ORDONNANCEMENT EN TEMPS REEL DANS LES PROBLEMES A EN-COURS LIMITES

Fabrice Chauvet

N° d'ordre : Année 1999

THÈSE

présentée à

L'UNIVERSITÉ DE METZ

FACULTÉ DES SCIENCES

UFR MATHÉMATIQUES, INFORMATIQUE, MÉCANIQUE

pour obtenir le titre de

DOCTEUR

Spécialité : Mathématiques

Mention : Mathématiques Appliquées et Informatique

par

Fabrice CHAUVET

Sujet de la thèse

ORDONNANCEMENT EN TEMPS RÉEL DANS LES PROBLÈMES À EN-COURS LIMITÉS

Soutenue le lundi 20 septembre 1999 devant le jury composé de :

Président : **M. Edouard WAGNEUR**, Professeur à l'Ecole des Mines de Nantes

Rapporteurs : **M. Chengbin CHU**, Professeur à l'Université de Troyes

M. Gerd FINKE, Professeur à l'Université de Grenoble

Examinateurs : **M. Jean-François CLAVER** du Centre de Recherche de Péchiney

M. Ahmedou Ould HAOUBA, Professeur à l'Université de Nouakchott

M. Christian PRINS, Professeur à l'Université de Troyes

Directeur de thèse : **M. Jean-Marie PROTH**, Directeur de recherche à l'INRIA, Metz

Cette thèse a été préparée à l'INRIA (Institut National de Recherche en Informatique et en Automatique) de septembre 1996 à septembre 1999 et co-financée par la Région Lorraine.

REMERCIEMENTS

C'est à Monsieur **Jean-Marie Proth**, Directeur de Recherche à l'Institut National de Recherche en Informatique et en Automatique, que reviennent mes plus vifs remerciements. Il n'a cessé de m'encourager depuis le début de mes travaux de recherche à l'INRIA. Je suis particulièrement honoré de sa présence dans mon jury.

Je veux aussi remercier le Professeur **Chengbin CHU**, Professeur à l'Université Technologique de Troyes et le Professeur **Gerd FINKE**, Professeur à l'Université de Grenoble, de me faire l'honneur de participer à mon jury et d'avoir accepté les responsabilités de rapporteurs.

Je désire exprimer ma profonde gratitude à Monsieur **Jean-François CLAVER**, Responsable de l'Unité Statistique et Productique Industrielle au Centre de Recherche de Péchiney, le Professeur **Ahmedou Ould HAOUBA**, Professeur et Doyen de l'Université de Nouakchott, le Professeur **Christian PRINS**, Professeur à l'Université Technologique de Troyes et le Professeur **Edouard WAGNEUR**, Professeur à l'Ecole des Mines de Nantes, d'avoir accepté de participer à ce jury.

Je remercie également tous ceux qui m'ont incité à m'engager dans la recherche en algorithmique et en optimisation combinatoire.

Je veux aussi souligner le soutien de ma femme et de ma famille. Leurs encouragements ont été essentiels dans le succès de ce travail.

Enfin, que les personnes qui m'ont accompagné tout au long de cette thèse - mes amis, les membres de l'INRIA et les nombreux chercheurs avec qui j'ai eu la chance de travailler - reçoivent toute ma reconnaissance et ma sympathie.

A mes parents

A Laurence

TABLE DES MATIÈRES

Chapitre II - Problèmes à gammes linéaires.....23

Chapitre III - Problèmes avec gammes d'assemblage et de désassemblage 103

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

INTRODUCTION

Cette thèse présente des travaux qui concernent **l'ordonnancement en "temps réel"**. L'ordonnancement fait partie des décisions opérationnelles permettant de gérer la production. Il consiste à définir les dates d'exécution des opérations à réaliser en tenant compte des disponibilités des ressources qui leur sont nécessaires. Nous voulons ici déterminer les dates d'exécution des opérations à réaliser sur chaque produit, dès que sa commande arrive, sans connaître les commandes futures et sans remettre en question les ordonnancements antérieurs. Nous dirons alors que le système de production est géré en temps réel. Cette pratique est très courante dans les entreprises. Elle permet de prendre les décisions rapidement et de réagir dynamiquement aux aléas de l'environnement.

Les problèmes d'ordonnancement auxquels nous nous intéressons ont les deux caractéristiques suivantes :

a) ils sont de type "**sans attente**", c'est-à-dire que les opérations exécutées sur un même produit s'enchaînent sans interruption,

b) ils sont "**à temps opératoires contrôlables**", ce qui signifie qu'il est possible de choisir le temps opératoire dans un intervalle donné.

Ces deux conditions sont en apparence restrictives. En fait, nous montrons dans le premier chapitre qu'elles permettent une gestion de la plupart des systèmes de fabrication : ateliers de traitements de surface, entreprises manufacturières, usines automatisées... De plus, ces deux conditions offrent de nouvelles possibilités : elles permettent de contrôler la **quantité des produits semi-finis et leur durée de séjour en cours de production**.

Le premier chapitre présente l'état de l'art des problèmes d'ordonnancement de type "sans attente". L'originalité de cette bibliographie réside dans la classification des problèmes. En effet, tous les problèmes d'ordonnancement (y compris ceux dans lesquels l'attente entre les opérations est permise) sont décrits comme des problèmes sans attente avec une contrôlabilité variable des durées opératoires. Ce chapitre introduit également l'éventail des domaines d'application des problèmes traités dans les trois chapitres qui le suivent.

Dans les Chapitres II à IV, nous proposons des méthodes de gestion en temps réel applicables aux systèmes de fabrication dans lesquels les produits subissent une suite d'opérations (voir Chapitre II - Problèmes à gammes linéaires), dans lesquels les produits nécessitent une opération d'assemblage ou de désassemblage (voir Chapitre III - Problèmes à gammes d'assemblage et désassemblage), dans lesquels plusieurs gammes différentes peuvent être choisies pour réaliser un produit (voir Chapitre IV - Problèmes à gammes alternatives).

Pour chaque situation, nous définissons précisément le cadre de l'étude avant de formaliser sous forme de programme mathématique toutes les contraintes à prendre en compte au cours de la gestion du système. Les méthodes de gestion proposées permettent d'atteindre trois objectifs :

a) **minimiser la date d'achèvement** du produit considéré. Ce critère est classique en ordonnancement et correspond à la minimisation du *makespan.* Nous donnons des algorithmes originaux et efficaces pour atteindre cet objectif. Ils sont de complexité polynomiale pour les produits à gammes linéaires, d'assemblage et de désassemblage. Dans le cas de produits à gammes alternatives, nous montrons que minimiser le *makespan* devient "NP-difficile" et nous donnons un algorithme pseudo polynomial.

b) **minimiser le temps d'utilisation** des ressources, tout en respectant le délai de livraison. Ce critère est utilisé en gestion de projet pour retarder le plus possible le début des travaux. Il s'oppose à l'objectif précédent. Les algorithmes que nous proposons pour minimiser le temps d'utilisation sont de même complexité que ceux qui sont développés pour minimiser la date d'achèvement du produit.

c) **minimiser le coût d'utilisation** des ressources. Cet objectif est tout à fait nouveau dans le domaine de l'ordonnancement en temps réel. Mais, atteindre cet objectif est également plus difficile. Nous donnons pour cela des algorithmes pseudo polynomiaux pour chaque situation.

Enfin, nous donnons des résultats sur le comportement de ces méthodes en "temps réel" par rapport au contexte *off-line* où les commandes futures sont connues au moment d'ordonnancer un produit.

L'Annexe I suit les 161 références bibliographiques. Elle complète la thèse en proposant des algorithmes de mise à jour des périodes de disponibilité des ressources après l'ordonnancement de chaque produit. La complexité de ces algorithmes est également étudiée précisément pour permettre leur utilisation en temps réel.

L'étude proposée dans cette thèse a déjà fait l'objet de publications dans des revues scientifiques prestigieuses [Chauvet et *al.*, 1998a] [Chauvet et *al.*, 1998b] [Antonio et *al.*, 1999] [Chauvet et Proth, 1999] [Chauvet et *al.*, 1999a] [Chauvet et *al.*, 1999b] et des conférences internationales [Chauvet et Proth, 1998a] [Chauvet et Proth, 1998b] [Chauvet et *al.*, 1998b].

CHAPITRE I

PRESENTATION ET ETAT DE L'ART

Résumé

Ce chapitre présente l'état de l'art des algorithmes d'ordonnancement du point de vue de leur complexité, composante essentielle des problèmes gérés en temps réel. L'originalité de cette présentation réside dans la classification des problèmes. En effet, tous les problèmes d'ordonnancement (y compris ceux qui sont dans lesquels l'attente entre les opérations est permise) sont décrits comme des problèmes sans attente avec une contrôlabilité variable des durées opératoires. Nous présentons également les domaines d'application des problèmes traités dans les chapitres suivants.

Sommaire

I.1. INTRODUCTION

I.1.1. L'ordonnancement

L'ordonnancement vise à définir les **dates d'exécution d'un ensemble d'opérations en tenant compte de la disponibilité des ressources** (matérielles, humaines...) nécessaires, de manière à optimiser un critère donné. L'ordonnancement est généralement réalisé à court terme pour le niveau opérationnel. Souvent, on cherche à placer l'ensemble des opérations de sorte qu'elles soient achevées le plus tôt possible : on dit alors que l'on minimise le *makespan*. D'autres critères existent comme la minimisation de la somme des retards (lorsque des délais de livraison interviennent), la minimisation des en-cours, la minimisation du coût de production, ou encore une répartition équilibrée des charges des ressources.

Le champ d'applications de l'ordonnancement est large : la gestion de la production dans l'industrie, la gestion de projet, l'organisation des emplois du temps, la gestion des processeurs en informatique... Ainsi, les ouvrages consacrés à l'ordonnancement (*scheduling* en anglais) sont nombreux comme le prouvent ces quelques références : Conway et *al.* (1967), Baker (1974), Coffman (1976), Rinnooy Kan (1976), Dempster et *al.* (1982), French (1982), Carlier et Chrétienne (1988), Blazewicz et *al.* (1993), Morton et Pentico (1993), Feldmann et *al.* (1994), Tanaev et *al.* (1994a), Tanaev et *al.* (1994b), Zweben et Fox (1994), Brucker (1995), Chrétienne et *al.* (1995), Pinedo (1995), Chu et Proth (1996), Koole (1996), Lai et *al.* (1997), et Shmelev (1997), Esquirol et Lopez (1999).

Dans la suite, nous considérons que les données relatives à chaque tâche (durées minimales et maximales d'exécution, date de disponibilité...) ne sont pas stochastiques, mais sont connues a priori. On dit, dans ce cas, que le problème est déterministe.

I.1.2. Les applications en temps réel

Dans une application en **temps réel**, on dit aussi **en ligne** (et, en anglais, *real-time* ou *on-line*), il est obligatoire de fournir une solution rapidement, sans pouvoir attendre les informations sur les demandes à long terme. Les premiers problèmes importants traités en temps réel ont concerné le pilotage de systèmes complexes (avion, centrale nucléaire...). De tels problèmes se sont également posés pour l'ordonnancement de radars [Orman et *al.*, 1998], ou de processeurs en informatique [Chrétienne et *al.*, 1995]. L'ordonnancement en temps réel permet aussi de fournir un délai de fabrication précis au client, à l'instant où il passe sa commande. Sgall (1998) décrit la littérature déjà abondante relative à l'ordonnancement "en temps réel".

Aujourd'hui, les applications en temps réel de l'ordonnancement se développent parce qu'elles permettent une grande souplesse dans la prise de décision. En effet, elles autorisent de prendre les décisions rapidement et, ainsi, de réagir dynamiquement aux aléas de l'environnement. De plus, des données techniques sont parfois difficiles à prendre en compte par les systèmes d'ordonnancement. Aussi, il est nécessaire de permettre à l'utilisateur d'intervenir dans le système d'ordonnancement pour contrôler les solutions proposées et lui fournir, le cas échéant, de nouvelles solutions mieux adaptées aux contraintes. Grâce à leur rapidité d'exécution, les applications en ligne fournissent ces possibilités aux utilisateurs.

Les applications en temps réel impliquent de développer des techniques rapides dans tous les cas de figures, permettant de fournir un résultat avant que d'autres événements ne viennent modifier l'état du système. La rapidité d'un algorithme en temps réel est mesurée par sa "**complexité**", c'est-à-dire l'évaluation du nombre d'opérations élémentaires effectuées par l'algorithme dans le pire des cas. Elle est donnée en fonction de la taille des problèmes traités [Knuth, 1968-1973] [Gaudel et *al.*, 1987]. Pour garantir que le temps d'exécution d'un algorithme ne dégénère pas sur des cas d'application en temps réel, il est souhaitable que sa complexité soit polynomiale (i.e. que le temps de calcul s'exprime comme un polynôme fonction de la taille des données).

A l'opposé, les problèmes **NP-difficiles au sens fort** forment une classe pour laquelle on ne connaît pas d'algorithme polynomial [Garey et Johnson, 1979]. Pour résoudre de tels problèmes de manière optimale, les seules méthodes connues consistent à explorer explicitement ou implicitement toutes les solutions, ce qui n'est possible que pour des exemples de taille très limitée. Ainsi, dans le cas d'applications réelles et quand la recherche de la solution optimale devient "combinatoire", on utilise des "heuristiques" rapides qui fournissent, au mieux, une solution proche de la solution optimale.

I.1.3. Caractéristiques des problèmes traités

Dans cette thèse, nous nous intéressons à des problèmes d'ordonnancement caractérisés par deux aspects.

Tout d'abord, le temps séparant la fin d'une opération de la suivante est nul. C'est ce que nous appelons des processus **sans délai inter opératoire**, ou **sans attente**, ou encore **sans temps mort** (dénommés *no-wait problems* dans la littérature anglo-saxonne).

De plus, les temps opératoires (en anglais, *processing times*) ne sont pas fixés a priori : la durée P_i de chaque tâche i peut être choisie dans un intervalle de valeurs admissibles : $P_i \in [l_i \ ; \ u_i]$ avec $l_i \leq u_i$. On parle de **temps opératoires contrôlables** (ou, en anglais, *controllable processing times*) par le système d'ordonnancement. Les valeurs l_i et u_i sont appelées respectivement durée minimale et durée maximale (ou en anglais *lower duration* et *upper duration*) du temps opératoire. Cette spécificité distingue ces problèmes des problèmes

"classiques" d'ordonnancement car il faut non seulement déterminer l'ordre de passage des produits, mais aussi les durées opératoires P_i. Par ailleurs, l'existence de tels intervalles augmente le nombre d'ordonnancements possibles. Suivant le type d'intervalle $[l_i ; u_i]$, les problèmes sans délai inter opératoire peuvent être classés en quatre catégories :

- **les problèmes à temps opératoires fixées (P.T.O.F.)** dans lesquels la durée de chaque tâche est fixée (i.e. $l_i \in \Re^+$ et $u_i = l_i$ pour toute tâche i, où $\Re^+$ désigne l'ensemble des nombres réels positifs). Il s'agit d'un pur problème sans délai inter opératoire. Les problèmes dits *no-wait* dans la littérature anglo-saxonne (pour lesquels il n'est pas précisé que les durées sont contrôlables) sont de ce type [Hall et Sriskandarajah, 1996].
- **les problèmes à temps opératoires illimités (P.T.O.I.)** dans lesquels chaque opération peut être prolongée autant qu'on le souhaite (i.e. $l_i \in \Re^+$ et $u_i = +\infty$ pour toute tâche i). Dans ce cas, une ressource utilisée pour exécuter une opération ne peut être réutilisée tant que le produit n'a pas été évacué, ceci même si l'opération sur le produit est achevée. Ces problèmes sont dits *blocking* [Hall et Sriskandarajah, 1996].
- **les problèmes à temps opératoires limités (P.T.O.L.)** dans lesquels chaque opération peut être prolongée dans une certaine mesure (i.e. $l_i \in \Re^+$ et $u_i \in \Re^+$ pour toute tâche i). Le problème est alors *no-wait* et *limited controllable processing times.*
- **les problèmes mixtes (P.M.)** dans lesquels certaines opérations sont de durée fixe, d'autres peuvent être prolongées de manière limitée ou illimitée (i.e. $l_i \in \Re^+$ et $u_i \in \Re^+ \cup \{+\infty\}$ pour toute tâche i). Ce type de problème est plus général que les 3 précédents et permet de modéliser des systèmes de production mixtes, c'est-à-dire composés d'opérations à durées contrôlables et sans délai inter opératoire, de sous-systèmes de type *blocking*, et même d'opérations où les délais inter opératoires sont permis.

Dans ce chapitre, nous répertorions les principales études académiques relatives à ces quatre types de problèmes. Nous présentons une étude comparative des problèmes dits d'atelier (ou, en anglais, *shop floor*). Même s'ils ne représentent pas toute la diversité des problèmes d'ordonnancement réels, les problèmes d'atelier sont les plus traités dans la littérature et permettent de bien appréhender le niveau de difficulté des quatre types de problèmes sans attente qui nous intéressent.

I.2. PROBLEMES A TEMPS OPERATOIRES FIXES (P.T.O.F.)

I.2.1. Introduction

Les problèmes d'ordonnancement présentés dans ce paragraphe se limitent aux cas où un produit ne peut attendre entre deux opérations successives : dès qu'une opération est terminée, la suivante doit immédiatement commencer. Ils sont dits *no-wait*, ou en abrégé *nwt*, selon la notation anglo-saxone introduite par Graham et *al.* (1979) et étendue notamment par Blazewicz (1987). De plus, la durée de chaque opération est fixée. La durée de toute tâche i est donnée par $P_i = l_i \in \Re^+$. Pour distinguer ce cas où les durées opératoires ne sont pas variables, nous notons la durée opératoire en minuscule, p_i.

Ces problèmes se posent soit à cause des caractéristiques du processus de fabrication, soit du fait de l'absence de possibilités de stockage entre les opérations à effectuer.

Dans la **sidérurgie**, les opérations (coulage, laminage, trempe...) s'enchaînent à la sortie des fours sans possibilité de temps mort, sans quoi la température et la viscosité diminuant, l'acier ne serait plus malléable [Callahan, 1971]. Des contraintes similaires existent dans la métallurgie en général, la **pétro-chimie** [Salvador, 1973] et la **plasturgie**. De même, les processus **chimiques** [Egli et Rippin, 1981] [Reklaitis, 1982] et **pharmaceutiques** requièrent très souvent d'être effectués sans pause inter opératoire. Le conditionnement et la livraison de certains produits **alimentaires** doivent également être effectués sans délai après leur transformation [Ramudhin et Ratliff, 1992]. Dans l'**informatique**, le stockage des résultats intermédiaires est parfois impossible [Reddi et Ramamoorthy, 1972].

Dans d'autres domaines, un délai inter opératoire existe bien, mais sa durée est fixée. Cette pause est alors modélisée comme une opération. C'est le cas de l'ordonnancement des radars. En effet, les opérations effectuées par un **radar** sont de deux types : émission d'une onde électromagnétique et réception de l'onde. Entre ces 2 opérations, un délai fixé doit s'écouler qui peut être modélisé par une opération ne requérant aucune ressource [Orman et *al.*, 1996] [Orman et *al.*, 1998].

I.2.2. Problème de *flow shop*

Le problème de *flow shop* est un problème classique d'atelier : on cherche à déterminer l'ordre dans lequel il faut fabriquer n produits, les n produits devant passer successivement sur m machines dans le même ordre. Comme dans tous les problèmes d'atelier, une machine ne peut exécuter qu'une opération à la fois, et chaque produit ne peut subir qu'une opération à la fois. Sous la contrainte que les opérations successives sur un même produit sont exécutées sans temps d'attente, le problème est noté $F_m|nwt|C_{max}$ [Graham et *al.*, 1979]. Le critère C_{max} signifie

que l'on cherche à minimiser le *makespan*. Röck (1984a) a prouvé que ce problème est **NP-difficile au sens fort dès que l'atelier comporte au moins 3 machines**.

Piehler (1960) a montré que ce problème pouvait être transformé en un **problème de voyageur de commerce**. Pour cela, deux produits fictifs notés 0 et $(n+1)$ sont ajoutés et fabriqués respectivement en début et en fin de séquence. Les durées opératoires de ces deux produits étant nulles, ils n'affectent pas la durée de l'ordonnancement tout en permettant de marquer les deux limites de la séquence. On définit pour tout couple (j, k) de produits :

$$D(j,k) = \max_{i=1,\ 2,\ \ldots,\ m} \left(\sum_{h=1}^{i} p_{h,j} - \sum_{h=1}^{i-1} p_{h,k} \right)$$

où $p_{i,j}$ est le temps opératoire du produit j sur la machine i. Comme le montre la Figure I- 1, la quantité $D(j,k)$ représente la durée séparant les instants de début de fabrication de deux produits, le produit j étant réalisé juste avant le produit k. On considère alors le graphe G à $(n+2)$ sommets où tout arc (j,k) est valué par $D(j,k)$. Chercher quelle séquence des n produits minimise le *makespan* revient alors à déterminer le plus court parcours du voyageur de commerce dans G entre les sommets 0 et $(n+1)$. Cette transformation a permis d'utiliser les techniques dédiées au problème du voyageur de commerce, pour résoudre le problème $F_m|nwt|C_{max}$. Pekny et Miller (1991), Song et *al.* (1993), Gandadharan et Rajendran (1993), Rajendran (1994) ont proposé des méthodes exactes et des solutions heuristiques à ce problème. Glass et Potts (1996) ont comparé les différentes méthodes de recherche locale pour les problèmes de *flow shop* sans contrainte de *no-wait*.

