

[87] Laporte, G., and Nobert, Y. Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics 31* (1987), 147–184.

[88] Laporte, G., and Osman, I. Routing problems : A bibliography. *Annals of Operations Research 61* (1995), 227–262.

[89] Lasserre, J. An integrated model for job-shop planning and scheduling. *Management Science 38*, 8 (1992), 1201–1211.

[90] Lawler, E. Optimal sequencing of a single machine subject to precedence constraints. *Management Science 19* (1973), 544–546.

[91] Lawler, E., and Labetoulle, J. Preemptive scheduling of unrelated parallel processors by linear programming. *J. Assoc. Comput. Mach. 25* (1978), 612–619.

[92] Lawler, E., Lenstra, J., Rinnooy Kan, A., and Schmoys, D. B. *Logistics of production and inventory - Handbooks in OR and MS.* S.C. Graves et al. (eds.), Elsevier Science Publishers, 1993, ch. Sequencing and scheduling : algorithms and complexity, pp. 445–522.

[93] Lawler, E., Lenstra, J., Rinnooy Kan, A., and Shmoys, D. *The Traveling Salesman Problem, a Guided Tour of Combinatorial Optimization.* Wiley, New York, 1985.

[94] Lenstra, J., and Rinnoy Kan, A. Complexity of scheduling under precedence constraints. *Operations Research 26* (1978), 22–35.

[95] Lenstra, J., Schmoys, D., and Tardos, E. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming 46* (1990), 259–271.

[96] Lin, S. Computer solutions of the travelling salesman problem. *Bell System Technical Journal 44* (1965), 2245–2269.

[97] Little, J., Murty, K., Sweeney, D., and Karel. An algoritm for the travelling salesman problem. *Operations Research 11* (1963), 972–989.

[98] Margot, F. Quick updates for p-opt tsp heuristics. *Operations Research 11* (1992), 45–46.

[99] Martello, S., Soumis, F., and Toth, P. Exact and approximation algorithms for makespan minimization on unrelated machines. *Discrete Applied Mathematics 75* (1997), 169–188.

[100] Mc Naughton, R. Scheduling with deadlines and loss functions. *Management Science 6* (1959), 1–12.

[101] Metters, R. Interdependent transportation and production activity at the United States postal services. *Journal of Operational Research Society 47* (1996), 27–37.

[102] Mittenthal, J., Raghavachari, M., and Rana, A. A class of single machine central tendency-dispersion bicriteria problems. *Journal of the Operational Research Society 47* (1996), 1355–1365.

[103] Muntz, R., and Coffman, E. Preemptive scheduling or real time tasks on multiprocessor systems. *J. Assoc. Comput. Mach 17* (1970), 324–338.

[104] Nagar, A., Haddock, J., and Heragu, S. Multiple and bicriteria scheduling : a literature review. *European Journal of Operational Research 81* (1995), 88–104.

[105] Nelson, R., Sarin, R., and Daniels, R. Scheduling with multiple performance measures : the one machine case. *Management Science 32*, 4 (1986), 464–479.

[106] Ovacik, I. M., and Uzsoy, R. Worst-case error bounds for parallel machine scheduling problems with bounded sequence-dependent setup times. *Operations Research Letters 14* (1993), 251–256.

[107] Panwalkar, S., Dudek, R., and Smith, M. *Symposium on the theory of Scheduling and its application.* Springer, New-York, 1973, ch. Sequencing research and the industrial scheduling problem.

[108] Panwalkar, S., and Iskander, W. A survey of scheduling rules. *Operations Research 25* (1977), 45–61.

[109] Pareto, V. *Cours d'économie Politique.* Rouge, Lausanne, 1896.

[110] Parker, R., Deane, R., and Holmes, R. On the use of a vehicle routing algorithm for thz parallel processor problem with sequence dependent changeover costs. *AIIE Transactions 9* (1977), 155–160.