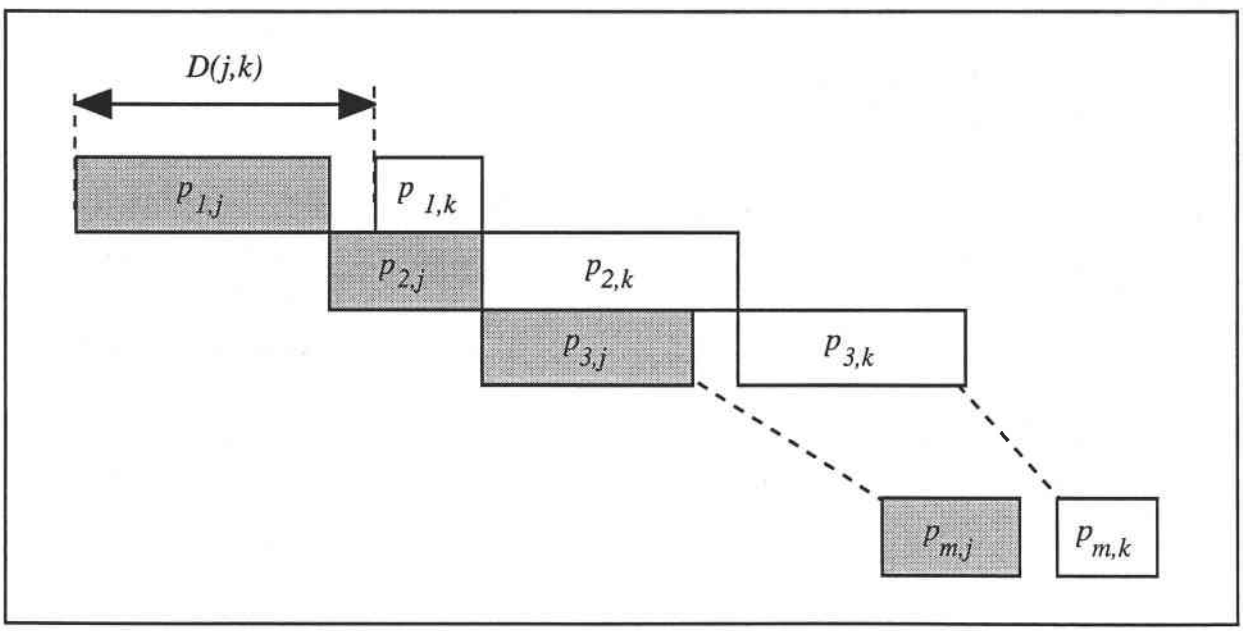

Figure I- 1 : Transformation en un problème de voyageur de commerce

I.2.3. Problème de *flow shop* à 2 machines

Le cas réduit à 2 machines a été étudié par **Gilmore et Gomory (1964)**. Ces auteurs ont montré que l'opération la plus courte sur la première machine et l'opération la plus courte sur la deuxième machine doivent être exécutées simultanément si possible. De même, les opérations classées suivant les durées croissantes sur la première machine et les opérations classées suivant les durées croissantes sur la deuxième machine doivent être exécutées simultanément. Dans le cas où cela n'est pas possible, ils proposent des modifications successives de l'ordonnancement qui le rendent admissible tout en minimisant la date d'achèvement du dernier produit. L'algorithme obtenu permet de résoudre le problème $F_2|nwt|C_{max}$ de manière optimale en **un temps** $O(n \log_2 n)$ [Hall et Sriskandarajah, 1996].

Plusieurs problèmes ont pu être résolus grâce à des adaptations de cet algorithme [Reddi et Ramamoorthy, 1972]. Ainsi, cet algorithme fournit une solution au problème $F_2|nwt|C_t$ dans lequel le critère consiste à réduire le temps de cycle dans un système de type *flow shop* répétitif [Matsuo, 1990]. Röck (1984b) a proposé un algorithme de complexité $O(n \log_2 n)$ utilisant la méthode de Gilmore et Gomory, pour le cas $F_2|nwt, p_{i,j}=1, res1..l|C_{max}$ où toutes les opérations ont la même durée et requièrent, en plus de leur ressource propre, une ressource commune. Le temps de changement d'outil peut également être pris en compte [Gupta et *al.*, 1990]. Sriskandarajah et Ladet (1986) ont montré que le problème à 3 machines pour lesquelles les durées sont identiques, $F_3|nwt, p_{i,j}=1|C_{max}$, peut être résolu en un temps $O(n^2)$.

I.2.4. Problème de *job shop*

Le problème de *job shop* est un problème d'atelier plus général que le *flow shop*. En effet, on cherche à déterminer l'ordre dans lequel il faut fabriquer n produits, chaque produit ayant sa propre gamme de fabrication. Chaque opération est exécutée par une des m machines du parc, qui est éventuellement utilisée à plusieurs stades de la fabrication. Il est noté $J_m|nwt|C_{max}$, de manière analogue au problème *flow shop* $F_m|nwt|C_{max}$. Ce problème est **NP-difficile au sens fort même pour 2 machines** [Sahni et Cho, 1979]. Blazewicz et *al.* (1996) ont présenté les différentes approches exactes et approchées pour les problèmes de *job shop* (avec ou sans délai inter opératoire). Des résultats ont été apportés concernant la stabilité des solutions par Sotskov et *al.* (1997), et de nouvelles heuristiques ont été proposées par Schuurman et Woeginger (1997).

Pour un *job shop* à 2 machines dans lequel toutes les opérations sont de durée égale, le problème est NP-difficile [Timkovsky, 1985a]. Les problèmes NP-difficiles forment une classe de problèmes dans laquelle les problèmes NP-difficiles au sens fort sont inclus [Garey et Johnson, 1979]. Kubiak (1989) a fourni un algorithme dit pseudo polynomial en $O(K n^5)$, où K représente le nombre total d'opérations à effectuer. Pour plus de 2 machines (i.e. $m>2$),

Sriskandarajah et Ladet (1986) ont prouvé que $J_m|nwt, prec, p_{i,j}=1|C_{max}$ est NP-difficile au sens fort. Kravchenko (1998) a étudié le cas d'un ordonnancement sans attente de n produits. Chaque produit est une chaîne d'opérations à exécuter alternativement sur 2 machines. Un algorithme en $O(n^6)$ permet de fournir la solution du problème $J_2|nwt, p_{i,j}=1|\Sigma C_j$ dans lequel on cherche à minimiser le flot moyen (i.e. la somme des dates d'achèvement des produits). Ce problème devient NP-difficile au sens fort si les durées ne sont pas égales [Röck, 1984b]. Timkovsky (1998a) a étendu ces résultats, prouvant qu'en présence de précédences de type chaîne entre les produits, le problème $J_2|nwt, chain, p_{i,j}=1|C_{max}$ est NP-difficile au sens fort.

I.2.5. Problème d'*open shop*

Le problème d'*open shop* est un autre problème "classique" : on cherche à déterminer l'ordre des n produits à fabriquer, les n produits devant passer successivement sur les m machines, dans un ordre quelconque. Il est noté $O_m|nwt|C_{max}$. Ce problème **est NP-difficile au sens fort même pour 2 machines** [Sahni et Cho, 1979]. Récemment, Sidney et Sriskandarajah (1999) ont proposé une heuristique dédiée à ce problème. Le problème $O_m|nwt, pmtn|C_{max}$ (qui autorise d'interrompre une opération pendant une période durant laquelle une autre opération est exécutée en continu) reste NP-difficile au sens fort [Strusevich, 1991].

La plupart des cas d'*open shop* pour lesquels on connaît un algorithme polynomial sont les problèmes dans lesquels toutes les opérations sont de durée égale. C'est ce qu'ont considéré Brucker et *al.* (1977), Garey et Johnson (1977), Lenstra et Rinnooy Kan (1978), Monma (1982), Simons et Warmuth (1989), Kubiak et *al.* (1991), Brucker et *al.* (1993), et Timkovsky (1998a-c). Dans ce cas, le temps opératoire nécessaire à chaque produit vaut m unités de temps. Dans de telles situations, un algorithme en $O(n)$ [Hu, 1961] [Sethi, 1976] permet de résoudre les problèmes d'assemblage et désassemblage de type $O|nwt, tree, p_{i,j}=1|C_{max}$. Dans le cas de précédences quelconques, le même problème $O|nwt, prec, p_{i,j}=1|C_{max}$ est NP-difficile au sens fort [Ullman, 1975]. Adiri et Amit (1984) ont proposé un algorithme en $O(nm)$ pour $O_m|nwt, p_{i,j}=1|\Sigma C_j$ où le critère est la minimisation du flot moyen. Utilisant les résultats de Coffman et Graham (1972), Lawler (1976) a proposé un algorithme en $O(n^2)$ pour un *open shop* à 2 machines avec des relations de précédence entre certains produits : $O_2|nwt, prec, p_{i,j}=1|C_{max}$.

I.2.6. Conclusion

Les problèmes d'ordonnancement sans attente apparaissent intrinsèquement difficiles (cf. Tableau I- 1). Il faut limiter le nombre de machines à 2 pour espérer résoudre un problème de manière optimale en un temps polynomial. Sriskandarajah et Ladet (1986), Goyal et Sriskandarajah (1988), Brucker et *al.* (1993), Ramudhin et Ratliff (1994), et Hall et Sriskandarajah (1996) fournissent d'autres résultats sur ces problèmes.

Tableau I- 1 : Difficulté des problèmes à temps opératoires fixées

Problème	Complexité	Référence
$F_3 \mid nwt \mid C_{max}$	NP-difficile au sens fort	Röck (1984a)
$J_2 \mid nwt \mid C_{max}$	NP-difficile au sens fort	Sahni et Cho (1979)
$J_2 \mid nwt \mid \Sigma C_j$	NP-difficile au sens fort	Röck (1984b)
$O_2 \mid nwt \mid C_{max}$	NP-difficile au sens fort	Sahni et Cho (1979)
$O_2 \mid nwt, pmtn \mid C_{max}$	NP-difficile au sens fort	Strusevich (1991)
$J_3 \mid nwt, p_{i,j}=1 \mid C_{max}$	NP-difficile au sens fort	Sriskandarajah et Ladet (1986)
$J_2 \mid nwt, chain, p_{i,j}=1 \mid C_{max}$	NP-difficile au sens fort	Timkovsky (1998a)
$J_2 \mid nwt, chain, p_{i,j}=1 \mid \Sigma C_j$	NP-difficile au sens fort	Timkovsky (1998a)
$O \mid nwt, prec, p_{i,j}=1 \mid C_{max}$	NP-difficile au sens fort	Ullman (1975)
$O \mid nwt, tree, p_{i,j}=1 \mid C_{max}$	Soluble en $O(n)$	Hu (1961), Sethi (1976)
$O_m \mid nwt, p_{i,j}=1 \mid \Sigma C_j$	Soluble en $O(nm)$	Adiri et Amit (1984)
$F_2 \mid nwt \mid C_{max}$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_2 \mid nwt \mid C_t$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_2 \mid nwt, p_{i,j}=1, res1..1 \mid C_{max}$	Soluble en $O(n \log_2 n)$	Röck (1984b)
$F_3 \mid nwt, p_{i,j}=1 \mid C_{max}$	Soluble en $O(n^2)$	Sriskandarajah et Ladet (1986)
$J_2 \mid nwt, p_{i,j}=1 \mid C_{max}$	NP-difficile Soluble en $O(K n^5)$	Timkovsky (1985a) Kubiak (1989)
$J_2 \mid nwt, p_{i,j}=1 \mid \Sigma C_j$	Soluble en $O(n^6)$	Kravchenko (1998)
$O_2 \mid nwt, prec, p_{i,j}=1 \mid C_{max}$	Soluble en $O(n^2)$	Coffman et Graham (1972), Lawler (1976)

n est le nombre de produits à fabriquer

m représente le nombre de machines dans l'atelier

K représente le nombre total d'opérations à effectuer

I.3. PROBLEMES A TEMPS OPERATOIRES ILLIMITES (P.T.O.I.)

I.3.1. Introduction

Dans les problèmes d'ordonnancement présentés dans ce paragraphe, il n'est pas possible de stocker un produit entre des opérations successives. Cependant, **un produit pour lequel une opération est terminée peut rester sur une machine**, la rendant indisponible pour toute autre opération : c'est une situation de blocage. Ces problèmes sont connus sous le nom de *blocking problems* [Hall et Sriskandarajah, 1996]. Chaque opération i a une durée P_i qui peut être choisie dans un intervalle $P_i \in [l_i\ ;\ +\infty[$, où la durée minimale $l_i \in \Re^+$ est donnée.

Ces problèmes surviennent dans les **systèmes automatisés**. Dans les chaînes d'assemblage (comme celles de l'automobile), un robot ne peut commencer une série d'opérations sur un produit tant que le produit précédent ne l'a pas libéré [Logendra et Sriskandarajah, 1996]. Des contraintes identiques apparaissent dans les systèmes de production équipés **de convoyeurs ou d'AGVs** (*Automated Guided Vehicule*) [Kise et *al.*, 1991] [Ganesharajah et *al.*, 1998]. Dans l'aéronautique, les produits de trop grande dimension ne permettent pas de stockages intermédiaires et quittent un poste de travail quand l'emplacement suivant n'est plus occupé. Notons enfin qu'un **système *Kan Ban*** crée de tels blocages puisqu'un produit ne peut être fabriqué tant qu'aucun *Kan Ban* n'est disponible.

I.3.2. Problème de *flow shop*

Considérons le problème d'atelier type *flow shop* (déterminer la séquence des n produits devant passer successivement sur m machines dans le même ordre). S'il est de type *blocking*, il est noté $F_m|block|C_{max}$. Hall et Sriskandarajah (1996) ont présenté le problème de *flow shop* à 2 machines avec un en-cours limité de capacité b=1 comme un cas particulier de $F_3|block|C_{max}$ où la deuxième opération aurait une durée minimale nulle. Ce problème, noté $F_2|b|C_{max}$ est NP-difficile au sens fort pour $0<b<n-1$ [Papadimitriou et Kanellakis, 1980]. En conséquence, $F_m|block|C_{max}$ est NP-difficile au sens fort dès que l'atelier comporte 3 machines.

Dans le cas $F_2|b=0|C_{max}$, le stockage est interdit entre les machines, les produits ne peuvent attendre que sur les machines et le problème est alors identique à $F_2|block|C_{max}$. Il est possible de transformer $F_2|block|C_{max}$ en un problème de voyageur de commerce [Pinedo, 1995]. Cette transformation permet de démontrer **l'équivalence de $F_2|block|C_{max}$ avec le problème $F_2|nwt|C_{max}$** (ce qui signifie que ces deux problèmes sont de même niveau de difficulté). Ainsi, comme dans le cas du problème à temps opératoires fixées, l'algorithme de Gilmore et Gomory (1964) est utilisé pour résoudre le problème $F_2|block|C_{max}$. De même, les problèmes $F_2|nwt|C_t$ et $F_2|block|C_t$ sont équivalents. Des résultats existent concernant des cas particuliers où les durées

de certaines opérations sont identiques. Si toutes les opérations sont de durées minimales identiques (i.e. $l_i=l$), il n'est pas nécessaire de prolonger une opération au-delà de la durée l pour atteindre l'ordonnancement optimal. Aussi, dans ce cas, les algorithmes développés pour les problèmes sans attente à durées opératoires fixées peuvent être utilisés pour les problèmes de type *blocking*. Dans le cas où toutes les opérations d'un même produit ont la même durée [Pinedo, 1995], un simple tri [Beauquier et *al.*, 1992] des produits dans l'ordre croissant des durées suffit à fournir une séquence optimale de F|block, $p_{i,j}=p_j|C_{max}$. Le problème F_m |block, $p_{i,j}=p_i|C_t$ où les **produits sont identiques et fabriqués de manière cyclique**, peut être résolu par un algorithme proposé par Hartmann et Orlin (1993) **en $O(m^2)$**.

I.3.3. Problème de *flow shop* avec robot

Un problème dérivé du problème d'atelier de type *flow shop* prend en compte les déplacements du produit entre opérations par un système robotisé. Le but est alors de déterminer la séquence des n produits et le trajet du (ou des) robot(s). La Figure I- 2 présente un exemple de tel système robotisé. Les gammes de chaque produit sont constituées des opérations effectuées par les m machines et des opérations de déplacement exécutées par le robot entre deux opérations de transformation. Si on néglige les déplacements à vide du robot, ce problème est de type *job shop*. L'entrée et la sortie des produits peuvent également être pris en compte en ajoutant éventuellement des opérations fictives. Vu les nombreuses applications [Lei et Wang, 1989] [Varnier, 1996] [Crama, 1997] [Kats et Levner, 1997], ce problème a fait l'objet d'études spécifiques, en particulier pour la minimisation du temps de cycle dans une production cyclique.

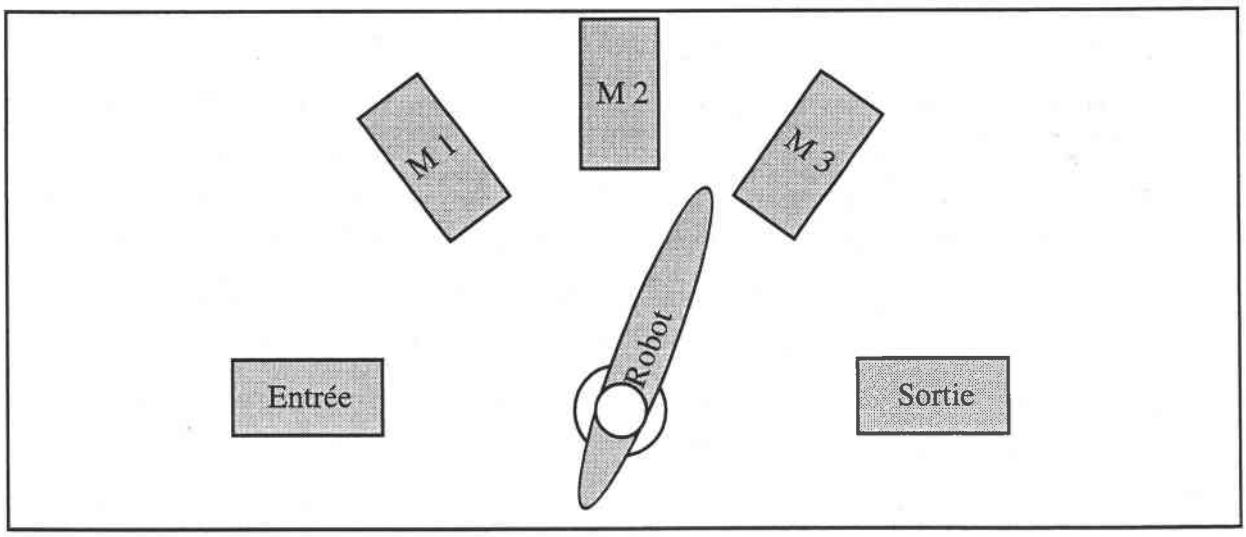

Figure I- 2 : Système robotisé à 3 machines

Dans ce type de problèmes, on suppose généralement que les durées des déplacements du robot vérifient l'inégalité triangulaire, sans quoi le problème devient combinatoire [Brauner et *al.*,

1997]. Des extensions de l'algorithme de Gilmore et Gomory (1964) pour **un *flow shop* à 2 machines** permettent de trouver l'ordonnancement optimal. Ces algorithmes sont **polynômiaux sous certaines hypothèses** : produits identiques [Crama et van de Klundert, 1997], ou connaissance du nombre de fois où un produit est réalisé dans la séquence périodique [Sethi et *al.*, 1992] [Hall et *al.*, 1997]. Au delà de 3 machines, il devient NP-difficile de déterminer l'ordonnancement cyclique, même pour un produit unique, du fait du mouvement du robot [Hall et *al.*, 1997]. D'autres travaux ont cherché à déterminer le temps cycle optimal pour une séquence d'entrée des produits et des déplacements du robot fixés [Hartmann et Orlin, 1993]. Des informations détaillées sont proposées par Hall et *al.* (1997), et Crama et *al.* (1998).

I.3.4. Conclusion

Les problèmes d'ordonnancement de type *blocking* sont étudiés depuis peu. De ce fait, de nombreux problèmes restent encore ouverts. Généralement, il apparaît qu'un *flow shop* de type *blocking* est au moins aussi difficile que le problème de *no-wait* correspondant. Le Tableau I- 2 reprend les résultats présentés ci-dessus. Des compléments peuvent être trouvées dans [Pinedo, 1995] et [Hall et Sriskandarajah, 1996].

Tableau I- 2 : Difficulté des problèmes à temps opératoires illimités

Problème	Complexité	Référence
$F_3 \mid block \mid C_{max}$	NP-difficile au sens fort	Hall et Sriskandarajah (1996)
$F \mid block, p_{i,j}=p_j \mid C_{max}$	Soluble en $O(n \log_2 n)$	Pinedo (1995)
$F_2 \mid block \mid C_{max}$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_2 \mid block \mid C_t$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_2 \mid b \mid C_{max}$ $(0<b<n-1)$	NP-difficile au sens fort	Papadimitriou et Kanellakis (1980)
$F_3 \mid b \mid C_{max}$ (b=0)	NP-difficile au sens fort	McCormick et Rao (1994)
$F_2 \mid b \mid C_{max}$ (b=0)	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_m \mid block, p_{i,j}=p_i \mid C_t$	Soluble en $O(m^2)$	Hartmann et Orlin (1993)

n est le nombre de produits à fabriquer

m représente le nombre de machines dans l'atelier

I.4. PROBLEMES A TEMPS OPERATOIRES LIMITES (P.T.O.L.)

I.4.1. Introduction

Les problèmes d'ordonnancement présentés dans ce paragraphe sont également sans attente : il n'est pas possible de stocker un produit entre des opérations successives. Cependant, **le produit peut rester sur une machine au delà de la durée effective de "fabrication"** l_i. Mais P_i, **durée de séjour sur la machine, est limitée** à $u_i \in \Re^+$ connu a priori. La machine est alors indisponible pour toute autre opération. La durée P_i d'une opération est contrôlable dans une certaine mesure : $P_i \in [l_i \ ; \ u_i]$ où $l_i \in \Re^+$ et $u_i \in \Re^+$. Certaines réactions d'un processus **chimique** peuvent être prolongées d'une durée raisonnable sans détérioration du produit, mais dès qu'une opération est terminée, la suivante doit commencer sans attendre [Hertz et *al.*, 1996] [Varnier, 1996]. Des contraintes similaires existent également dans l'industrie **alimentaire.**

I.4.2. Problèmes classiques d'atelier

Dans le cas des problèmes à temps opératoires limités, la durée effective d'une opération peut être choisie dans un intervalle $[l_i \ ; \ u_i]$. Ainsi, les problèmes à temps opératoires fixés (P.T.O.F.) (i.e. $l_i = u_i$) sont des cas particuliers des problèmes de type P.T.O.L. De ce fait, à chaque P.T.O.F. NP-difficile au sens fort correspond un P.T.O.L. NP-difficile au sens fort. Par conséquent, le problème d'atelier $\mathbf{F}_m|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$ **est NP-difficile au sens fort dès que l'atelier comporte 3 machines.** Il en est de même pour $\mathbf{J}_m|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$ $(m \ge 2)$ **et** $\mathbf{O}_m|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$, **dès que le nombre de machines atteint** $m \ge 2$.

Dans le cas d'un *flow shop* réduit à 2 machines, augmenter la durée d'une opération au delà de la durée minimale ne permet jamais de réduire le *makespan.* Par conséquent, le problème $\mathbf{F_2}|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$ est équivalent au problème $\mathbf{F_2}|\mathbf{nwt}|C_{\max}$ et peut être résolu par l'algorithme de Gilmore et Gomory (1964). De même, $F_2|nwt, l_{i,j} \le P_{i,j} \le u_{i,j}|C_t$ et $F_2|nwt|C_t$ sont deux problèmes équivalents. Au delà de deux machines, les problèmes $F_m|nwt, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$ et $F_m|nwt|C_{\max}$ ne sont plus similaires. En effet, augmenter la durée d'une opération peut permettre de réduire le *makespan.* La Figure I- 3 montre les ordonnancements optimaux de 3 produits notés A, B et C sur 3 machines correspondant au cas où aucune opération ne peut être prolongée et au cas où la deuxième opération du produit B peut être prolongée d'une unité de temps. Dans le premier cas, le temps nécessaire pour fabriquer les 3 produits est de 12 unités. Dans le second, alors qu'une opération est prolongée, le *makespan* est réduit à 11 unités. En fait, l'allongement de la durée opératoire permet d'espérer un meilleur ordonnancement dans ce cas.

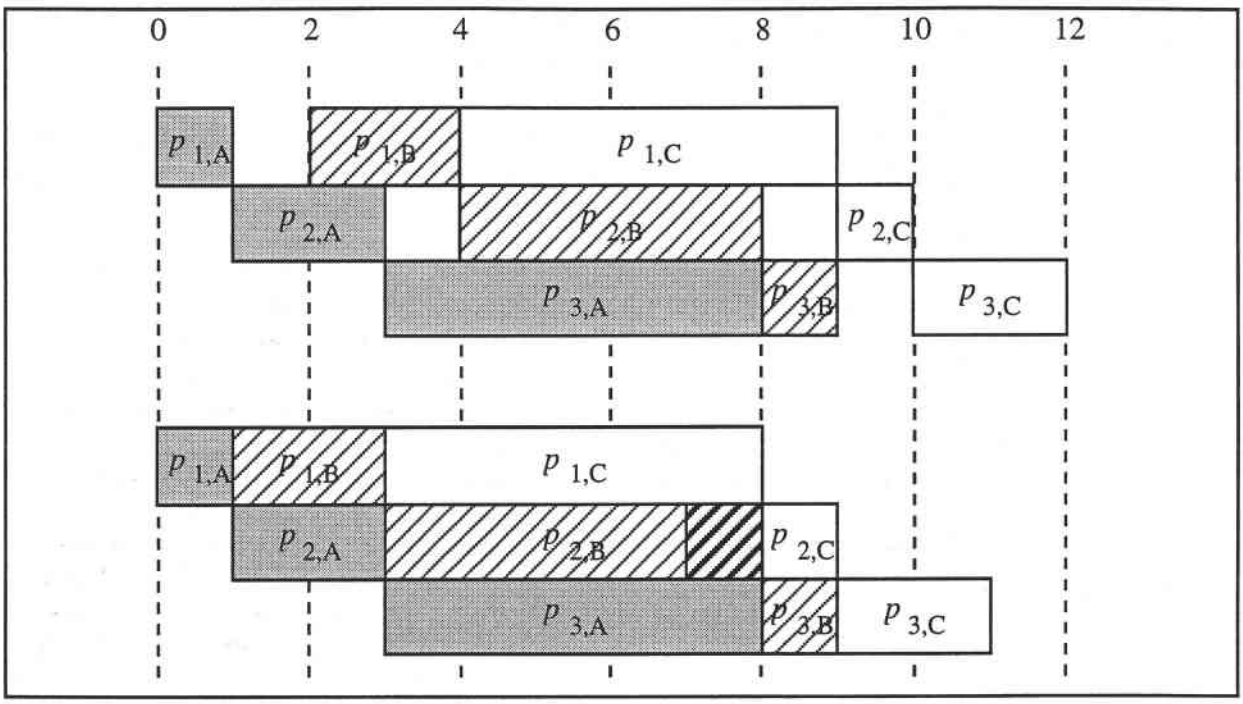

Figure I- 3 : Effet d'un allongement d'une durée opératoire

I.4.3. Autres problèmes d'atelier

Des critères autres que la minimisation du *makespan* ont été examinés en tenant compte non seulement de la productivité du système mais aussi de **l'investissement nécessaire**. Dans [Strusevich, 1995], il est supposé que les produits sont fabriqués de manière répétitive et que la durée des opérations est proportionnelle à la vitesse des machines. On cherche alors à déterminer le niveau technologique des machines (i.e. leur vitesse) qui induit le profit maximal, prenant en compte la productivité du système (directement fonction du *makespan*) et l'investissement nécessaire en machines en tenant compte de leur vitesse. Strusevich (1995) propose un algorithme en $O(n^3)$ pour résoudre ce problème dans le cas d'un atelier de type *flow shop* à n produits et 2 machines. Une approche par algorithme génétique [Karabati et Kouvelis, 1997] a été développée pour des problèmes plus généraux. D'autres auteurs tels Vickson (1980), Nowicki et Zdrzalka (1990), van Hoesel (1991), van Vielt (1991), Panwalkar et Rajagopalan (1992), Nowicki (1994), et Trick (1994) proposent des algorithmes pour les problèmes d'atelier avec des durées opératoires contrôlables mais sans contrainte de type *no-wait* entre opérations.

I.4.4. Conclusion

Les problèmes d'ordonnancement à temps opératoires limités ont été très peu étudiés. Néanmoins, des résultats peuvent être déduits des problèmes de *no-wait* comme indiqué dans le Tableau I- 3.

Tableau I- 3 : Difficulté des problèmes à temps opératoires limités

Problème	Complexité	Référence
$F_3 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	NP-difficile au sens fort	Röck (1984a)
$J_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	NP-difficile au sens fort	Sahni et Cho (1979)
$O_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	NP-difficile au sens fort	Sahni et Cho (1979)
$F_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_t$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)

n est le nombre de produits à fabriquer

m représente le nombre de machines dans l'atelier

$l_{i,j}$ est la durée minimale d'une opération et appartient à $\Re^+$

$u_{i,j}$ est la durée maximale d'une opération et appartient à $\Re^+$

I.5. PROBLEMES MIXTES (P.M.)

I.5.1. Introduction

Dans ce paragraphe, les problèmes d'ordonnancement qui nous intéressent sont sans attente : dès qu'une opération est terminée, la suivante doit immédiatement commencer. Mais, alors que certaines opérations ont leur durée fixée, d'autres opérations peuvent être prolongées sans limite, et d'autres, enfin, peuvent être prolongées de manière limitée. La durée P_i de toute tâche i peut être choisie dans un intervalle de valeurs admissibles $P_i \in [l_i \ ; \ u_i]$ avec $l_i \leq u_i$. Les extrémités de cet intervalle, appelées respectivement durée minimale et durée maximale, sont données pour chaque tâche i : $l_i \in \Re^+$ et $u_i \in \Re^+ \cup \{+\infty\}$.

Ces problèmes permettent d'appréhender des **systèmes de production complexes** dont une part est sans attente et à temps opératoires limités, une autre est sans attente et à temps opératoires illimités. Ils sont très fréquents dans toutes les industries. En effet, les systèmes à temps opératoires limités de la sidérurgie [Callahan, 1971], ou la pétrochimie [Salvador, 1973] sont rarement isolés, mais inclus dans des systèmes de transformation plus importants qu'il faut gérer de concert. De même, les systèmes automatisés ou équipés de convoyeurs [Kise et *al.*, 1991] sont intégrés au système de production, tel celui représenté dans la Figure I- 4.

D'autre part, les systèmes où **l'attente et le stockage sont permis** peuvent être vus comme des systèmes mixtes ayant une structure particulière. Toutes les attentes et les stockages sont alors modélisés comme des opérations à part entière. Cette modélisation donne l'avantage de permettre de gérer et contrôler les lieux de stockage comme les machines.

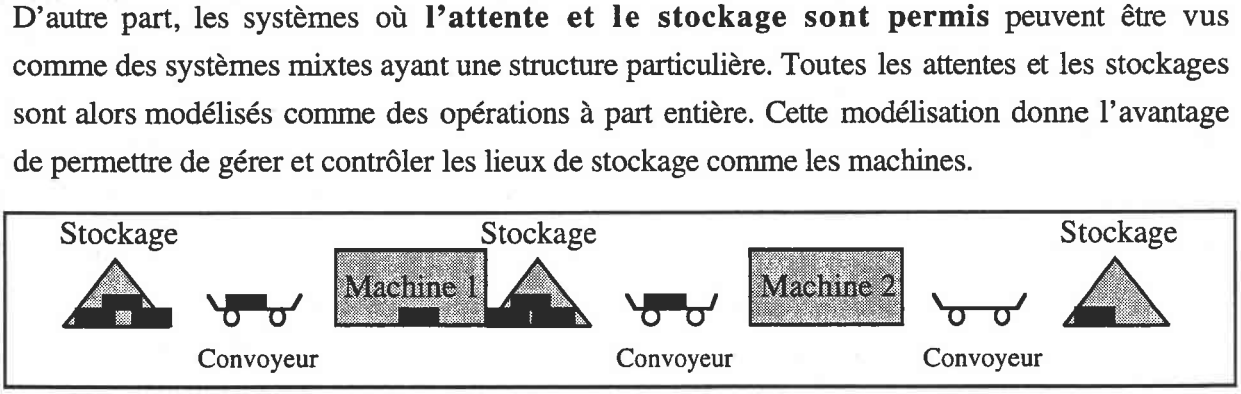

Figure I- 4 : Système de production complexe

I.5.2. Problèmes classiques d'atelier

Les problèmes mixtes englobent les problèmes présentés dans les paragraphes précédents et les problèmes avec attente. L'inconvénient résultant de leur généralité est la difficulté de résolution de tels problèmes. En particulier, les problèmes NP-difficiles présentés précédemment de type P.T.O.F., P.T.O.I. ou P.T.O.L., sont également NP-difficiles au sens fort pour les problèmes

mixtes. Ainsi, le problème d'atelier $\mathbf{F}_m|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$ **est NP-difficile au sens fort dès que l'atelier comporte 3 machines.** Il en est de même pour $\mathbf{J}_m|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}(m \ge 2)$ et $\mathbf{O}_m|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$, **dès que le nombre de machines est** $m \ge 2$.

Rappelons que la durée maximale de certaines opérations d'un problème mixte peut être infinie, $u_{i,j} \in \Re^+ \cup \{+\infty\}$. Ceci est la différence essentielle avec les problèmes à temps opératoires limités. Dans le cas d'un *flow shop* réduit à 2 machines, l'allongement possible des opérations n'apporte pas de souplesse supplémentaire. Ainsi, le problème $\mathbf{F}_2|\mathbf{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$ est équivalent au problème $F_2|nwt|C_{max}$ et peut être résolu par **l'algorithme de Gilmore et Gomory** (1964).

I.5.3. Problème d'atelier sans contrainte de *no-wait*

Les problèmes où l'attente et le stockage sont permis peuvent être vus comme des problèmes mixtes sans attente dans lesquels l'attente et le stockage peuvent être contrôlés. Par exemple, le problème $F_m||C_{max}$ (i.e. *flow shop* à m machines dans lequel l'attente et le stockage sont permis) est un problème à $(2m+1)$ opérations composées de m opérations de transformation sur les machines, et de $(m+1)$ opérations de stockage (comprenant le stockage initial, les $(m-1)$ stockages intermédiaires et le stockage final). Les opérations de transformation sur les machines ont des durées fixes tandis que les opérations de stockage ont des durées nulles et peuvent être prolongées sans limite.

De très nombreux travaux ont permis de classer, du point de vue de leur complexité, les problèmes dans lesquels l'attente et le stockage sont permis : Garey et *al.* (1976), Gonzalez et Sahni (1978), Graham et *al.* (1979), Lageweg et *al.* (1981), Blazewicz et *al.* (1983), Lawler (1983), Liu et Bulfin (1985), Kawaguchi et Kyan (1987), Lawler et *al.* (1989), Kubiak et *al.* (1991), Sotskov (1991), Albers et Brucker (1993), Hoogeveen et *al.* (1994), Sriskandarajah et Wagneur (1994), Brucker et *al.* (1995), Brucker et *al.* (1997), Gladky (1997), Brucker et Knust (1998), et Jain et Meeran (1999). La plupart des problèmes réels avec attente ne peuvent pas non plus être résolus de manière optimale en temps réel. En effet, **les problèmes de *flow shop*** $\mathbf{F}_3||C_{max}$, $\mathbf{F}_2||\Sigma C_j$ **et** $\mathbf{F}|prec, p_{i,j}=1|C_{max}$, **de *job shop*** $\mathbf{J}_2||C_{max}$, $\mathbf{J}_2||\Sigma C_j$ **et** $\mathbf{J}_3|p_{i,j}=1|C_{max}$ **et d'*open shop*** $\mathbf{O}_2|r_j|C_{max}$ **et** $\mathbf{O}_2||\Sigma C_j$ **sont NP-difficiles au sens fort.** La notation r_j indique l'existence d'une date de disponibilité pour chaque produit : le produit j ne peut être commencé avant cette date. Le Tableau I- 4 donne les références relatives à chacune des études. Vu le niveau de difficulté de ces problèmes de nombreuses heuristiques leur ont été dédiées, notamment des méthodes approchées à base de règles [Panwalker et Iskander, 1977] [Blackstone et *al.*, 1982]. Le problème *open shop* pour 3 machines $O_3||C_{max}$ est NP-difficile [Gonzalez et Sahni, 1976]. Gonzalez et Sahni (1976) ont fourni un algorithme en $O(n)$ pour le cas limité à 2 machines $O_2||C_{max}$. Johnson (1954) a montré que le problème

correspondant de *flow shop*, $F_2||C_{max}$, est soluble en $O(n \log_2 n)$. Jackson (1956) a étendu la méthode pour le cas d'un *job shop* $J_2|m_j \leq 2|C_{max}$ pour lequel tous les produits n'ont pas plus de deux opérations.

Les problèmes à durées opératoires égales ont fait l'objet d'études spécifiques. Bruno et *al.* (1980), et Brucker et Knust (1998) ont développé des algorithmes polynomiaux pour le *flow shop*. Le *job shop* pour 2 machines et à opérations de même durée a été aussi très étudié. Différents algorithmes [Hefetz et Adiri, 1982] [Kubiak et *al.*, 1995] [Timkovsky, 1985b] ont été proposés pour le problème $J_2|p_{i,j}=1|C_{max}$. Récemment, **Timkovsky (1997) a donné un algorithme en $O(n^2)$ qui s'applique aussi en présence de dates de disponibilité $J_2|r_j, p_{i,j}=1|C_{max}$.** Kubiak et Timkovsky (1996) ont fourni un algorithme pseudo polynomial pour le cas $J_2|p_{i,j}=1|\Sigma C_j$. **Tautenhahn et Woeginger (1997) ont développé un algorithme en $O(n^2)$** pour le problème d'*open shop* suivant $O_m|r_j, p_{i,j}=1|\Sigma C_j$ dans lesquels le nombre de machines est constant.

I.5.4. Problème d'atelier à en-cours limité

Les problèmes à en-cours limités sont des problèmes mixtes. Considérons le problème d'atelier de type *flow shop* (déterminer la séquence des n produits devant passer successivement sur m machines dans le même ordre). Dans le cas de **2 machines avec un en-cours limité de capacité b**, le problème est noté $F_2|b|C_{max}$. **Pour 0<b<n-1, ce problème est NP-difficile au sens fort** [Papadimitriou et Kanellakis, 1980]. Papadimitriou et Kanellakis (1980) ont décrit une heuristique qui fournit, dans le pire des cas, une solution dont le *makespan* est à 3/2 de celui de l'ordonnancement optimal d'un $F_2|b=1|C_{max}$.

Dans le cas où le stockage est interdit entre les machines, le problème est identique à $F_2|block|C_{max}$ et $F_2|b=0|C_{max}$ peut être résolu par l'algorithme de Gilmore et Gomory (1964) en $O(n \log_2 n)$. Pour le problème $F_2|b|C_{max}$ avec b≥n-1, Johnson (1954) a proposé un algorithme en $O(n \log_2 n)$.

I.5.5. Conclusion

Deux cas particuliers de problèmes mixtes d'ordonnancement sans attente ont été étudiés : les problèmes où l'attente est permise et les problèmes avec un en-cours limité. Les résultats sont repris dans le Tableau I- 4. Dans ce tableau, n est le nombre de produits à fabriquer ; m représente le nombre de machines dans l'atelier et m_j est le nombre d'opérations effectuées sur le produit j ; K représente le nombre total d'opérations à effectuer ; $l_{i,j}$ est la durée minimale d'une opération et appartient à $\Re^+$; $u_{i,j}$ est la durée maximale d'une opération et appartient à $\Re^+ \cup \{+\infty\}$ (alors que l'ensemble des valeurs possibles pour $u_{i,j}$ était restreint à $\Re^+$ dans le Tableau I- 3).

Tableau I- 4 : Difficulté des problèmes mixtes

Problème	Complexité	Référence
$F_3 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	NP-difficile au sens fort	Röck (1984a)
$J_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	NP-difficile au sens fort	Sahni et Cho (1979)
$O_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	NP-difficile au sens fort	Sahni et Cho (1979)
$F_2 \mid nwt, l_{i,j} \le P_{i,j} \le u_{i,j} \mid C_{max}$	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_3 \mid\mid C_{max}$	NP-difficile au sens fort	Garey et *al.* (1976)
$F_2 \mid\mid \Sigma C_j$	NP-difficile au sens fort	Garey et *al.* (1976)
$F \mid prec, p_{i,j}=1 \mid C_{max}$	NP-difficile au sens fort	Timkovsky (1998c)
$J_2 \mid\mid C_{max}$	NP-difficile au sens fort	Lenstra et Rinnooy Kan (1979)
$J_2 \mid\mid \Sigma C_j$	NP-difficile au sens fort	Garey et *al.* (1976)
$J_3 \mid p_{i,j}=1 \mid C_{max}$	NP-difficile au sens fort	Lenstra et Rinnooy Kan (1979)
$O_3 \mid\mid C_{max}$	NP-difficile	Gonzalez et Sahni (1976)
$O_2 \mid r_j \mid C_{max}$	NP-difficile au sens fort	Lawler et *al.* (1981)
$O_2 \mid\mid \Sigma C_j$	NP-difficile au sens fort	Achugbue et Chin (1982)
$F_2 \mid\mid C_{max}$	Soluble en $O(n \log_2 n)$	Johnson (1954)
$J_2 \mid m_j \le 2 \mid C_{max}$	Soluble en $O(n \log_2 n)$	Jackson (1956)
$J_2 \mid r_j, p_{i,j}=1 \mid C_{max}$	Soluble en $O(n^2)$	Timkovsky (1997)
$J_2 \mid p_{i,j}=1 \mid \Sigma C_j$	Soluble en $O(n \log_2 n \log_2 K)$	Kubiak et Timkovsky (1996)
$O_2 \mid\mid C_{max}$	Soluble en $O(n)$	Gonzalez et Sahni (1976)
$O_m \mid r_j, p_{i,j}=1 \mid \Sigma C_j$	Soluble en $O(n^2)$	Tautenhahn et Woeginger (1997)
$F_2 \mid b \mid C_{max}$ ($0<b<n-1$)	NP-difficile au sens fort	Papadimitriou et Kanellakis (1980)
$F_2 \mid b \mid C_{max}$ ($0<b<n-1$)	NP-difficile au sens fort	Papadimitriou et Kanellakis (1980)
$F_3 \mid b \mid C_{max}$ ($b=0$)	NP-difficile au sens fort	McCormick et Rao (1994)
$F_2 \mid b \mid C_{max}$ ($b=0$)	Soluble en $O(n \log_2 n)$	Gilmore et Gomory (1964)
$F_2 \mid b \mid C_{max}$ ($b \ge n-1$)	Soluble en $O(n \log_2 n)$	Johnson (1954)

I.6. CONCLUSION

Les méthodes d'ordonnancement utilisables en temps réel doivent être rapides, c'est-à-dire de complexité faible. Pourtant, aucun algorithme de complexité polynomiale ne permet de résoudre de manière optimale des problèmes d'ordonnancement, même réduits. Par exemple, l'ordonnancement d'un atelier qui comporterait 3 machines de type $F_3 \| C_{max}$ est NP-difficile au sens fort [Garey et *al.*, 1976].
Notre ambition dans cette thèse est d'offrir des méthodes optimales, utilisables en temps réel et appliquées aux problèmes mixtes. Comme nous l'avons vu, les problèmes mixtes permettent de modéliser des systèmes de production très variés. Ils sont caractérisés par deux aspects.

Premièrement, le temps séparant la fin d'une opération de la suivante est nul. Ce sont des processus **sans délai inter opératoire**, ou encore **sans attente**.