[111] Perraud-Echalier, F. *Problèmes d'ordonnancement sur machines parallèles : Apport du recuit simulé.* PhD thesis, Université Claude Bernard, Lyon I, 1991.

[112] Pinedo, M. *Scheduling : theory, algorithms and systems.* Prentice Hall Series, New Jersey, 1995.

[113] PORTMANN, M. *Méthodes de décomposition spatiale et temporelle en ordonnancement de la production.* PhD thesis, Thèse d'état, Université de Nancy I, 1987.

[114] POTTS, C. N. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics 40* (1985), 155–164.

[115] PROUST, C. *Un système d'aide à la décision pour une élaboration rationnelle et intéractive de calendrier de production.* PhD thesis, Université Claude Bernard (Lyon I), 1977.

[116] PROUST, C., AND GRÜNENBERGER, E. Planification de production dans un contexte de "flowshop" hybride: conception et interprogrammation d'ariane 2000. *Revue d'Automatique et de Productique Appliquée (RAPA) 8* (1995), 715–734.

[117] PYKE, D., AND COHEN, M. Multiproduct integrated production-distribution systems. *European Journal of Operational Research 74* (1994), 18–49.

[118] RAJENDRAN, C. Two-stage flowshop scheduling problem with bicriteria. *Journal of the Operational Research Society 43*, 9 (1992), 871–884.

[119] RAJGOPAL, J., AND BIDANDA, B. On scheduling parallel machines with two setup classes. *International Journal of production Research 29*, 12 (1991), 2443–2458.

[120] RICHARD, P. *Contribution des réseaux de Petri à l'étude de problèmes de recherche opérationnelle.* PhD thesis, Université françois rabelais, Tours, 1997.

[121] RICHARD, P., BARRIER, J., AND PROUST, C. Maximizing production margin in short-term planning in the glass industry. In *International Conference on Industrial Engineering and Production Management (IEPM'97)* (1997), vol. II, Lyon, France, pp. 22–32.

[122] RICHARD, P., BARRIER, J., AND PROUST, C. Planification court terme à la meilleure marge dans l'industrie verrière. Rapport interne 184, EIII, Laboratoire d'Informatique, Tours, 1997.

[123] RINNOY KAN, A. *Machine scheduling problems: classification, complexity and computations.* Nijhoff, The Hague, 1976.

[124] ROCHAT, Y. *Modélisation et résolution de problèmes de distribution et d'ordonnancement.* PhD thesis, Ecole Polytechnique Fédérale de lausanne (Suisse), 1996.

[125] ROTHKOPF, M. Scheduling independent tasks on parallel processors. *Management Science 12* (1966), 347–447.

[126] ROY, B., AND BOUYSSOU, D. *Aide Multicritère à la Décision : Méthodes et Cas*. Economica, Paris, 1993.

[127] ROY, B., AND VINCKE, P. Multicriteria analysis : survey and new directions. *European Journal of Operational Research 8* (1981), 207–218.

[128] SAKAROVITCH, M. *Optimisation Combinatoire : Programmation discrete*. Hermann, Paris, 1984.

[129] SCHÄRLIG, A. *Décider sur plusieurs critères : Panorama de l'aide à la décision multicritère*. Presse Polytechnique Romandes, Lausanne, 1985.

[130] SO, K. C. Some heuristics for scheduling jobs on parallel machines with setups. *Management Science 36*, 4 (1990), 467–475.

[131] SOLOMON, M. Algorithms for the vehicle routing and scheduling problems with time windows constraints. *Operations Research 35*, 2 (1987), 254–265.

[132] SUMICHRAST, R., AND BAKER, J. Scheduling parallel processors: An integer linear programming based heuristic for minimizing setup time. *International Journal of Production Research 25*, 5 (1987), 761–771.

[133] TANG, C. Scheduling batches on parallel machines with major and minor set-ups. *European Journal of Operational Research 46* (1990), 28–37.