Deuxièmement, les temps opératoires ne sont pas fixés a priori : la durée P_i de chaque tâche i peut être choisie dans un intervalle de valeurs admissibles : $P_i \in [l_i \ ; \ u_i]$ avec $l_i \leq u_i$. Ce sont des processus à **temps opératoires contrôlables**. Les possibilités d'allongement des opérations sont considérées comme propres à chaque opération : certaines opérations peuvent avoir une durée fixée, d'autres opérations peuvent être prolongées sans limite, et d'autres encore ne peuvent être prolongées que de manière limitée. Ces possibilités d'allongement distinguent ces problèmes des problèmes "classiques" d'ordonnancement car il faut non seulement déterminer l'ordre de passage des produits, mais aussi les durées opératoires P_i.

CHAPITRE II

PROBLEMES A GAMMES LINEAIRES

Résumé

Dans ce chapitre, nous proposons des méthodes de gestion en temps réel applicables à des systèmes de production fabriquant des produits dont la gamme est linéaire. Ces systèmes de production sont du type des *problèmes mixtes*, terminologie introduite dans le premier chapitre. Ainsi, notre étude peut s'appliquer à un large éventail de systèmes de production. Nous illustrons notre propos avec l'exemple de la gestion d'un atelier de traitement de surface. Les algorithmes proposés sont originaux et de complexité polynomiale. Ils permettent de déterminer le délai de livraison de chaque produit, et d'ordonnancer chaque produit de manière à minimiser le temps et le coût d'utilisation des ressources.

Sommaire

II.1. INTRODUCTION

Dans ce chapitre, nous nous intéressons à des systèmes de production répondant aux caractéristiques suivantes :

- Chaque produit subit une série d'opérations (i.e. la gamme du produit est linéaire). La séquence de ressources visitées peut varier d'un produit à l'autre comme dans le cas d'un *job shop* de type $J_m|nwt, l_{i,j} \leq P_{i,j} \leq u_{i,j}|C_{max}$.
- Les durées opératoires sont fixées par le gestionnaire, mais doivent être comprises entre deux limites.
- Dès qu'une opération effectuée sur le produit se termine, l'opération suivante doit impérativement commencer.
- Une ressource ne peut être utilisée pour effectuer une opération nouvelle que si elle est libre.
- Le système de gestion ne peut modifier l'ordonnancement des produits en cours de fabrication.

L'objectif est de trouver un ordonnancement en temps réel pour chaque produit dès que sa commande est passée. Pour faciliter la lecture, nous illustrons notre propos par une application à la gestion d'un atelier de traitement de surface que la société SODETEG T.A.I. avait à réaliser.

II.1.1. L'ordonnancement des ateliers de traitement de surface

Un tel atelier est généralement constitué d'un ensemble de cuves contenant les solutions chimiques ou électrolytiques dans lesquelles les pièces à réaliser sont immergées successivement, comme le montre la Figure II- 1. Le transport des pièces entre les cuves est assuré par un ensemble de robots. Ces robots peuvent être des palans de manutention (en anglais *hoist*), des ponts roulants (en anglais *crane*), ou encore des systèmes automatiques circulant sur un rail. Dans ces ateliers, les opérations doivent être exécutées en séquence, sans attente entre les opérations successives. En effet, tout arrêt en cours de traitement pourrait être préjudiciable à la qualité du produit. Par contre, il est possible d'allonger, dans une certaine mesure, la durée des opérations sans dommage pour le produit. Dans ce cas, la cuve dans laquelle s'effectue une telle opération reste inutilisable pour d'autres opérations tant qu'elle n'est pas libérée.

Varnier (1996) a présenté de manière précise ce type d'atelier dans le cas de la galvanoplastie, une technique de dépôt de couche de métal destinée à éviter la corrosion. Des travaux récents ont été produits en particulier pour la minimisation du temps de cycle dans une production cyclique

[Chen et *al.*, 1996] et donnent de nombreuses références [Varnier, 1996]. Des résultats plus généraux ont été introduits sur ce type de problèmes au chapitre précédent [Lei et Wang, 1989] [Crama, 1997] [Kats et Levner, 1997]. Contrairement au cas d'une fabrication cyclique dans laquelle tous les produits sont connus à l'avance, nous ne connaissons les produits à ordonnancer qu'au moment où ils sont commandés. La société SODETEG T.A.I. située à Buc (78) et l'INRIA ont étudié ce type de problème voici 15 ans [Bolignano et *al.*, 1983].

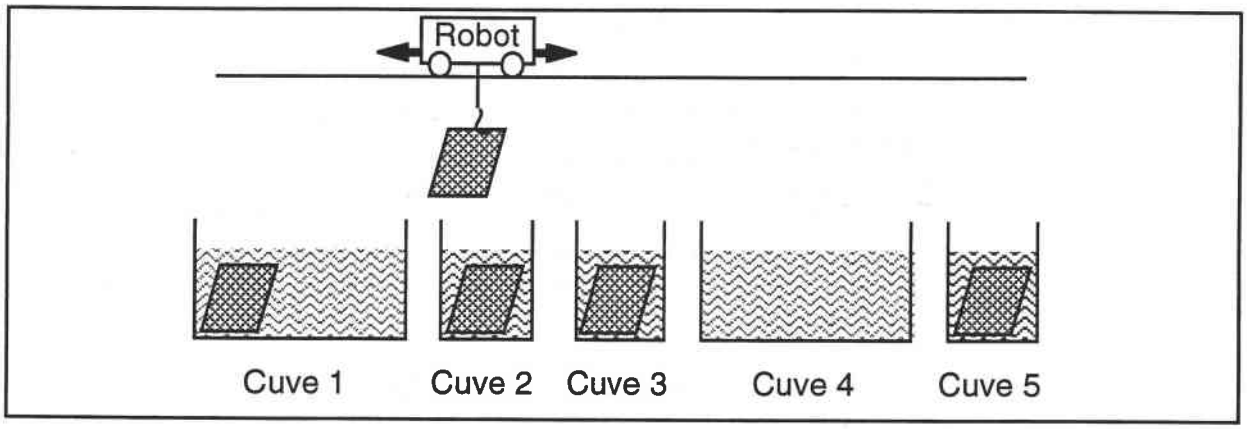

Figure II- 1 : Atelier de traitement de surface

Plusieurs méthodes de gestion en temps réel pour de tels systèmes ont été proposées. Elles sont basées sur des systèmes experts [Yih, 1988] ou des modèles de décision Markoviens [Yih et Thesen, 1991]. Ces méthodes proposent une solution dès qu'un événement se produit (arrivée d'une commande, fin d'une opération sur un produit...). Les décisions sont prises à court terme sans anticiper sur les opérations futures. Ainsi, elles ne garantissent pas le respect des contraintes futures. En particulier, rien ne garantit que les durées maximales de trempe des produits soient respectées à moyen terme. Une telle gestion engendre donc beaucoup de pertes et n'est pas envisagée ici.

Récemment, d'autres algorithmes qui tentent de placer toutes les opérations effectuées sur un produit au moment de son ordonnancement ont été proposés. A chaque nouvelle commande (ou à chaque fois qu'un événement aléatoire se produit, une panne par exemple), les opérations restant à réaliser sont alors ordonnancées. Chaque produit est ordonnancé sans connaître les commandes à venir car il est impossible d'attendre que toutes les commandes soient connues pour réaliser l'ordonnancement. Evidemment, l'ordonnancement obtenu ne garantit pas une productivité aussi importante qu'un ordonnancement qui se ferait en connaissant toutes les opérations à réaliser. Ces algorithmes qui ordonnancent les produits sans connaître les commandes à venir, sont souvent considérés comme des heuristiques dans la littérature.

Les heuristiques classiques d'ordonnancement [Baker, 1974] [Song et *al.*, 1993] ne sont pas applicables dans notre cas, car les durées opératoires varient dans un intervalle donné. Des

heuristiques spécifiques ont été développées. On peut les classer en deux catégories suivant la situation à laquelle elles s'appliquent :

a) **Les premières ne remettent pas en cause les produits précédemment ordonnancés**. C'est dans ce cadre que nous nous situons dans ce chapitre. Ce cadre correspond à l'Hypothèse II- 3 décrite dans la suite. Yin et Yih (1992) ont proposé un algorithme pour répondre à ce problème dans le cas où tous les produits sont immergés dans les cuves selon un même ordre (du type *flow shop*). Ces auteurs se restreignent à la situation où une seule cuve existe pour chaque solution chimique. Enfin, ils n'utilisent la tolérance des durées opératoires que sur les opérations en conflit (i.e. utilisant une cuve ou le robot alors qu'il n'est pas encore disponible), et donc n'assurent pas que le produit sera réalisé le plus tôt possible. Dans la suite, nous verrons comment procéder pour ordonnancer le produit au plus tôt.

b) **Les secondes autorisent de remettre en cause les produits précédemment ordonnancés**. Nous envisagerons cette hypothèse dans les extensions proposées au chapitre suivant. Cette situation, quand elle est possible, permet évidemment d'atteindre une meilleure productivité du système de production. Yih (1994) a modifié la méthode proposée dans [Yin et Yih, 1992] de manière à pouvoir utiliser la tolérance sur les opérations du dernier produit ordonnancé. Cette heuristique ne reconsidère pas la séquence d'entrée des produits, contrairement à d'autres heuristiques proposées depuis. Ces dernières s'appuient sur les problèmes de satisfaction de contraintes [Cheng et Smith, 1995], sur des techniques arborescentes de type Procédure par Séparation et Evaluation Progressive [Ge et Yih, 1995] [Lamothe, 1996], ou sur des méta heuristiques [Bloch, 1999]. Le plus souvent, elles utilisent les algorithmes développés pour la situation a).

Dans ce chapitre, nous sommes dans la situation a) puisque nous supposons qu'il n'est pas possible de remettre en cause des produits précédemment ordonnancés.

Dans d'autres approches, on considère qu'au moment d'ordonnancer un produit, les commandes de produit à venir sont connues et on associe à chaque commande une certaine probabilité qu'elle soit confirmée. Nous ne faisons pas cette hypothèse ici. Cependant, dans la section 5, nous pouvons utiliser les informations sur les commandes à venir pour évaluer la charge de travail future des machines. Nous libérons alors le plus possible de temps sur les machines goulots afin d'espérer augmenter la productivité du système.

II.1.2. Approche proposée

Dans la section 2, nous précisons le cadre de l'étude en présentant les hypothèses faites tout au long de ce chapitre. Notre objectif est de proposer des méthodes de gestion en temps réel qui donnent un ordonnancement de chaque produit au moment où sa commande est enregistrée.

La première méthode de gestion consiste **à fabriquer le produit au plus tôt**. Pour cela, nous définissons les dates d'exécution au plus tôt des opérations effectuées sur le produit compatibles avec les contraintes de notre système de production (c'est l'objet de la section 3). En outre, cette méthode permet de fournir au client un délai minimum de livraison précis.

Ensuite, nous proposons une deuxième méthode de gestion qui revient **à commencer la fabrication du produit au plus tard**, tout en respectant la date de livraison fixée au plus tôt. Dans ce but, nous calculons les dates d'exécution au plus tard des opérations (voir section 4). Cette méthode permet de minimiser le temps pendant lequel des ressources sont utilisées pour fabriquer le produit. L'effet escompté est double : réduire le coût de fabrication du produit, et donner plus de liberté pour ordonnancer les produits futurs.

Enfin, nous donnons une troisième méthode de gestion dont le but est de **minimiser le coût d'utilisation des ressources** (voir section 5), tout en livrant le produit au plus tôt. Cette méthode peut être utilisée pour diminuer l'utilisation des ressources goulots et ainsi espérer accroître la productivité du système.

Ces trois méthodes de gestion garantissent d'obtenir l'optimum relativement à un critère qui est soit une fin de fabrication au plus tôt du produit considéré, un début de fabrication au plus tard, ou encore un coût d'utilisation des ressources aussi réduit que possible. L'évaluation de ces critères est faite produit par produit puisqu'il est impossible d'attendre que toutes les commandes soient connues pour fixer la gestion du système. En ce sens, ces critères sont locaux. **Notre objectif global est d'obtenir une gestion qui maximise la productivité du système de production à long terme**. Dans la section 6, nous nous proposons d'appliquer ces trois méthodes de gestion sur une longue période de production à différents ateliers *job shop* (voir les définitions données dans le chapitre précédent) de type $J_m|nwt, l_{i,j} \leq P_{i,j} \leq u_{i,j}|C_{max}$, où $u_{i,j} \in \Re^+ \cup \{+\infty\}$. Nous comparons alors la productivité du système suivant les méthodes de gestion utilisées.

Chaque section est illustrée par un exemple de produit fabriqué dans un atelier de traitement de surface. Pour simplifier la présentation, nous supposons que les collisions entre robots de transport sont impossibles, et nous négligeons les déplacements à vide des robots. Nous montrons, dans la conclusion, comment nous pouvons étendre les méthodes de gestion proposées dans les sections 3 à 5 aux autres situations. Une partie des résultats obtenus a déjà été publiée dans l'article [Chauvet et *al.*, 1999a].

II.2. HYPOTHESES

Dans cette section, nous décrivons l'ensemble des hypothèses faites tout au long de ce chapitre. Ces hypothèses sont réalistes et permettent de gérer en temps réel une large variété de systèmes de production, dont celui qui est présenté dans la section précédente. En fait, le but est de définir précisément le cadre d'application de cette étude et les notations utilisées.

Hypothèse II- 1 : L'ordonnancement est en temps réel.

Dans cette thèse, nous considérons des systèmes de production où les produits à fabriquer ne sont pas connus à l'avance. Les commandes de produits arrivent successivement, à des instants aléatoires, et nous devons déterminer les dates de début et de fin d'exécution des opérations effectuées sur un produit à l'instant où la commande arrive. Notre objectif est de placer chaque produit de manière à ce qu'il soit achevé le plus tôt possible. L'ordonnancement étant exécuté en temps réel, nous sommes en mesure de fournir une date de livraison précise dès qu'une commande est enregistrée (en supposant, bien entendu qu'aucun dysfonctionnement ne se produise). Pour les besoins de l'exposé, nous désignons par J l'ensemble des produits à ordonnancer. Nous les numérotons de 1 à n dans l'ordre d'arrivée des demandes, n étant le nombre de produits dans J. Dans la pratique, n est infini car l'horizon de fonctionnement du système de production n'est pas limité.

Hypothèse II- 2 : La matière première et les composants du produit à fabriquer sont disponibles.

Nous supposons que la matière première et les composants du produit à fabriquer sont disponibles quand la commande apparaît. Ainsi, la date à laquelle le produit j peut commencer à être fabriqué (en anglais *release time* , ou *release date*) est l'instant où la commande apparaît. Cette date est notée r_j et $r_j \in \Re^+$, où $\Re^+$ désigne l'ensemble des nombres réels positifs. Nous supposons que l'instant initial de fonctionnement du système de production est la date 0. Lorsqu'un approvisionnement (en matière première ou en composant) est nécessaire, la *release date* est reculée d'une durée correspondant au délai d'attente de livraison.

Hypothèse II- 3 : L'ordonnancement d'un produit ne remet pas en question l'ordonnancement des précédents.

Dans ce chapitre, nous considérons que l'ordonnancement d'un produit est défini une fois pour toutes et ne peut donc être remis en question quand une nouvelle commande apparaît. Cela signifie que ni les dates de début et de fin, ni les durées des opérations, ni les ressources

sélectionnées pour effectuer les opérations ne peuvent être modifiées. Cette restriction apparaît dans les situations suivantes.

a) Les coûts de gestion et de production induits par la remise en cause de l'ordonnancement ne sont pas compensés par le gain escompté par le nouvel ordonnancement.

b) Le système de gestion ne peut physiquement modifier les dates de début et de fin d'exécution des opérations sur les produits déjà ordonnancés, car il n'a pas la totale maîtrise du système de production. C'est le cas de systèmes de production qui nécessitent des réglages lourds et minutieux avant l'opération de fabrication elle-même.

c) Un engagement est pris avec les clients dont les commandes sont arrivées antérieurement, et il n'est pas possible de les déplacer.

d) On interdit au système de gestion de modifier les dates de début et de fin d'exécution des opérations car on estime que le flot de nouvelles demandes de fabrication de produits est trop important pour permettre une telle remise en question dans un contexte temps réel. Les demandes de fabrication des produits arrivent une à une et doivent être prises en compte immédiatement.

Bien sûr, un système dans lequel l'ordonnancement des produits peut être remis en cause à tout instant peut permettre d'atteindre une meilleure utilisation des ressources. De tels systèmes sont examinés dans les extensions du chapitre suivant, mais s'avèrent délicats à gérer en temps réel du fait de leur niveau de complexité.

Hypothèse II- 4 : La gamme de chaque produit est linéaire.

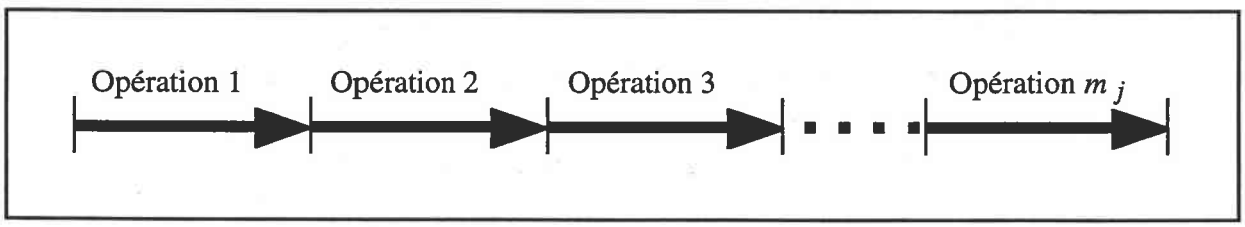

Figure II- 2 : Représentation de la gamme linéaire d'un produit

Nous supposons dans ce chapitre que les gammes des produits à réaliser sont linéaires, c'est-à-dire qu'elles sont constituées d'une suite d'opérations à exécuter séquentiellement. La Figure II-2 représente une telle gamme ; à chaque arc correspond une opération. Nous désignons par opération, ou encore tâche, toute intervention d'un même ensemble de ressources sur un produit pendant une certaine période. Ces ressources peuvent être diverses : opérateur, machine, convoyeur... L'ensemble des ressources est noté R.

Dans le cas d'un atelier de traitement de surface, les ressources sont de trois types : les opérateurs, les différents postes de travail (correspondants à la préparation et au dégraissage,

aux différents types de bains chimiques ou électrolytiques, au rinçage...), les robots transportant les pièces. Comme la représente la Figure II- 3, la gamme d'exécution pour chaque produit consiste en une succession d'actions sur les postes de travail et de mouvements du robot.

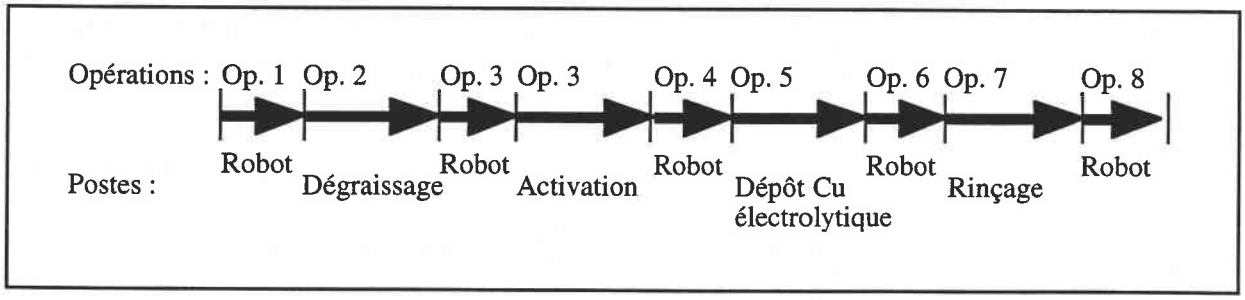

Figure II- 3 : Gamme d'un produit dans un atelier de traitement de surface

Les gammes des produits peuvent être différentes d'un produit à l'autre, comme pour les problèmes d'atelier de type *job shop* présentés dans le chapitre précédent. Cela signifie que les opérations peuvent avoir des durées différentes et peuvent nécessiter des ressources distinctes. Pour le produit j dont la gamme comporte m_j opérations, nous désignons par i la i-ème opération de la gamme avec $i \in \{1, 2, \ldots, m_j\}$. Nous notons I_j l'ensemble des opérations à effectuer sur le produit j : $I_j = \{1, 2, \ldots, m_j\}$.

Les problèmes dans lesquels les gammes comportent des opérations d'assemblage ou de désassemblage seront traités au chapitre suivant.

Hypothèse II- 5 : Nous savons quelles sont les ressources interchangeables en cours d'opération.

Nous supposons que deux types de ressources peuvent apparaître : les ressources interchangeables en cours d'opération et celles qui ne le sont pas. Un exemplaire d'une ressource dite interchangeable peut être remplacé par un autre en cours d'opération. C'est le cas d'opérateurs ayant les mêmes compétences et qui peuvent ainsi se relayer sans que l'opération s'arrête. Inversement, un exemplaire d'un type de machine choisi au début d'une opération ne peut être substitué à un autre : une machine est dite non-interchangeable. L'ensemble des ressources dont les exemplaires sont interchangeables en cours d'opération est noté R_{in}. Celui des ressources non-interchangeables est noté R_{en}. On a : $R = R_{in} \cup R_{en}$.

Hypothèse II- 6 : Le temps de changement de ressource en cours d'opération est nul.

Nous supposons que le temps de passage d'un exemplaire de ressource interchangeable à un autre en cours d'opération est nul.

Hypothèse II- 7 : Chaque opération peut nécessiter l'utilisation de plusieurs types de ressources.

Nous étudions les problèmes pour lesquels chaque opération peut nécessiter des ressources de plusieurs types en quantités différentes, par exemple une machine et deux opérateurs. Nous sommes donc dans une situation plus générale que celle qui est envisagée dans les problèmes de *flow shop* ou de *job shop*. Les ressources sont utilisées pendant toute la durée de l'opération dans la même configuration (i.e. le même nombre d'exemplaires).

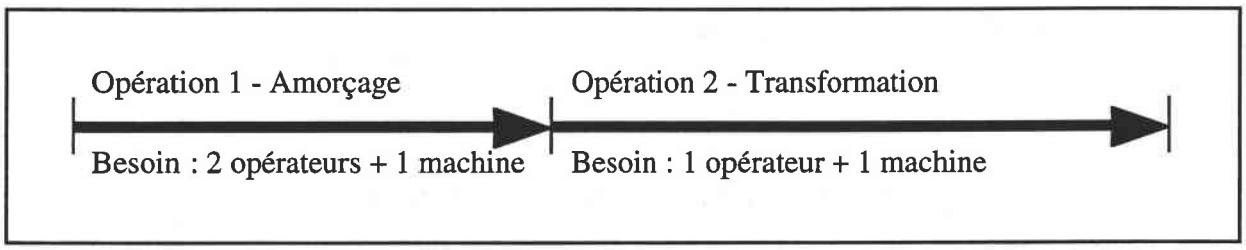

Figure II- 4 : La configuration des ressources est inchangé pendant toute l'opération

En fait, chaque opération est définie de manière à ce que le même ensemble de ressources intervienne pendant toute sa durée. Supposons, comme dans l'exemple de la Figure II- 4, que pour la transformation d'un produit, la présence de deux opérateurs soit nécessaire pour amorcer le processus, alors qu'un seul opérateur suffit pour le reste de la transformation. Nous dirons qu'il y a deux opérations successives : l'amorçage, puis la transformation elle-même, bien que la même machine soit utilisée tout au long du processus.

Nous désignons par $q_{r,i,j}$, le nombre de ressources de type r nécessaires pour exécuter l'opération i sur le produit j, où $r \in R$. Cette quantité est nulle si l'opération correspondante ne nécessite pas de ressource de type r et strictement positive sinon. Donc, $q_{r,i,j} \geq 0$.

Hypothèse II- 8 : L'utilisation d'une ressource ne dépend pas d'autres utilisations de la même ressource.

Nous supposons, dans ce chapitre, que le choix d'une ressource pour une opération n'influence, ni n'est influencé, par les choix de cette ressource pour les autres opérations. Pratiquement, cela signifie que l'utilisation d'une ressource pour une opération donnée n'implique pas son utilisation pour une autre opération sur le même produit. Le cas où le choix d'une ressource pour une opération i_1 effectuée sur un produit j dépend du choix de la ressource pour une autre opération i_2 effectuée sur le même produit j sera étudié dans les chapitres suivants. Ceci peut apparaître, par exemple, quand la même machine doit être utilisée pour des opérations successives. Dans ce cas, nous utilisons une gamme alternative. Nous étudions ce type de choix dans le Chapitre IV.

Hypothèse II- 9 : Les exemplaires d'un même type de ressources sont identiques.

Les exemplaires d'un même type de ressources sont supposés identiques. Par conséquent, ils ont des coûts d'utilisation égaux et des vitesses d'exécution identiques. Il est donc indifférent de choisir un exemplaire plutôt qu'un autre dans un type de ressource donné. Les problèmes où des vitesses d'exécution sont différentes sont traités au Chapitre IV par adjonction de gammes dites alternatives.

Dans le cas d'un atelier de traitement de surface, plusieurs postes de travail peuvent permettre de réaliser la même opération (préparation et dégraissage, bains chimiques ou électrolytiques, rinçage...). Nous distinguons les postes de travail sur lesquels les mêmes opérations peuvent être réalisées. Chaque poste correspond à un exemplaire différent de même ressource. Si une seule cuve contient le bain chimique "Cu électrolytique" et que 3 produits peuvent être immergés dans cette cuve, nous considérerons que nous avons 3 exemplaires de la ressource de type "Cu électrolytique".

Le coût d'utilisation d'un exemplaire de la ressource de type r pendant une unité de temps est noté s_r, $s_r \in \Re^+$. Par conséquent, le coût induit par l'opération i effectuée sur le produit j par unité de temps est : $s_{i,j} = \sum_{r \in R} q_{r,i,j} \; s_r$. Ce coût correspond à la rémunération des opérateurs, à l'usure des outils, à l'amortissement des machines, etc.

Hypothèse II- 10 : Les disponibilités des ressources sont connues.

Les ressources peuvent être indisponibles :

- soit parce qu'elles ont été réservées pour des opérations de fabrication de produits préalablement ordonnancés,
- soit pour des raisons extérieures à la fabrication proprement dite (maintenance des machines, pauses légales des opérateurs, etc).

Pour chaque type de ressource r, où $r \in R$, nous connaissons les périodes où elle est utilisée, et en combien d'exemplaires. A partir des disponibilités de chaque ressource restant après les $(j-1)$ premiers produits ordonnancés, nous déduisons les fenêtres de temps durant lesquelles chaque opération i peut être exécutée sur le produit j, c'est-à-dire les périodes durant lesquelles le nombre de ressources de type r nécessaires pour exécuter l'opération i sur le produit j est supérieure ou égale à $q_{r,i,j}$. Les structures des données et les algorithmes permettant de gérer les disponibilités des ressources sont décrits dans l'Annexe 1.

Dans la suite, nous supposons défini l'ensemble $W_{i,j}$ des $y_{i,j}$ fenêtres de temps où l'opération i peut être effectuée sur le produit j : $W_{i,j} = \{[a_{z,i,j} \; ; \; b_{z,i,j}] \; / \; 1 \leq z \leq y_{i,j}\}$. Les dates de début et de fin de la z-ième période pendant laquelle l'opération i peut être exécutée sur le produit j

sont notées respectivement $a_{z,i,j}$ et $b_{z,i,j}$. On a : $a_{z,i,j} \in \Re^+$, $b_{z,i,j} \in \Re^+ \cup \{+\infty\}$ et $a_{z,i,j} < b_{z,i,j}$, pour $1 \leq z \leq y_{i,j}$, $i \in I_j$, $j \in J$.

L'exemple de la Figure II- 5 concerne le cas d'une opération qui nécessite une ressource d'un certain type r, $q_{r,i,j}=1$. Il existe trois exemplaires de cette machine : X1, X2 et X3. Les disponibilités de chaque exemplaire sont données par la Figure II- 5 : les plages grisées correspondent à des périodes où la ressource est déjà utilisée. L'instant 0 est supposé être l'instant initial. Nous avons, dans cet exemple, un ensemble de $y_{i,j}=7$ fenêtres différentes, $W_{i,j}=\{[0;1], [2;6], [3;7], [4;9], [8; +\infty[, [10; +\infty[, [11; +\infty[\}$.

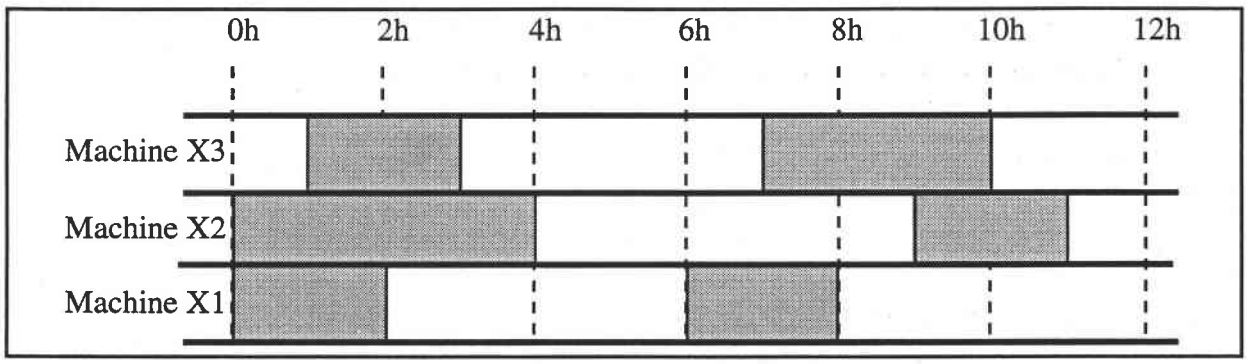

Figure II- 5 : Périodes d'exécution d'une opération

Hypothèse II- 11 : Les opérations se suivent sans interruption.

Dans la suite, les problèmes traités sont mixtes, c'est-à-dire sans attente et à durées opératoires contrôlables, suivant la terminologie introduite au Chapitre I. Comme nous l'avons vu, ce type de contraintes permet de modéliser **une très large variété de problèmes d'ordonnancement, et même des problèmes pour lesquels l'attente est permise.**

La contrainte "sans attente" est très importante dans les ateliers de traitement de surface. En effet, tout arrêt en cours de traitement met le résultat du traitement en péril.

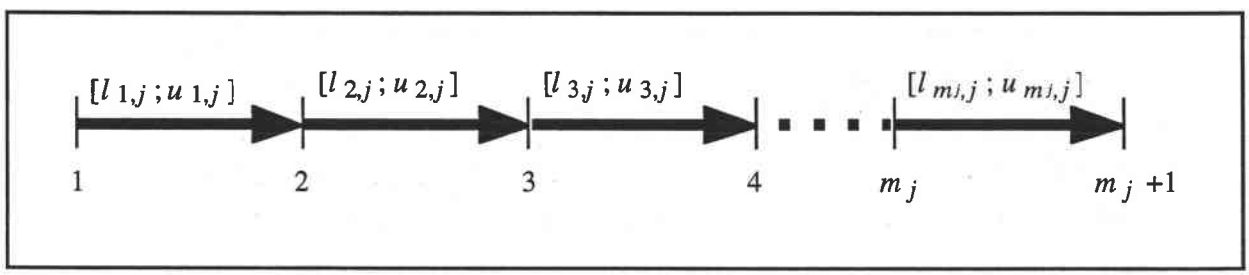

Figure II- 6 : Graphe représentant un produit

Nous représentons les données relatives à un produit j par un graphe valué $G_j=(H_j,I_j)$, comme celui de la Figure II- 6. Ce graphe est composé d'un ensemble H_j de (m_j+1) sommets marquant les instants de début et de fin de chaque opération. Ces sommets sont reliés par m_j

arcs représentant l'ensemble I_j des opérations à exécuter. Puisque les opérations se suivent sans attente, l'instant de fin d'une opération i est également l'instant de début de la suivante, pour $1 \leq i < m_j$. Une opération peut être référencée soit par $i \in I_j$, soit par $(h, h+1)=i$ où $h \in H_j$. Chaque opération est caractérisée par l'intervalle $[l_{i,j} ; u_{i,j}]$ des durées opératoires admissibles.

Hypothèse II- 12 : Les durées des opérations sont contrôlables.

La durée $P_{i,j}$ de l'opération i peut être choisie dans un intervalle de valeurs admissibles : $P_{i,j} \in [l_{i,j} ; u_{i,j}]$ avec $l_{i,j} \leq u_{i,j}$. Déterminer cette durée $P_{i,j}$ constitue la spécificité des problèmes mixtes. La durée minimale (en anglais *lower duration*) du temps opératoire est positive : $l_{i,j} \in \Re^+$. La durée maximale (en anglais *upper duration*) peut être illimitée : $u_{i,j} \in \Re^+ \cup \{+\infty\}$. $\Re^+$ désigne l'ensemble des valeurs réelles positives. Pour nous, une opération de durée nulle qui requiert une ressource n'équivaut pas à l'absence d'opération : cette opération doit être ordonnancée, ce qui influence le résultat global.

Le traitement de surface autorise, dans une certaine mesure, d'allonger des opérations chimiques ou électrolytiques sans détériorer le résultat. Le Tableau II- 1 donne les durées opératoires d'un produit j dont la gamme comporte 5 opérations. Cet exemple est repris pour illustrer les algorithmes présentés dans les sections suivantes. Cette tolérance sur la durée des opérations permet de compenser, dans une certaine limite, l'absence de stockage pour les produits en cours de traitement. Nous supposons que les cuves contenant un même bain sont regroupées et que, par conséquent, les durées de déplacement du robot entre les cuves contenant deux bains différents sont fixées. Dans le cas où cette hypothèse n'est pas vérifiée, nous utilisons des gammes alternatives (voir Chapitre IV).

Tableau II- 1 : Durées opératoires d'un produit

Numéro de l'opération i	Type d'opération	Durée minimale $l_{i,j}$	Durée maximale $u_{i,j}$
1	Transport par robot	1	$+\infty$
2	Traitement dans solution 1	3	4
3	Transport par robot	1	1
4	Traitement dans solution 2	2	4
m_j=5	Transport par robot	1	$+\infty$

Hypothèse II- 13 : La préemption est interdite.

Nous ne nous intéressons pas aux cas où les opérations peuvent être interrompues pendant un laps de temps puis reprises (ou en anglais *preemption*). Une adaptation possible de ce type de problèmes est présentée dans les extensions du chapitre suivant.

Hypothèse II- 14 : L'objectif est d'ordonnancer chaque produit dès que sa commande est enregistrée de manière à maximiser la productivité du système à long terme.

La date de début de l'opération i effectuée sur le produit j est notée $B_{i,j}$ et sa date de fin $C_{i,j}$. Puisque les opérations s'exécutent sans attente, $B_{i+1,j}=C_{i,j}$ pour tout $i \in I_j \setminus \{m_j\}$, $j \in J$. Dans ce chapitre, ces (m_j+1) instants de début et de fin de chaque opération sont numérotés chronologiquement de 1 à (m_j+1). Nous identifions l'indice du début de l'opération à l'indice de l'opération elle-même. On note $T_{h,j}$ les dates correspondants à ces (m_j+1) instants, pour $h \in H_j$. On a : $T_{i,j}=B_{i,j}$ et $T_{i+1,j}=C_{i,j}$ pour $i \in I_j$. **Ordonnancer un produit revient à déterminer durant quelle période $Z_{i,j}$ doit être exécutée chaque opération i sur le produit j, ainsi que ses dates de début $T_{i,j}$ et de fin $T_{i+1,j}=T_{i,j}+P_{i,j}$.**

Pour atteindre une productivité maximale du système à long terme, nous proposons, tout d'abord, de **terminer chaque produit le plus tôt possible**. Ceci signifie que nous cherchons à minimiser le *makespan* de chaque produit, i.e. la date $\max_{1 \le i \le m_j} C_{i,j}=C_{m_j,j}=T_{m_j+1,j}$.

C'est ce que nous faisons dans la section suivante. En outre, cette méthode de gestion permet de fournir en temps réel un délai de livraison précis et réalisable.

Ensuite, nous donnons une deuxième méthode de gestion qui revient **à fabriquer le produit au plus tard**, tout en respectant la date de livraison fixée au plus tôt (voir section 4). L'intérêt est de réduire le temps pendant lequel des ressources sont utilisées pour fabriquer le produit. L'effet escompté est double : réduire le coût de fabrication du produit, et donner plus de liberté pour ordonnancer les produits futurs. Cette méthode doit permettre d'atteindre une meilleure productivité à long terme.

Enfin, nous donnons une troisième méthode de gestion dont le but est de **minimiser le coût d'utilisation des ressources de chaque produit**, tout en respectant la date de livraison fixée au plus tôt (voir section 5). Nous montrons comment utiliser cette méthode pour diminuer l'utilisation des ressources goulots et ainsi espérer accroître la productivité du système.

Le Tableau II- 2 reprend les notations introduites et donne pour chacune son intervalle de définition. Etant donné que le produit j est ordonnancé sans remettre en cause les produits déjà ordonnancés (voir Hypothèse II- 3), les données concernant les produits sont figées. Aussi, l'indice j qui apparaît dans les notations est facultatif. Il est cependant conservé pour que les notations restent cohérentes tout au long de la thèse.

Tableau II- 2 : Récapitulatif des notations

Nom	Désignation	Type	Intervalle de définition
$a_{z,i,j}$	Date de début de la z-ième période pendant laquelle l'opération i peut être exécutée sur le produit j.	Valeur donnée	$a_{z,i,j} \in \Re^+$, $a_{z,i,j} < b_{z,i,j}$, pour $1 \le z \le y_{i,j}$, $i \in I_j$, $j \in J$.
$b_{z,i,j}$	Date de fin de la z-ième période pendant laquelle l'opération i peut être exécutée sur le produit j.	Valeur donnée	$b_{z,i,j} \in \Re^+ \cup \{+\infty\}$, $a_{z,i,j} < b_{z,i,j}$, pour $1 \le z \le y_{i,j}$, $i \in I_j$, $j \in J$.
$B_{i,j}$	Date de début de l'opération i effectuée sur le produit j.	Variable de décision	$B_{i,j} \in \Re^+$, $B_{i+1,j} = C_{i,j}$, $C_{i,j} = B_{i,j} + P_{i,j}$, pour $i \in I_j$, $j \in J$.
$C_{i,j}$	Date de fin de l'opération i effectuée sur le produit j.	Variable de décision	$C_{i,j} \in \Re^+$, $B_{i+1,j} = C_{i,j}$, $C_{i,j} = B_{i,j} + P_{i,j}$, pour $i \in I_j$, $j \in J$.
d_j	Date avant laquelle le produit doit être livré.	Valeur donnée	$d_j \in \Re^+ \cup \{+\infty\}$, pour $j \in J$.
G_j	Graphe valué du produit j.	Graphe donné	$G_j = (H_j, I_j)$, pour $j \in J$.
H_j	Ensemble des (m_j+1) instants h de début et de fin des opérations sur le produit j.	Ensemble donné	$H_j = \{h \ / \ 1 \le h \le m_j + 1\}$ pour $j \in J$.
I_j	Ensemble des m_j opérations i à effectuer sur le produit j.	Ensemble donné	$I_j = \{i \ / \ 1 \le i \le m_j\}$ pour $j \in J$.
J	Ensemble des produits j à ordonnancer, numérotés de 1 à n.	Ensemble donné	$J = \{j \ / \ 1 \le j \le n\}$.
$l_{i,j}$	Durée minimale admissible de l'opération i effectuée sur le produit j.	Valeur donnée	$l_{i,j} \in \Re^+$, $l_{i,j} \le u_{i,j}$, pour $i \in I_j$, $j \in J$.
m_j	Nombre d'opérations à effectuer sur le produit j.	Nombre donné	$m_j = \lvert I_j \rvert$, pour $j \in J$.
n	Nombre de produits à ordonnancer.	Nombre donné	$n = \lvert J \rvert$.
$P_{i,j}$	Durée de l'opération i effectuée sur le produit j.	Variable de décision	$P_{i,j} \in \Re^+$, $P_{i,j} \in [l_{i,j} \ ; \ u_{i,j}]$, pour $i \in I_j$, $j \in J$.

$q_{r,i,j}$	Nombre d'unités de ressources de type r nécessaires pour exécuter l'opération i sur le produit j.	Valeur donnée	$q_{r,i,j} \geq 0$, pour $r \in R$, $i \in I_j$, $j \in J$.
r_j	Date à partir de laquelle le produit peut commencer à être fabriqué.	Valeur donnée	$r_j \in \Re^+$, pour $j \in J$.
R	Ensemble des types de ressources. Il regroupe les ressources de R_{in} dont les exemplaires sont interchangeables en cours d'opération, et celles de R_{en}.	Ensemble donné	$R = \{r \ / \ 1 \leq r \leq \lvert R \rvert\}$, $R = R_{in} \cup R_{en}$.
s_r	Coût d'utilisation d'un exemplaire de la ressource de type r par unité de temps.	Valeur donnée	$s_r \in \Re^+$, pour $r \in R$.
$s_{i,j}$	Coût induit par l'opération i effectuée sur le produit j par unité de temps.	Valeur donnée	$s_{i,j} \in \Re^+$, $s_{i,j} = \sum_{r \in R} q_{r,i,j} \ s_r$, pour $i \in I_j$, $j \in J$.
$T_{h,j}$	Date des instants h de début et de fin des opérations effectuées sur le produit j.	Variable de décision	$T_{h,j} \in \Re^+$, pour $h \in H_j$, $j \in J$. $T_{h,j} = B_{(h,h+1),j}$, $T_{h+1,j} = C_{(h,h+1),j}$, pour $i = (h, h+1) \in I_j$, $j \in J$.
$u_{i,j}$	Durée maximale admissible de l'opération i effectuée sur le produit j.	Valeur donnée	$u_{i,j} \in \Re^+ \cup \{+\infty\}$, $l_{i,j} \leq u_{i,j}$, pour $i \in I_j$, $j \in J$.
$W_{i,j}$	Ensemble des $y_{i,i}$ fenêtres de temps durant lesquelles l'opération i peut être exécutée sur le produit j.	Ensemble donné	$W_{i,j} = \{[a_{z,i,j} \ ; \ b_{z,i,j}] / \ 1 \leq z \leq y_{i,j}\}$, pour $i \in I_j$, $j \in J$.
$y_{i,j}$	Nombre de périodes pendant lesquelles l'opération i peut être exécutée sur le produit j.	Nombre donné	$y_{i,j} = \lvert W_{i,j} \rvert$, pour $i \in I_j$, $j \in J$.
y_j	Nombre de périodes relatives aux opérations effectuées sur le produit j.	Nombre donné	$y_j = \sum_{1 \leq i \leq m_j} y_{i,j}$, pour $j \in J$.
$Z_{i,j}$	Indice de la période pendant laquelle l'opération i est exécutée sur le produit j.	Variable de décision	$1 \leq Z_{i,j} \leq y_{i,j}$, pour $i \in I_j$, $j \in J$.

$\lvert E \rvert$ représente le cardinal de l'ensemble E.

$\Re^+$ est l'ensemble des valeurs réelles positives.

II.3. MINIMISATION DU *MAKESPAN*

II.3.1. Introduction

Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit au plus tôt. Ceci signifie que nous cherchons à minimiser le *makespan* de chaque produit. D'autre part, cette méthode de gestion permet de fournir, au moment où une commande arrive, un délai minimum de livraison réalisable et précis.

Yin et Yih (1992) ont étudié ce problème dans un cas particulier de type *flow shop*. Ces auteurs n'utilisent pas la tolérance des durées opératoires que sur les tâches en conflit (i.e. utilisant la ressource alors qu'elle n'est pas encore disponible) et ainsi n'assurent pas de commencer le plus tôt possible.