[134] THIERRY, C., BEL, G., AND ESQUIROL, P. A constraint based model for multi-site scheduling. In *12th IFAC World Congress* (1993), Sydney, Australia, p. 4 pages.

[135] THIERRY, C., BEL, G., AND ESQUIROL, P. Multi-site scheduling : a constraint based approach. In *International Conference on Industrial Engineering and production management (IEPM'93)* (1993), Fucam Mons, Belgique, p. 10 pages.

[136] T'KINDT, V., AND BILLAUT, J. Les problèmes d'ordonnancement d'atelier multicritères. Rapport Interne 206, LI/ E3I/ Tours, 1998.

[137] T'KINDT, V., BILLAUT, J., AND PROUST, C. Scheduling jobs on unrelated parallel machines : resolution of an industrial problem. In *Sixth international workshop on project management and scheduling* (1998), Istanbul (Turkey), pp. 301–304.

[138] T'Kindt, V., Richard, P., Proust, C., and Billaut, J. Resolution of a 2-machine bicriteria flowshop scheduling problem. In *International Conference on Methods and Applications of Multicriteria Decision Making (MAMDM'97)* (1997), Mons (Belgique), pp. 139–143.

[139] Ullman, J. *Scheduling in Computer and JobShop Systems*. E.G. Coffman Jr (eds.), J. Wiley, New-York, 1976, ch. Complexity of Sequencing problems.

[140] Van Wassenhove, L., and Baker, K. A bicriterion approach to time/cost trade-offs in sequencing. *European Journal of Operational Research 11* (1982), 48–54.

[141] Van Wassenhove, L., and Gelders, L. Solving a bicriterion scheduling problem. *European Journal of Operational Research 4* (1980), 42–48.

[142] Williams, D., and Wirth, A. A new heuristic for a single machine scheduling problem with set-ups times. *Journal of the Operational Research Society 47* (1996), 175–180.

[143] Williams, J. Heuristic techniques for simultaneous scheduling of production and distribution in multi-echelon structures : theory and empirical comparisons. *Management Science 27* (1981), 336–351.

[144] Xuong, N. *Mathématiques discrètes et informatique*. Masson, Paris, 1992.

Index

Notations

Lorsque les aspects multipériode sont considérés, l'exposant ^{t} indique que la donnée (ou la variable) correspond à la période t. Par exemple, B_m^t indique la capacité de la machine m pour la période t.

Entités et indices

c	Indice des clients
$C(i)$	ensemble de clients ayant commandé du produit i
F	Ensemble des fabriques ($= \{1 \ldots \lvert F \rvert\}$)
F_r	Ensemble de fabriques (usines) de la région r $(1 \ldots \lvert F_r \rvert)$
i	Indice de tâches
J	Ensemble des tâches à réaliser ($= \{1 \ldots \lvert J \rvert\}$)
j	Indice de tâches
J_f	Ensemble des tâches à réaliser par l'usine f $(1 \ldots \lvert J_f \rvert)$
J_r	Ensemble de tâches à réaliser par la région r
k	Indice de processeurs
l	Indice relatif aux positions d'un ordonnancement
L_m	Ensemble des position valides pour le processeur m
M	Ensemble des processeurs
m	Nombre de processeurs
M_f	Ensemble des machines de l'usine f $(1 \ldots \lvert M_f \rvert)$
M_j	Ensemble des processeurs potentiels pour l'exécution de la tâche j
N	Ensemble des tâches
n	Nombre de tâches
P	Ensemble des processeurs
$PR(s)$	Ensemble des prédécesseurs compatibles avec la séquence s
s	Indice des séquences
S_m	Ensemble des séquences admissibles pour m
$SU(s)$	Ensemble des successeurs compatibles avec la séquence s
w	Indice des stocks