Pour déterminer le délai minimum de livraison, nous définissons les dates d'exécution au plus tôt des opérations compatibles avec leurs durées opératoires admissibles et les fenêtres de temps dans lesquelles les opérations peuvent être exécutées. Après avoir formalisé le problème, nous proposons un algorithme original de très bonne complexité.

II.3.2. Formalisation

Nous voulons déterminer quand exécuter les opérations sur le produit j sachant que la date de disponibilité est r_j (voir contraintes (II-1)). Le produit j est composé d'un ensemble I_j de m_j opérations s'exécutant successivement sans attente. Ainsi, $T_{i,j}$ et $T_{i+1,j}$ définissent les instants de début et de fin de chaque opération $i \in I_j$. La durée d'exécution $P_{i,j}=T_{i+1,j}-T_{i,j}$ de chaque opération i est à déterminer et doit vérifier : $l_{i,j} \leq P_{i,j} \leq u_{i,j}$ (voir contraintes (II-2)). Chaque opération i peut être exécutée dans l'une des $y_{i,j}$ fenêtres de temps de l'ensemble $W_{i,j}=\{[a_{z,i,j}\ ;\ b_{z,i,j}]\ /\ 1 \leq z \leq y_{i,j}\}$ (voir contraintes (II-3)-(II-4)). La fenêtre retenue est la fenêtre d'indice $Z_{i,j}$ (voir contraintes (II-5)). Notre objectif est de minimiser la date $T_{m_j+1,j}$ d'achèvement du produit. Les contraintes (II-6) et (II-7) donnent les intervalles de définition de toutes les variables de décisions $T_{i,j}$ et $Z_{i,j}$. $\Re^+$ et N désignent respectivement les ensembles des nombres réels et des entiers naturels. Formellement, le problème à résoudre s'écrit :

Problème II- 1 : Minimisation du *makespan* de chaque produit

$$\min \quad T_{m_j+1,j}$$

$$\text{tel que} \begin{cases} r_j \le T_{1,j} & & (\text{II-1}) \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & (\text{II-2}) \\ a_{Z_{i,j},i,j} \le T_{i,j} & \forall 1 \le i \le m_j & (\text{II-3}) \\ T_{i+1,j} \le b_{Z_{i,j},i,j} & \forall 1 \le i \le m_j & (\text{II-4}) \\ 1 \le Z_{i,j} \le y_{i,j} & \forall 1 \le i \le m_j & (\text{II-5}) \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & (\text{II-6}) \\ Z_{i,j} \in \mathrm{N} & \forall 1 \le i \le m_j & (\text{II-7}) \end{cases}$$

Ce problème peut également se formaliser comme un problème linéaire mixte :

Problème II- 2 : Minimisation du *makespan* de chaque produit

$$\min \quad T_{m_j+1,j}$$

$$\text{tel que} \begin{cases} r_j \le T_{1,j} & & & (\text{II-8}) \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & & (\text{II-9}) \\ a_{z,i,j} \le T_{i,j} + (1-\lambda_{z,i,j})M & \forall 1 \le z \le y_{i,j} & \forall 1 \le i \le m_j & (\text{II-10}) \\ T_{i+1,j} \le b_{z,i,j} + (1-\lambda_{z,i,j})M & \forall 1 \le z \le y_{i,j} & \forall 1 \le i \le m_j & (\text{II-11}) \\ \sum_{1 \le z \le y_{i,j}} \lambda_{z,i,j} = 1 & \forall 1 \le i \le m_j & & (\text{II-12}) \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & & (\text{II-13}) \\ \lambda_{z,i,j} \in \{0,1\} & \forall 1 \le z \le y_{i,j} & \forall 1 \le i \le m_j & (\text{II-14}) \end{cases}$$

Les contraintes (II-8)-(II-9) sont équivalentes à (II-1)-(II-2). Dans cette formulation, chaque variable $\lambda_{z,i,j}$ est booléenne. Elle vaut 1 si la fenêtre z est choisie pour exécuter l'opération i sur le produit j, et vaut 0 sinon. Les contraintes (II-10) à (II-11) assurent que chaque opération i effectuée sur le produit j dans la fenêtre z pour laquelle $\lambda_{z,i,j}$=1. La constante M est une valeur arbitrairement grande. On peut choisir, par exemple :

$$M = \max\left\{ r_j ; \max_{\substack{1 \le z \le y_{i,j} \\ 1 \le i \le m_j}} \left\{ \max\left\{ a_{z,i,j} ; b_{z,i,j} \ne +\infty \right\} \right\} \right\} + \sum_{1 \le i \le m_j} \max\left\{ l_{i,j} ; u_{i,j} \ne +\infty \right\}$$

où $\max\{a;b \ne +\infty\}$ vaut a si $b=+\infty$ et vaut $\max\{a;b\}$ sinon. Les égalités (II-12) garantissent qu'une seule fenêtre est choisie pour chaque opération. Les intervalles des variables de décision $T_{i,j}$ et $\lambda_{z,i,j}$ sont donnés par (II-13) et (II-14).

Plutôt que de résoudre directement le Problème II- 2 par une technique arborescente coûteuse en temps de calcul, nous proposons dans les deux sections suivantes une approche de résolution permettant de fournir une solution optimale à ce problème en un temps polynomial. Elle consiste à résoudre une suite de sous-problèmes décrits dans le paragraphe suivant.

II.3.3. Sélection des durées opératoires

L'objectif de cette section est de déterminer $T_{i,j}$ et $T_{i+1,j}$, dates de début et de fin d'exécution au plus tôt des opérations, si chaque opération i est exécutée sur le produit j dans la fenêtre donnée $Z_{i,j}$, $i \in I_j$. Nous ne tenons pas compte, dans cette formulation, de la date de fin de la fenêtre $b_{z_{i,j},i,j}$, i.e. la contrainte (II-4). Etant donné que les opérations se suivent sans attente, $(T_{i+1,j}-T_{i,j})$ représente la durée effective définie pour l'opération i. Le problème s'écrit :

Problème II- 3 : Définition des dates au plus tôt des opérations quand leur fenêtre d'exécution est donnée

$$\min \sum_{1 \le i \le m_j+1} \mu_i T_{i,j}$$

$$\text{tel que} \begin{cases} r_j \le T_{1,j} & (\text{II-1}) \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} \quad \forall 1 \le i \le m_j & (\text{II-2}) \\ a_{z_{i,j},i,j} \le T_{i,j} \quad \forall 1 \le i \le m_j & (\text{II-3}) \\ T_{i,j} \in \Re^+ \quad \forall 1 \le i \le m_j+1 & (\text{II-6}) \end{cases}$$

Dans ce problème, l'ensemble $\{\mu_i \ / \ 1 \le i \le m_j+1\}$ est un ensemble arbitraire de valeurs positives. La solution optimale du Problème II- 3 pour l'ensemble $\{\mu_{i_0}=1\} \cup \{\mu_i=0 / 1 \le i \le m_j+1,\ i \ne i_0\}$ correspond au cas où la date T_{i_0} est calée au plus tôt. Si la solution optimale de ce problème est unique quel que soit l'ensemble $\{\mu_i \ / \ 1 \le i \le m_j+1\}$ alors cette solution correspond au cas où chaque date est calée au plus tôt. C'est une telle solution que nous recherchons ici. Si, de plus, cette solution vérifie les contraintes (II- 4) (i.e. $T_{i+1,j} \le b_{z_{i,j},i,j}$, pour $1 \le i \le m_j$), alors cette solution est réalisable et calée au plus tôt pour cet ensemble de fenêtres $\{Z_{i,j}\}_{1 \le i \le m_j}$. Sinon, aucune solution n'est réalisable pour ces fenêtres.

En fait, dans le Problème II- 3, nous recherchons à placer le produit j comme nous définirions les longueurs des marches d'un escalier. Chaque marche correspond à une opération. La longueur des marches est alors ajustable dans une certaine limite de la même manière que les durées opératoires sont contrôlables. De plus, le début de chaque marche i doit être au moins distant de l'origine d'une longueur correspondant à la date $a_{z_{i,j},i,j}$. L'objectif revient alors à trouver l'escalier le plus court. L'Algorithme II- 1 permet de résoudre le Problème II- 3 par une méthode qui se déroule en deux étapes. Tout se passe comme si dans la première étape, nous définissions une borne inférieure τ_i de la longueur qui sépare le début de chaque marche i de l'origine en tenant compte de $a_{z_{i,j},i,j}$ et de la somme des longueurs minimales des marches précédentes. Dans la seconde étape, nous fixons la longueur de chaque marche i en prenant en considération τ_i et les longueurs maximales des marches suivantes. Plus précisément, dans la première étape de l'Algorithme II- 1, on définit τ_i, borne inférieure des dates de début d'exécution de chaque opération i, en tenant compte des dates de début des périodes et des durées minimales des opérations. Dans la seconde étape, on raffine cette borne en tenant compte des durées maximales des opérations. Cette borne est alors la date optimale.

Algorithme II- 1 : Calage au plus tôt des opérations sur un produit [Chauvet et al., 1999a]

Procédure Définition des dates au plus tôt d'exécution des opérations effectuées sur le produit j.

Entrée : j est l'indice du produit considéré.

r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.

m_j est le nombre d'opérations à effectuer sur le produit j.

I_j est l'ensemble des opérations à effectuer sur le produit j.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$Z_{i,j}$ est la fenêtre dans laquelle on veut réaliser l'opération i sur le produit j, pour $i \in I_j$.

$a_{z_{i,j},i,j}$ est la date de début de la période durant laquelle on réalise l'opération i, pour $i \in I_j$.

Sortie : $T_{i,j}$ est la date de début au plus tôt de l'opération i effectuée sur le produit j, où $i \in I_j$.

$T_{m_j+1,j}$ est la date de fin au plus tôt du produit j.

1. **Etape 1.**

1.1. **Poser** $\tau_1 := \max\{r_j \ ; \ a_{z_{1,j},1,j}\}$. *On note τ_i la borne inférieure de la date de début d'exécution de l'opération i.*

1.2. **Pour** i allant de 2 à m_j : *On note i l'indice de l'opération sélectionnée.*

1.2.1. **Poser** $\tau_i := \max\{\tau_{i-1}+l_{i-1,j} \ ; \ a_{z_{i,j},i,j}\}$.

2. **Etape 2.**

2.1. **Poser** $T_{m_j+1,j} := \tau_{m_j}+l_{m_j,j}$.

2.2. **Pour** i allant de m_j à 1 : *On note i l'indice de l'opération sélectionnée.*

2.2.1. **Poser** $T_{i,j} := \max\{T_{i+1,j}-u_{i,j} \ ; \ \tau_i\}$.

Exemple

Tableau II- 3 : Données relatives à l'exécution de l'Algorithme II- 1

ENTREES				RESULTATS INTERMEDIAIRES	SORTIES
Indice de l'opération i	Durée minimale $l_{i,j}$	Durée maximale $u_{i,j}$	Date de début de la période durant laquelle on réalise l'opération i $a_{z_{i,j},i,j}$	Borne inférieure de la date au plus tôt τ_i	Date au plus tôt $T_{i,j}$
1	1	$+\infty$	2	2	2
2	3	4	0	3	3
3	1	1	5	6	6
4	2	4	7	7	7
m_j=5	1	$+\infty$	11	11	11
m_j+1=6					12

Nous proposons un exemple d'application de l'Algorithme II- 1. Les données relatives au produit j à réaliser sont fournies dans le Tableau II- 3 et reprises sur la Figure II- 7. Le produit peut être réalisé à partir de la date r_j=0.

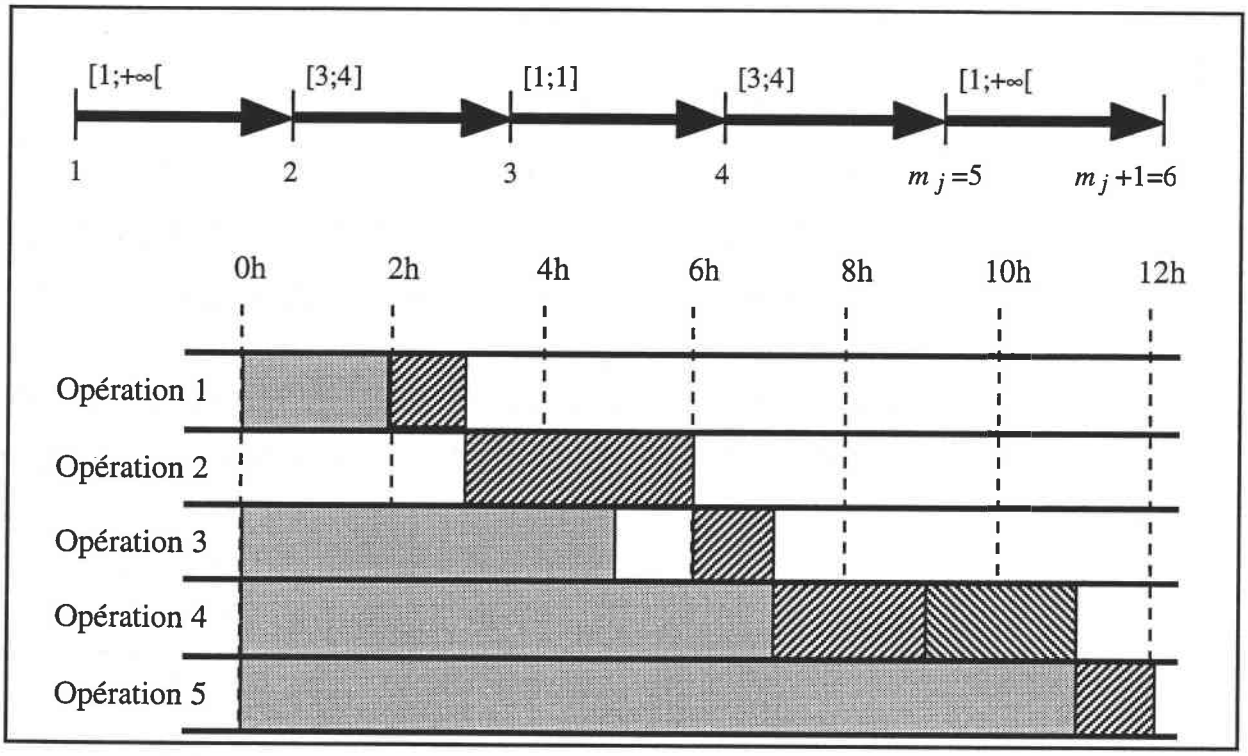

Figure II- 7 : Application à l'Algorithme II- 1

Les résultats de l'exécution sont explicités sur la Figure II- 7. Les opérations effectuées sur le produit j à ordonnancer sont représentées par des zones hachurées. Dans le cas où la durée d'une opération dépasse la durée minimale, on utilise des hachures différentes pour la partie représentant le dépassement : c'est le cas de l'opération 4 dans l'exemple précédent.

Théorème II- 1 [Chauvet et *al.*, 1999a]

L'Algorithme II- 1 détermine les dates de début et de fin d'exécution au plus tôt de toutes les opérations, i.e. il fournit une solution unique optimale au Problème II- 3.

Démonstration

a) Nous montrons tout d'abord que la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournie par l'Algorithme II- 1 est réalisable.

Suivant les pas 1.1 et 2.2.1 de l'algorithme, on a $r_j \le \tau_1 \le T_{1,j}$. Par conséquent, la solution vérifie l'inéquation (II-1).

On déduit du pas 2.2.1 de l'algorithme que $T_{i,j}+l_{i,j}=\max\{T_{i+1,j}-(u_{i,j}-l_{i,j})\ ;\ \tau_i+l_{i,j}\}$, pour $1\le i\le m_j$. Puisque, par hypothèse, $l_{i,j}\le u_{i,j}$, on a $T_{i,j}+l_{i,j}\ge\max\{T_{i+1,j}\ ;\ \tau_i+l_{i,j}\}$.

Comme $\tau_i+l_{i,j}\leq\tau_{i+1}\leq T_{i+1,j}$, pour $1\leq i<m_j$ (cf. pas 1.2.1 et 2.2.1), on obtient $T_{i,j}+l_{i,j}\geq\max\{T_{i+1,j}\ ;\ T_{i+1,j}\}$. Considérant le pas 2.1, on en déduit $T_{i,j}+l_{i,j}\geq T_{i+1,j}$, pour $1\leq i\leq m_j$. De plus, selon le pas 2.2.1 de l'algorithme, $T_{i,j}\leq T_{i+1,j}-u_{i,j}$, pour $1\leq i\leq m_j$. On peut conclure que $l_{i,j}\leq T_{i+1,j}-T_{i,j}\leq u_{i,j}$, pour $1\leq i\leq m_j$. Donc, la solution satisfait les contraintes (II-2).

Suivant les pas 1.1, 1.2.1 et 2.2.1 de l'algorithme, on a $a_{z_{i,j},i,j}\leq\tau_i\leq T_{i,j}$, pour $1\leq i\leq m_j$. Par conséquent, la solution vérifie les inéquations (II-3).

La solution fournie est réalisable car elle satisfait les contraintes (II-1), (II-2), (II-3) et (II-6).

b) Nous montrons ensuite que la solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ fournie par l'Algorithme II- 1 est optimale.

Selon les contraintes (II-1) et (II-3), la valeur $\max\{r_j\ ;\ a_{z_{1,j},1,j}\}$ est une borne inférieure de la date de début de la première opération. D'après le pas 1.1 de l'Algorithme II- 1, τ_1 est donc une borne inférieure de cette date.

Si τ_{i-1} est une borne inférieure de la date de début de l'opération $i-1$, pour $m_j\geq i>1$, selon les contraintes (II-2) et (II-3), la valeur $\max\{\tau_{i-1}+l_{i-1,j}\ ;\ a_{z_{i,j},i,j}\}$ est une borne inférieure de la date de début de l'opération i. D'après le pas 1.2.1 de l'Algorithme II- 1, τ_i est donc une borne inférieure de cette date, pour $m_j\geq i\geq 1$.

Selon la contrainte (II-2), la valeur $\tau_{m_j}+l_{m_j,j}$ est une borne inférieure de la date de fin de la dernière opération. D'après le pas 2.1 de l'Algorithme II- 1, $T_{m_j+1,j}$ est donc une borne inférieure de cette date.

Si $T_{i+1,j}$ et τ_i sont des bornes inférieures des dates respectivement de début et de fin de l'opération $i+1$, pour $m_j\geq i\geq 1$, selon la contrainte (II-2), la valeur $\max\{T_{i+1,j}-u_{i,j}\ ;\ \tau_i\}$ est une borne inférieure de la date de début de l'opération i. D'après le pas 2.2.1 de l'Algorithme II- 1, $T_{i,j}$ est donc une borne inférieure de cette date, pour $m_j\geq i\geq 1$.

Chaque date $T_{i+1,j}$ fournie par l'Algorithme II- 1, est une borne inférieure de la date de fin de l'opération i, pour $m_j\geq i\geq 1$. On en conclut que l'Algorithme II- 1 détermine les dates d'exécution au plus tôt de toutes les opérations. Par conséquent, il fournit la solution optimale au Problème II- 3 quel que soit l'ensemble $\{\mu_i\ /\ 1\leq i\leq m_j+1\}$ de valeurs strictement positives.

CQFD

Complexité

Dans tous les cas, l'Algorithme II- 1 nécessite $2m_j$ comparaisons (effectuées aux pas 1.1, 1.2.1 et 2.2.1) et $2m_j+1$ affectations (effectuées aux pas 1.1, 1.2.1, 2.1 et 2.2.1), où m_j est le nombre d'opérations effectuées sur le produit j à ordonnancer. La complexité de l'Algorithme

II- 1 est en $O(m_j)$. Tout algorithme doit au moins examiner chacune des opérations pour résoudre le problème. Par conséquent, aucun algorithme ne peut avoir une complexité inférieure à $O(m_j)$, c'est-à-dire inférieure à celle de l'Algorithme II- 1.

Remarque

Le Problème II- 3 peut être représenté sous forme d'un graphe "potentiels étapes" [Roy, 1960], ou d'un graphe PERT [Malcom et *al.*, 1959], comme sur la Figure II- 8. Dans cette représentation, la valuation $\theta_{(h,h')}$ de chaque arc (h,h') donne la durée minimale qui doit séparer les dates T_h et $T_{h'}$ correspondant aux instants h et h'. On a : $T_{h'}-T_h \geq \theta_{(h,h')}$. Ainsi, pour s'assurer que chaque opération i commence après la date $a_{Z_{i,j},i,j}$, un arc relie l'instant initial, noté 0, au début de l'opération i. De plus, à chaque limite minimale $l_{i,j}$ de la durée de l'opération i effectuée sur le produit j correspond un arc séparant le début de l'opération i de sa fin valué par $l_{i,j}$. Enfin, pour représenter la limite maximale $u_{i,j}$ de la durée de chaque opération, on introduit des arcs "retour" portant une valeur négative. Pour chaque opération i, un arc valué par $-u_{i,j}$ et reliant la fin de l'opération à son début est défini. Mais de tels arcs créent des circuits dans le graphe PERT si bien que des algorithmes spécifiques [Ford, 1956] [Bellman, 1958] [Gondran et Minoux, 1979] doivent être utilisés. La complexité de ces algorithmes en $O(m_j^2)$ est moins bonne que celle de l'Algorithme II- 1.

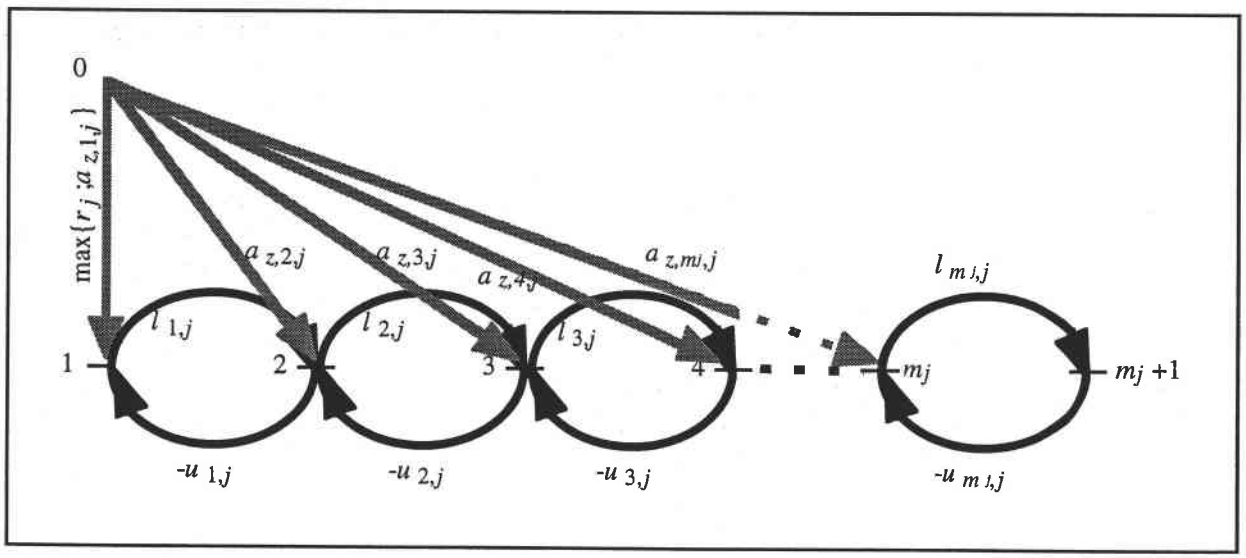

Figure II- 8 : Représentation du problème sous forme de graphe PERT

Dans le cas où l'attente est permise entre les opérations et les en-cours non limités, les durées sont fixées à la limite minimale $l_{i,j}$. Les arcs "retour" ne sont plus indispensables sur le graphe PERT. Ainsi, un algorithme classique de plus long chemin [Gondran et Minoux, 1979] permet de trouver la solution optimale. On peut également utiliser l'Algorithme II- 1 en modélisant les attentes comme des opérations ne requérant aucune ressource, de durée minimale égale à 0 et non limitée en durée. Les deux approches conduisent à des algorithmes en $O(m_j)$.

II.3.4. Sélection des fenêtres

L'objectif de cette section est de résoudre le Problème II- 1 et son équivalent, le Problème II- 2. Autrement dit, nous allons déterminer dans quelle fenêtre $Z_{i,j}$, l'opération i doit être effectuée de manière à ce que le produit j soit terminé le plus tôt possible. Nous utilisons les résultats de la section précédente qui permettent, pour un ensemble de fenêtres $\{Z_{i,j}\}_{1 \le i \le m_j}$, de déterminer les dates au plus tôt $\{T_{i,j}\}_{1 \le i \le m_j+1}$ d'exécution des opérations effectuées sur le produit j. Dans le cas où ces dates ne vérifient pas les contraintes (II- 4) (i.e. $T_{i+1,j} \le b_{z_{i,j},i,j}$, pour $1 \le i \le m_j$), alors aucune solution réalisable n'existe pour ces fenêtres.

Algorithme II- 2 : Choix des fenêtres pour une fin au plus tôt [Chauvet et al., 1999a]

Procédure Définition des fenêtres permettant d'exécuter les opérations au plus tôt sur le produit j.
Entrée : j est l'indice du produit considéré.
r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.
m_j est le nombre d'opérations à réaliser sur le produit j.
I_j est l'ensemble des opérations à réaliser sur le produit j.
$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.
$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.
$y_{i,j}$ est le nombre de fenêtres dans laquelle on peut réaliser l'opération i, pour $i \in I_j$.
$a_{z,i,j}$ est la date de début de la z-ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \le z \le y_{i,j}$, $i \in I_j$.
$b_{z,i,j}$ est la date de fin de la z-ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \le z \le y_{i,j}$, $i \in I_j$.
Sortie : $T_{i,j}$ est la date de début au plus tôt de l'opération i effectuée sur le produit j, où $i \in I_j$.
$T_{m_j+1,j}$ est la date de fin au plus tôt du produit j.
$Z_{i,j}$ est la fenêtre dans laquelle on doit réaliser l'opération i effectuée sur le produit j, où $i \in I_j$.

1. **Initialisation.**
   1.1. **Trier** les fenêtres $[a_{z,i,j}\ ;\ b_{z,i,j}]$ pour $1 \le z \le y_{i,j}$, $i \in I_j$,
   dans l'ordre croissant des $a_{z,i,j}$ et, en cas d'égalité, dans l'ordre croissant des $b_{z,i,j}$.
   Dans la suite de l'algorithme, la z-ième fenêtre est le z-ième élément dans cet ordre. $1 \le z \le y_{i,j}$.
   1.2. **Poser** $Trouvé$:=0. *On note $Trouvé$ la variable qui vaut* 1 *si une solution réalisable est trouvée,*
   1.3. **Pour** $i \in I_j$: *et qui vaut* 0 *sinon.*
      1.3.1. **Poser** $Z_{i,j}$:=1.
2. **Tant que** ($Trouvé$=0) : *Aucune solution réalisable n'a été trouvée.*
   2.1. **Appliquer** l'Algorithme II- 1.
   Entrée : j, r_j, m_j, I_j, $\{l_{i,j}\}_i$, $\{u_{i,j}\}_i$, $\{Z_{i,j}\}_i$, $\{a_{z_{i,j},i,j}\}_{i \in I_j}$.
   Sortie : $\{T_{i,j}\}_{i \in I_j}$, $T_{m_j+1,j}$.
   2.2. **Si** ($T_{i+1,j} \le b_{z_{i,j},i,j}$, pour $i \in I_j$) **alors** : *Chaque opération i se termine avant $b_{z_{i,j},i,j}$ c'est-à-dire avant la fin de la fenêtre $Z_{i,j}$.*
      2.2.1. **Poser** $Trouvé$:=1. *Une solution réalisable a été trouvée.*
   2.3. **Sinon** :
      2.3.1. **Pour** $i \in I_j$: *On note i l'indice de l'opération examinée.*
         2.3.1.1. **Tant que** ($T_{i+1,j} > b_{z_{i,j},i,j}$) **alors** : *L'opération i ne peut être terminée dans la fenêtre $Z_{i,j}$.*
            2.3.1.1.1. **Poser** $Z_{i,j}$:=$Z_{i,j}$+1. *On sélectionne la fenêtre suivante.*
            2.3.1.1.2. **Si** ($Z_{i,j} > y_{i,j}$) **alors** :
               2.3.1.1.2.1. **Quitter**. *Il n'existe aucune solution réalisable.*

L'Algorithme II- 2 permet de résoudre le Problème II- 1 et le Problème II- 2. Pour cela, il examine les fenêtres relatives à chaque opération dans leur ordre chronologique sans qu'un retour arrière ne soit nécessaire. En effet, si une fenêtre ne permet pas de terminer l'opération au pas 2.3.1.1, tout nouvel ordonnancement où cette fenêtre serait utilisée, ne pourra se terminer plus tôt, c'est-à-dire avant la fin de la fenêtre. La fenêtre peut être ignorée.

Si l'Algorithme II- 2 s'achève au pas 2.3.1.1.2.1, la solution courante n'est pas réalisable et même, il n'existe aucune solution réalisable (comme il le sera démontré dans la suite). Cela signifie qu'il est impossible de fabriquer le produit avec les contraintes imposées au système de production. Ceci ne se produit jamais si, pour chaque opération i, il existe une période z de disponibilité non limitée, i.e. $b_{z,i,j}=+\infty$. Ce cas est le plus courant.

Exemple

Nous proposons un exemple d'application de l'Algorithme II- 2. Les données relatives au produit j à réaliser sont fournies dans le Tableau II- 4. Les durées opératoires considérées sont celles du Tableau II- 3. Le produit peut être réalisé à partir de la date $r_j=0$.

Tableau II- 4 : Données illustratives de l'Algorithme II- 2

ENTREES		RESULTATS INTERMEDIAIRES				SORTIES		
		Itération 1		Itération 2		Itération 3		
i	$\{[a_{z,i,j};b_{z,i,j}] / 1\le z\le y_{i,j}\}$	$[a_{z,i,j};b_{z,i,j}]$	$T_{i,j}$	$[a_{z,i,j};b_{z,i,j}]$	$T_{i,j}$	$[a_{z,i,j};b_{z,i,j}]$	$T_{i,j}$	$Z_{i,j}$
1	[0;1], [2;4], [5;9], [11;+∞[	[0;1]	0	[0;1]	0	[2;4]	0	2
2	[0;8], [11;+∞[	[0;8]	1	[0;8]	2	[0;8]	2	1
3	[0;1], [2;4], [5;9], [11;+∞[	[0;1]	4	[5;9]	6	[5;9]	6	3
4	[0;3], [7;+∞[	[0;3]	5	[7;+∞[	7	[7;+∞[	7	2
m_j=5	[0;1], [2;4], [5;9], [11;+∞[	[0;1]	7	[5;9]	9	[11;+∞[	11	4
m_j+1=6			8		10		12	

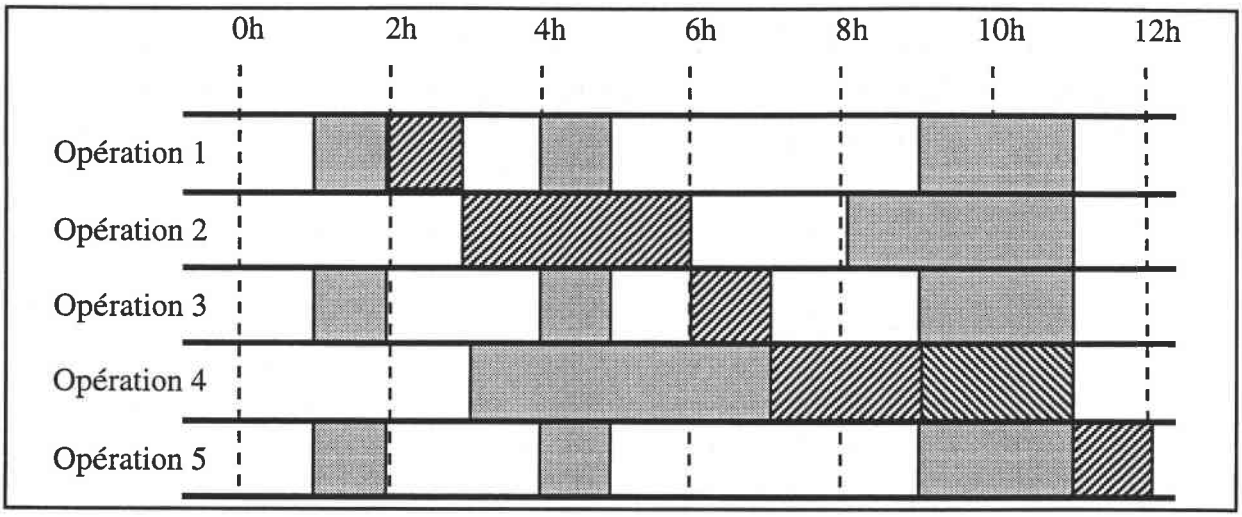

Figure II- 9 : Résultats fournis par l'Algorithme II- 2

Les résultats de l'exécution sont explicités sur la Figure II- 9 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Théorème II- 2 [Chauvet et *al.*, 1999a]

Si le Problème II- 1 a une solution réalisable, l'Algorithme II- 2 détermine les fenêtres et les dates d'exécution qui permettent de finir au plus tôt toutes les opérations effectuées sur le produit. Par conséquent, la solution fournie par l'Algorithme II- 2 est optimale pour le Problème II- 1.

Démonstration

a) Nous montrons tout d'abord que si l'Algorithme II- 2 fournit une solution $\{\{T_{i,j}\}_{1\le i\le m_j+1};\{Z_{i,j}\}_{1\le i\le m_j}\}$, elle est réalisable pour le Problème II- 1.

Les dates $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournies par l'Algorithme II- 2 sont déterminées au pas 2.1 par l'Algorithme II- 1 pour les fenêtres retenues $\{Z_{i,j}\}_{1\le i\le m_j}$. D'après le Théorème II- 1, cette solution vérifie les contraintes (II-1), (II-2), (II-3) et (II-6). Puisque, tout au long de l'Algorithme II- 2, la variable $Z_{i,j}$ prend des valeurs dans $\{1, 2, \ldots, y_{i,j}\}$, les contraintes (II-5) et (II-7) sont vérifiées par la solution. Au pas 2.2 de l'Algorithme II- 2, on s'assure que toutes les contraintes (II-4) (i.e. $T_{i+1,j}\le b_{Z_{i,j},i,j}$, pour $i\in I_j$) sont vérifiées avant de déclarer qu'une solution réalisable est trouvée. Par conséquent, la solution fournie par l'Algorithme II- 2 est réalisable.

b) Nous montrons ensuite que la première solution réalisable rencontrée par l'Algorithme II- 2 est optimale pour le Problème II- 1.

D'après le Théorème II- 1, pour les fenêtres $\{Z_{i,j}\}_{1\le i\le m_j}$, les dates $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournies par l'Algorithme II- 1 au pas 2.2 de l'Algorithme II- 2, sont fixées au plus tôt pour toutes les

opérations. Par conséquent, si une des contraintes (II-4) n'est pas vérifiée (i.e. $T_{i_0+1,j}>b_{Z_{i_0,j},i_0,j}$, où $i_0 \in I_j$), aucune solution n'existe pour cet ensemble de fenêtres $\{Z_{i,j}\}_{1\le i\le m_j}$.

Puisque les fenêtres sont classées au pas 1.1 dans l'ordre des $a_{z,i,j}$ croissants, pour $1\le z\le y_{i,j}$, nous avons $a_{Z_{i,j},i,j}\le a_{Z^*_{i,j},i,j}$ si $Z_{i,j}\le Z^*_{i,j}$ pour tout $1\le i\le m_j$. Pour les fenêtres correspondant à $\{Z_{i,j}\}_{1\le i\le m_j}$ et $\{Z^*_{i,j}\}_{1\le i\le m_j}$, nous désignons respectivement par $\{T_{i,j}\}_{1\le i\le m_j+1}$ et $\{T^*_{i,j}\}_{1\le i\le m_j+1}$ les dates fournies par l'Algorithme II- 1 au pas 2.2 de l'Algorithme II- 2. D'après le Théorème II- 1, ces dates sont une solution au Problème II- 3 et donc vérifient les contraintes (II-3), i.e. $a_{Z_{i,j},i,j}\le T_{i,j}$ pour $1\le i\le m_j$. De plus, ces dates sont fixées au plus tôt et puisque $a_{Z_{i,j},i,j}\le a_{Z^*_{i,j},i,j}$ pour $1\le i\le m_j$, on a $T_{i,j}\le T^*_{i,j}$ pour tout $1\le i\le m_j$. Par conséquent, si la contrainte (II-4) associée à l'opération i_0 n'est pas vérifiée pour la fenêtre $Z_{i_0,j}$ (i.e. $b_{Z_{i_0,j},i_0,j}<T_{i_0+1,j}$), alors on a $b_{Z_{i_0,j},i_0,j}<T^*_{i_0+1,j}$. Cela signifie que, pour tout ensemble $\{Z^*_{i,j}\}_{1\le i\le m_j}$ tel que $Z_{i,j}\le Z^*_{i,j}$ où $1\le i\le m_j$ et $Z_{i_0,j}=Z^*_{i_0,j}$, la contrainte (II-4) ne peut être vérifiée. Ainsi, il est inutile de reconsidérer une fenêtre $Z_{i_0,j}$ d'une opération i_0 pour laquelle cette contrainte n'est pas vérifiée. Cette fenêtre est supprimée au pas 2.3.1.1.1 de l'Algorithme II- 2 en incrémentant $Z_{i_0,j}$ de 1, et on ne considère dans la suite de l'algorithme que les fenêtres à partir de $Z_{i_0,j}$ pour l'opération i_0. Sont conservées les seules fenêtres susceptibles de fournir une solution réalisable. **Si plus aucune fenêtre n'existe pour une opération, c'est ce qui se passe au pas 2.3.1.1.2.1, alors le problème n'a pas de solution réalisable.** De plus, si une solution est fournie au pas 2.2.1, cette solution est déterminée pour l'ensemble $\{Z_{i,j}\}_{1\le i\le m_j}$ correspondant aux fenêtres susceptibles de fournir une solution réalisable et de $a_{Z_{i,j},i,j}$ minimal. Par conséquent, toute autre solution réalisable ne pourrait être meilleure. On en déduit que la première solution réalisable rencontrée par l'Algorithme II- 2 est optimale pour le Problème II- 1.

CQFD

Complexité

Le tri au pas 1.1 de l'Algorithme II- 2 requiert $O(\sum_{1\le i\le m_j} y_{i,j}\log_2 y_{i,j})\le O(y_j \log_2 y_j)$ opérations [Beauquier et *al.*, 1992], où $y_j=\sum_{1\le i\le m_j} y_{i,j}$ est le nombre de fenêtres où chaque opération i peut être exécutée sur le produit j. Les pas 1.2 à 1.3 nécessitent $O(m_j)$ opérations. A chaque nouvelle itération du pas 2, au moins une opération i_0 ne satisfait pas la relation $T_{i_0+1,j}\le b_{Z_{i_0,j},i_0,j}$ du pas 2.2. Nous incrémentons alors $Z_{i_0,j}$ de 1 au pas 2.3.1.1. Puisque $Z_{i,j}$ est limité par $y_{i,j}$ au pas 2.3.1.1.2, le pas 2 est exécuté au plus $y_j=\sum_{1\le i\le m_j} y_{i,j}$ fois.

L'Algorithme II- 1 et les pas 2.1 à 2.3.1.1.2.1 requièrent $O(m_j)$ opérations. Par conséquent, **la complexité de l'Algorithme II- 2 est en $O(y_j\ m_j\ +\ y_j\ \log_2\ y_j)$.**

Remarques

Les fenêtres sont triées dans l'ordre croissant des $a_{z,i,j}$ et, en cas d'égalité, dans l'ordre croissant des $b_{z,i,j}$ au pas 1.1 de l'Algorithme II- 2. Nous avons donc choisi, en cas d'égalité de deux $a_{z,i,j}$ et $a_{z+1,i,j}$ successifs, de placer en premier la fenêtre qui est incluse dans la seconde, i.e. $b_{z,i,j} \leq b_{z+1,i,j}$. Ce choix est arbitraire et n'est pas nécessaire pour garantir l'optimalité des algorithmes. L'intérêt est de conserver les plus longues fenêtres possibles et d'espérer placer plus facilement des opérations futures dans ces fenêtres.

Le champ d'application de l'Algorithme II- 2 est très large. Il peut être utilisé pour résoudre des problèmes très différents dans lesquels des opérations doivent être réalisées dans des fenêtres. En fonction du problème à résoudre, nous utilisons, au pas 2.1 de cet algorithme, une procédure spécifique pour caler au plus tôt les opérations dans les fenêtres spécifiées. Il est ainsi possible de traiter les problèmes de routage avec fenêtres de temps [Hohzaki et *al.*, 1996] en utilisant les algorithmes de plus court chemin prenant en compte les contraintes de fenêtres [Chauvet et *al.*, 1998a].

II.3.5. Conclusion

Dans cette section, nous avons défini les dates d'exécution au plus tôt des opérations compatibles avec leurs durées opératoires admissibles et les fenêtres de temps dans lesquelles les opérations doivent être exécutées. Pour résoudre ce problème formalisé par le Problème II-1, nous avons proposé l'Algorithme II- 2 de complexité $O(y_j\ m_j\ +\ y_j\ \log_2\ y_j)$, où y_j est le nombre total de fenêtres dans lesquelles les m_j opérations peuvent être réalisées sur le produit. Nous avons prouvé que l'Algorithme II- 2 détermine les dates d'exécution au plus tôt des opérations sur le produit et ainsi minimise la date d'achèvement du produit, i.e. le *makespan* de chaque produit. A chaque arrivée d'une nouvelle commande, le produit peut ainsi être placé et on lance pour cela l'Algorithme II- 2. Aussi, nous sommes capables de fournir rapidement un délai minimum de livraison réalisable et précis.

L'opération i effectuée sur le produit j est affectée à la $Z_{i,j}$-ième fenêtre parmi les $y_{i,j}$ fenêtres possibles. En plaçant cette opération dans la fenêtre $[a_{Z_{i,j},i,j}\ ;\ b_{Z_{i,j},i,j}]$, outre les $(y_{i,j}\text{-}1)$ autres fenêtres, au plus deux fenêtres restent disponibles, à savoir les fenêtres $[a_{Z_{i,j},i,j}\ ;\ T_{i,j}]$, et $[T_{i+1,j}\ ;\ b_{Z_{i,j},i,j}]$. Par conséquent, si le produit $(j+1)$ nécessite la même série de machines que j (comme dans un *job shop*), on a $m_{j+1}=m_j$ et $y_{j+1}=\sum_{1\leq i\leq m_{j+1}}(y_{i,j}+1)=y_j+m_j$. Dans ce cas, le nombre d'opérations nécessaires pour ordonnancer de cette façon les n produits sur les m machines est de l'ordre de :

$$\sum_{1\le j\le n}\left(y_j m_j + y_j \log_2 y_j\right) = y_1 m + (n-1)m^2 + y_1 \log_2\left(y_1 + (n-1)m\right) + (n-1)m \log_2\left(y_1 + (n-1)m\right)$$
$$\le \left(y_1 + nm\right)\left[m + \log_2\left(y_1 + nm\right)\right].$$

Si tous les produits passent dans le même ordre sur la série de machines, c'est-à-dire si le système considéré est un *flow shop*, la contrainte de *no-wait* interdit à un produit de "dépasser" un produit lancé avant lui en production. Par conséquent, si la séquence d'entrée des produits dans un tel *flow shop* est imposée, l'Algorithme II- 2 exécuté successivement pour chaque produit permet de minimiser le flot des produits et le *makespan* de l'ensemble des produits. Dans le cas où tous les produits sont identiques, la séquence d'entrée est unique et l'Algorithme II- 2 exécutée successivement pour chaque produit permet de minimiser le flot et le *makespan* de toute la production.