Données

B_m	Capacité de la machine m
B_f	Capacité de production de l'usine f
B_r	Capacité de production de la région r
Ca_{mc}	Coût d'acheminement de m à c
Cc_{ijf}	Coûts de changement de la tâche i à la tâche j dans l'usine f
Cc_{ijm}	Coût de changement de la tâche i à la tâche j sur la machine m
Cc_{sm}	Coût de changement relatif à la séquence s sur la machine m
Cf_{im}	Coût fixe relatif à la fabrication de i sur m
Ch_m	Coût horaire de la machine m
Cl_{sc}	Coût de livraison de s à c
Cp_{if}	Coût de production de la tâche i dans l'usine f
Cp_{im}	Coût de production de la tâche i sur la machine m
Cs_{is}	Coût de stockage de i dans le stock s
Ct_{if}	Coût de transport de la tâche i depuis l'usine f
Ct_{im}	Coût de distribution de la tâche i depuis la ligne m
Ct_{ms}	Coût de transport de m à s
D_{ic}	Demande en produit i du client c
D_{il}	Demande des clients
D_{is}	Prévisions de stockage (demande artificielle)
d_j	Date d'échéance souhaitée de la tâche j
$\tilde{d}_j$	Date d'échéance impérative de la tâche j
K_s	Capacité de stockage du stock s
K_w	Capacité de stockage du stock w
p_{im}	Temps d'exécution de la tâche i sur le processeur k
r_j	Date de disponibilité de la tâche j
s_{ijk}	Temps de changement entre la tâche i et la tâche j sur le processeur k
T	Longueur de la période
Tc_{ijf}	Temps de changement de la tâche i à la tâche j dans l'usine f.
Tc_{ijm}	Temps de changement de la tâche i à la tâche j sur la machine m
Tc_{sm}	Temps de changement relatif à la séquence s sur la machine m
Tp_{if}	Temps de production de la tâche i dans l'usine f
Tp_{im}	Temps de production de la tâche i sur la machine m
V_i	Volume du produit i
w_j	Poids de la tâche j

Variables et Résultats

σ	Ordonnancement réalisable
α	Poids donné au critère temps
a_{imc}	Quantité de i acheminée directement de m à c
c_j	Date de fin d'exécution de la tâche j
C_{max}	Date de fin d'exécution de la dernière tâche ($C_{max} = max_j C_j$)
CA_{im}	Coût d'affectation de la tâche i sur le processeur m
f_j	Durée du flot de la tâche j
F	Durée du flot totale ($F = \sum_j f_j$)
l_{imc}	Quantité de i livrée depuis s jusqu'à c
l_j	Retard algébrique de la tâche j ($l_j = c_j - d_j$)
L_{max}	Retard algébrique maximal $L_{max} = max_j l_j$
$L(m)$	Nombre maximal de positions valides pour le processeur m
O_m	Ordonnancement de la ligne m
p_{im}	Quantité de produit i fabriquée sur la ligne m
q_{im}	Quantité de i fabriquée sur m
q_{wc}	Quantité de produit transportée depuis le stock w jusqu'au client c
s_{is}	Quantité de i dans le stock s
s_{iw}	Quantité de produit i traversant le stock w
TA_{im}	Temps d'affectation de la tâche i sur le processeur m
t_{ims}	Quantité de i transportée depuis m jusqu'à s
t_{imw}	Quantité de produit i transportée depuis la ligne m jusqu'au stock w
t_j	Retard de la tâche j ($t_j = max_j(0, l_j)$)
T_{max}	Retard maximal $T_{max} = max_j t_j$
x_{ijm}	$= \begin{cases} 1 & \text{Si la tâche j suit la tâche i sur la machine m} \\ 0 & \text{Sinon} \end{cases}$
y_{ilm}	$= \begin{cases} 1 & \text{Si la tâche i est exécutée sur m en } l^e \text{ position} \\ 0 & \text{Sinon} \end{cases}$
y_{ms}	$= \begin{cases} 1 & \text{Si la séquence } s \text{ est choisie pour la ligne } m \\ 0 & \text{Sinon} \end{cases}$