II.4. MINIMISATION DU TEMPS TOTAL D'UTILISATION DES RESSOURCES

II.4.1. Introduction

Comme nous l'avons vu dans la section précédente, dès qu'une commande arrive, nous sommes capables de fournir rapidement un délai minimum de fabrication et ainsi de négocier avec le client la date de livraison du produit. Si la date de livraison fixée par le client est antérieure à la date de fin au plus tôt du produit, nous refusons la commande. Nous désignons la date de livraison du produit par d_j dans la suite.

Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit au plus tard, tout en respectant la date de livraison fixée au plus tôt par l'Algorithme II- 2 (voir section précédente). D'autre part, cette méthode de gestion permet de fournir, dès qu'une commande arrive, la marge de temps avant de commencer la fabrication du produit sans dépasser le délai de livraison.

En retardant le plus possible le début de fabrication du produit, nous limitons le temps d'occupation des ressources. Nous espérons ainsi laisser plus de liberté pour ordonnancer les produits futurs. En outre, en minimisant le temps d'utilisation des ressources, nous réduisons le coût de fabrication du produit. Pour retarder le plus possible la date de début de fabrication, nous définissons les dates d'exécution au plus tard des opérations compatibles avec leurs durées opératoires admissibles et les fenêtres de temps dans lesquelles les opérations peuvent être exécutées. Après avoir formalisé le problème, nous proposons un algorithme original de très bonne complexité. A notre connaissance, il n'existe pas d'étude sur ces problèmes avec des contraintes de type sans attente.

II.4.2. Formalisation

Nous voulons déterminer quand exécuter les opérations sur le produit j sachant que la date de livraison est d_j (voir contraintes (II-15)), $d_j \in \Re^+ \cup \{+\infty\}$. Cette date d_j est ultérieure à la date de fin au plus tôt du produit (établie dans la section précédente) sans quoi il est impossible de fabriquer le produit à temps. $T_{i,j}$ et $T_{i+1,j}$ définissent les instants de début et de fin de chaque opération $i \in I_j$ puisque les m_j opérations s'exécutent successivement sans attente. Nous devons déterminer la durée d'exécution $P_{i,j}=T_{i+1,j}-T_{i,j}$ de chaque opération i. Cette durée doit vérifier : $l_{i,j} \leq P_{i,j} \leq u_{i,j}$ (voir contraintes (II-2)). Nous devons également décider, pour chaque opération i, dans quelle fenêtre de temps de l'ensemble $W_{i,j}=\{[a_{z,i,j} ; b_{z,i,j}] / 1 \leq z \leq y_{i,j}\}$ cette opération peut être exécutée (voir contraintes (II-

3)-(II-4)). La fenêtre retenue est la fenêtre d'indice $Z_{i,j}$ (voir contraintes (II-5)). Notre objectif est de maximiser la date $T_{1,j}$ de commencement du produit. Le problème à résoudre s'écrit :

Problème II- 4 : Minimisation du temps total d'utilisation des ressources

$$\max \quad T_{1,j}$$

$$\text{tel que} \begin{cases} T_{m_j+1,j} \le d_j & & \text{(II-15)} \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & \text{(II-2)} \\ a_{Z_{i,j},i,j} \le T_{i,j} & \forall 1 \le i \le m_j & \text{(II-3)} \\ T_{i+1,j} \le b_{Z_{i,j},i,j} & \forall 1 \le i \le m_j & \text{(II-4)} \\ 1 \le Z_{i,j} \le y_{i,j} & \forall 1 \le i \le m_j & \text{(II-5)} \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & \text{(II-6)} \\ Z_{i,j} \in \mathrm{N} & \forall 1 \le i \le m_j & \text{(II-7)} \end{cases}$$

Les contraintes (II-6) et (II-7) donnent les intervalles de définition de toutes les variables de décisions $T_{i,j}$ et $Z_{i,j}$. $\Re^+$ et N désignent respectivement les ensembles des réels et des entiers.

Le Problème II- 3 est symétrique par rapport à l'axe des temps au Problème II- 1. D'ailleurs, il est possible de montrer l'équivalence entre les deux problèmes en faisant le changement de variable $T'_{i,j}=M-T_{m_j+2-i,j}$ pour $1 \le i \le m_j+1$ (où M est choisi arbitrairement grand). Aussi, l'approche de résolution du problème (que nous décrivons dans les deux sections suivantes) est similaire à celle utilisée pour résoudre le Problème II- 1 et se fait en temps polynomial. Elle consiste à résoudre une suite de sous-problèmes que nous décrivons dans les paragraphes qui suivent.

II.4.3. Sélection des durées opératoires

L'objectif de cette section est de déterminer $T_{i,j}$ et $T_{i+1,j}$, dates de début et de fin d'exécution au plus tard des opérations, si chaque opération i est exécutée sur le produit j dans la fenêtre donnée $Z_{i,j}$, $i \in I_j$. Nous ne tenons pas compte, dans la formulation, de la date de début des fenêtres $a_{Z_{i,j},i,j}$, i.e. les contraintes (II- 3). Le problème s'écrit :

Problème II- 5 : Définition des dates au plus tard des opérations quand leur fenêtre d'exécution est donnée

$$\max \quad \sum_{1 \le i \le m_j+1} \mu_i T_{i,j}$$

$$\text{tel que} \begin{cases} T_{m_j+1,j} \le d_j & & \text{(II-15)} \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & \text{(II-2)} \\ T_{i+1,j} \le b_{Z_{i,j},i,j} & \forall 1 \le i \le m_j & \text{(II-4)} \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & \text{(II-6)} \end{cases}$$

La solution optimale du Problème II- 5 pour l'ensemble $\{\mu_{i_0}=1\}\cup\{\mu_i=0/1\leq i\leq m_j+1,\ i\neq i_0\}$ correspond au cas où la date T_{i_0} est calée au plus tard. Si la solution optimale au Problème II- 5 est unique quel que soit l'ensemble $\{\mu_i\ /\ 1\leq i\leq m_j+1\}$ de valeurs positives, alors cette solution correspond au cas où chaque date est calée au plus tard. Nous recherchons une telle solution dans la suite. Si, de plus, la solution optimale de ce problème vérifie les contraintes (II- 3) (i.e. $a_{Z_{i,j},i,j}\leq T_{i,j}$, pour $1\leq i\leq m_j$), alors cette solution correspond à la solution calée au plus tard dans l'ensemble de fenêtres $\{Z_{i,j}\}_{1\leq i\leq m_j}$. Sinon, aucune solution n'est réalisable en pratique dans cet ensemble de fenêtres.

L'Algorithme II- 3 permet de résoudre le Problème II- 5 par une méthode qui se déroule en deux étapes, similaire à l'Algorithme II- 1. Dans la première, on définit τ_i, borne supérieure des dates de fin d'exécution de chaque opération $(i\text{-}1)$, en tenant compte des dates de fin des périodes et des durées minimales des opérations. Dans la seconde étape, on raffine cette borne en tenant compte des durées maximales des opérations. Cette nouvelle borne est alors la date optimale recherchée.

Algorithme II- 3 : Calage au plus tard des opérations du produit

Procédure Définition des dates au plus tard d'exécution des opérations effectuées sur le produit j.
Entrée : j est l'indice du produit considéré.
d_j est la date avant laquelle on doit livrer le produit j.
m_j est le nombre d'opérations à réaliser sur le produit j.
I_j est l'ensemble des opérations à réaliser sur le produit j.
$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i\in I_j$.
$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i\in I_j$.
$Z_{i,j}$ est la fenêtre dans laquelle on veut réaliser l'opération i sur le produit j, pour $i\in I_j$.
$b_{Z_{i,j},i,j}$ est la date de fin de la période durant laquelle on réalise l'opération i sur le produit j, pour $i\in I_j$.
Sortie : $T_{i,j}$ est la date de début au plus tard de l'opération i effectuée sur le produit j, où $i\in I_j$.
$T_{m_j+1,j}$ est la date de fin au plus tard du produit j.

1. **Etape 1.**
1.1. **Poser** $\tau_{m_j+1}:=\min\{d_j\ ;\ b_{Z_{m_j,j},m_j,j}\}$. *On note τ_i la borne supérieure de la date de fin d'exécution de l'opération $(i-1)$.*
1.2. **Pour** i allant de m_j à 2 : *On note i l'indice de l'opération sélectionnée.*
1.2.1. **Poser** $\tau_i:=\min\{\tau_{i+1}-l_{i,j}\ ;\ b_{Z_{i-1,j},i-1,j}\}$.
2. **Etape 2.**
2.1. **Poser** $T_{1,j}:=\tau_2-l_{1,j}$.
2.2. **Pour** i allant de 2 à m_j+1 : *On note i l'indice de l'opération sélectionnée.*
2.2.1. **Poser** $T_{i,j}:=\min\{T_{i-1,j}+u_{i-1,j}\ ;\ \tau_i\}$.

Exemple

Nous proposons un exemple d'application de l'Algorithme II- 3. Les données relatives au produit j à réaliser sont fournies dans le Tableau II- 5. Le produit doit être réalisé avant la date $d_j=12$.

Les résultats de l'exécution de l'Algorithme II- 3 sont explicités sur la Figure II- 10. Les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Tableau II- 5 : Données relatives à l'exécution de l'Algorithme II- 3

ENTREES				RESULTATS INTERMEDIAIRES	SORTIES
Indice de l'opération i	Durée minimale $l_{i,j}$	Durée maximale $u_{i,j}$	Date de fin de la période durant laquelle on réalise l'opération i $b_{z_{i,j},i,j}$	Borne supérieure de la date au plus tard τ_i	Date au plus tard $T_{i,j}$
1	1	+∞	4		3
2	3	4	8	4	4
3	1	1	9	8	8
4	2	4	+∞	9	9
m_j=5	1	+∞	+∞	11	11
m_j+1=6				12	12

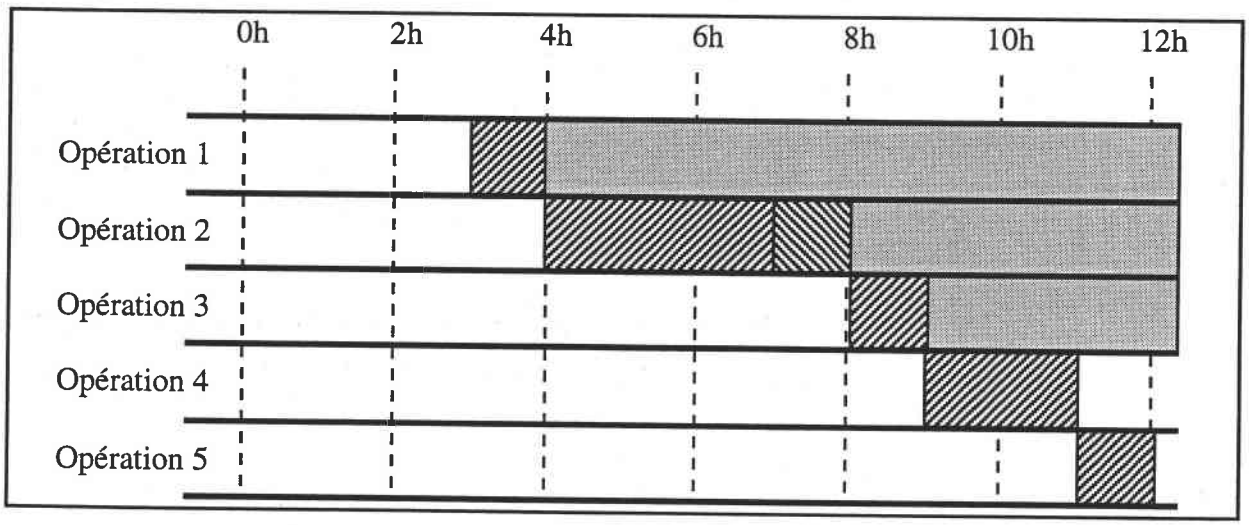

Figure II- 10 : Résultats fournis par l'Algorithme II- 3

Théorème II- 3

L'Algorithme II- 3 détermine les dates de début et de fin d'exécution au plus tard de toutes les opérations, i.e. il fournit une solution unique optimale au Problème II- 5.

Démonstration

a) Nous montrons tout d'abord que la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournie par l'Algorithme II- 3 est réalisable.

Suivant les pas 1.1 et 2.2.1 de l'algorithme, on a $T_{m_j+1,j}\le\tau_{m_j+1}\le d_j$. Par conséquent, la solution vérifie l'inéquation (II-15).

Du pas 2.2.1 de l'algorithme, on déduit $T_{i+1,j}-l_{i,j}=\min\{T_{i,j}+(u_{i,j}-l_{i,j})\ ;\ \tau_{i+1}-l_{i,j}\}$, pour $1\le i\le m_j$. En considérant l'hypothèse $l_{i,j}\le u_{i,j}$, on obtient $T_{i+1,j}-l_{i,j}\ge\min\{T_{i,j}\ ;\ \tau_{i+1}-l_{i,j}\}$. Comme $T_{i,j}\le\tau_i\le\tau_{i+1}-l_{i,j}$, pour $2\le i\le m_j$ (cf. pas 2.2.1 et 1.2.1), on a $T_{i+1,j}-l_{i,j}\ge\min\{T_{i,j};T_{i,j}\}$. Considérant le pas 2.1, on en déduit $T_{i,j}+l_{i,j}\ge T_{i+1,j}$, pour $1\le i\le m_j$. De plus, selon le pas 2.2.1 de l'algorithme, $T_{i+1,j}\le T_{i,j}+u_{i,j}$, pour $1\le i\le m_j$. On conclut que, pour $1\le i\le m_j$, $l_{i,j}\le T_{i+1,j}-T_{i,j}\le u_{i,j}$. Donc, la solution satisfait les contraintes (II-2).

Suivant les pas 1.1, 1.2.1 et 2.2.1 de l'algorithme, on a $b_{z_{i-1,j},i-1,j}\le\tau_i\le T_{i,j}$, pour $2\le i\le m_j+1$. Par conséquent, la solution vérifie les inéquations (II-4).

La solution fournie par l'algorithme satisfait les contraintes (II-15), (II-2), (II-4) et (II-6). Elle est réalisable.

b) Nous montrons ensuite que la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournie par l'Algorithme II- 3 est optimale.

Selon les contraintes (II-15) et (II-4), la valeur $\min\{d_j\ ;\ b_{z_{m_j+1,j},m_j+1,j}\}$ est une borne supérieure de la date de fin de la dernière opération. D'après le pas 1.1 de l'Algorithme II- 3, τ_{m_j+1} est donc une borne supérieure de cette date.

Si τ_{i+1} est une borne supérieure de la date de fin de l'opération i, pour $m_j\ge i\ge 2$, selon les contraintes (II-2) et (II-4), la valeur $\min\{\tau_{i+1}-l_{i,j}\ ;\ b_{z_{i-1,j},i-1,j}\}$ est une borne supérieure de la date de fin de l'opération $(i-1)$. D'après le pas 1.2.1 de l'Algorithme II- 3, τ_i est donc une borne supérieure de cette date, pour $m_j+1\ge i\ge 2$.

Selon la contrainte (II-2), la valeur $\tau_2-l_{1,j}$ est une borne supérieure de la date de début de la première opération. D'après le pas 2.1 de l'Algorithme II- 3, $T_{1,j}$ est donc une borne supérieure de cette date.

Si $T_{i-1,j}$ et τ_i sont des bornes supérieures des dates respectivement de début et de fin de l'opération $i-1$, pour $2\le i\le m_j+1$, selon la contrainte (II-2), la valeur $\min\{T_{i-1,j}+u_{i,j}\ ;\ \tau_i\}$ est une borne supérieure de la date de début de l'opération i. D'après le pas 2.2.1 de l'Algorithme II- 1, $T_{i,j}$ est donc une borne supérieure de cette date, pour $m_j\ge i\ge 1$.

Chaque date $T_{i,j}$ fournie par l'Algorithme II- 3, est une borne supérieure de la date de début de l'opération i, pour $m_j\ge i\ge 1$. D'après a, on conclut que l'Algorithme II- 3 détermine les dates

d'exécution au plus tard de toutes les opérations. Par conséquent, il fournit la solution optimale au Problème II- 4 quelque soit l'ensemble $\{\mu_i \ / \ 1 \leq i \leq m_j+1\}$ de valeurs strictement positives.

CQFD

Complexité

Dans tous les cas, l'Algorithme II- 3 nécessite $2m_j$ comparaisons (effectuées aux pas 1.1, 1.2.1 et 2.2.1) et $2m_j+1$ affectations (effectuées aux pas 1.1, 1.2.1, 2.1 et 2.2.1), où m_j est le nombre d'opérations effectuées sur le produit à ordonnancer. La complexité de l'Algorithme II-3 est en $O(m_j)$.

Tout algorithme doit au moins examiner chacune des opérations pour résoudre le problème. Par conséquent, aucun algorithme ne peut avoir une complexité inférieure à $O(m_j)$, c'est-à-dire inférieure à celle de l'Algorithme II- 3.

Remarque

Comme le Problème II- 3, le Problème II- 5 peut être représenté sous forme d'un graphe "potentiels étapes" [Roy, 1960], ou de son équivalent, le graphe PERT [Malcom et *al.*, 1959]. Cependant, une telle modélisation impose de créer des circuits dans le graphe PERT si bien que des algorithmes spécifiques [Ford, 1956] [Bellman, 1958] [Gondran et Minoux, 1979] doivent être utilisés. La complexité de ces algorithmes en $O(m_j^2)$ est moins bonne que celle de l'Algorithme II- 3 pour un résultat identique.

II.4.4. Sélection des fenêtres

L'objectif de cette section est de résoudre le Problème II- 4. Autrement dit, nous allons déterminer dans quelle fenêtre $Z_{i,j}$, l'opération i doit être effectuée de manière à ce que le produit j soit terminée le plus tard possible. Nous utiliserons les résultats de la section précédente. L'Algorithme II- 4 permet de résoudre ces problèmes. Pour cela, il examine les fenêtres relatives à chaque opération dans leur ordre chronologique sans qu'aucun retour arrière ne soit nécessaire.

Algorithme II- 4 : Choix des fenêtres pour un début de fabrication au plus tard

Procédure Définition des fenêtres permettant d'exécuter les opérations au plus tard sur le produit j.

Entrée : j est l'indice du produit considéré.

d_j est la date avant laquelle on doit livrer le produit j.

m_j est le nombre d'opérations à réaliser sur le produit j.

I_j est l'ensemble des opérations à réaliser sur le produit j.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$y_{i,j}$ est le nombre de fenêtres dans laquelle on peut réaliser l'opération i sur le produit j, pour $i \in I_j$.

$a_{z,i,j}$ est la date de début de la z-ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \le z \le y_{i,j}$, $i \in I_j$.

$b_{z,i,j}$ est la date de fin de la z-ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \le z \le y_{i,j}$, $i \in I_j$.

Sortie : $T_{i,j}$ est la date de début au plus tard de l'opération i effectuée sur le produit j, où $i \in I_j$.

$T_{m_j+1,j}$ est la date de fin au plus tard du produit j.

$Z_{i,j}$ est la fenêtre dans laquelle on doit réaliser l'opération i sur le produit j, où $i \in I_j$.

1. **Initialisation.**

1.1. **Trier** les fenêtres $[a_{z,i,j}\ ;\ b_{z,i,j}]$ pour $1 \le z \le y_{i,j}$, $i \in I_j$,
dans l'ordre croissant des $b_{z,i,j}$ et, en cas d'égalité, dans l'ordre croissant des $a_{z,i,j}$.
Dans la suite de l'algorithme, la z-ième fenêtre est le z-ième élément dans cet ordre, $1 \le z \le y_{i,j}$.

1.2. **Pour** $i \in I_j$:

1.2.1. **Poser** $Z_{i,j} := y_{i,j}$.

1.3. **Poser** $Trouvé := 0$. *On note $Trouvé$ la variable qui vaut* 1 *si une solution réalisable est trouvée,* 0 *sinon.*

2. **Tant que** ($Trouvé = 0$) : *Aucune solution réalisable n'a été trouvée.*

2.1. **Appliquer** l'Algorithme II- 3.

Entrée : j, d_j, m_j, I_j, $\{l_{i,j}\}_i$, $\{u_{i,j}\}_i$, $\{Z_{i,j}\}_i$, $\{b_{Z_{i,j},i,j}\}_{i \in I_j}$.

Sortie : $\{T_{i,j}\}_{i \in I_j}$, $T_{m_j+1,j}$.

2.2. **Si** ($a_{Z_{i,j},i,j} \le T_{i,j}$, pour $i \in I_j$) **alors** : *Chaque opération i commence après $a_{Z_{i,j},i,j}$ c'est-à-dire après le début de la fenêtre $Z_{i,j}$.*

2.2.1. **Poser** $Trouvé := 1$. *Une solution réalisable a été trouvée.*

2.3. **Sinon** :

2.3.1. **Pour** $i \in I_j$: *On note i l'indice de l'opération examinée.*

2.3.1.1. **Tant que** ($a_{Z_{i,j},i,j} > T_{i,j}$) **alors** : *L'opération i ne peut être commencée dans la fenêtre $Z_{i,j}$.*

2.3.1.1.1. **Poser** $Z_{i,j} := Z_{i,j} - 1$. *On sélectionne la fenêtre précédente.*

2.3.1.1.2. **Si** ($Z_{i,j} < 1$) **alors** : *Aucune fenêtre n'a permis de trouver une solution réalisable.*

2.3.1.1.2.1. **Quitter**. *Il n'existe aucune solution réalisable.*

Exemple

Nous proposons un exemple d'application de l'Algorithme II- 4. Les données relatives au produit j à réaliser sont fournies dans le Tableau II- 6 et dans le Tableau II- 5 pour les durées opératoires. Le produit doit être livré avant la date $d_j = 12$.

Les résultats de l'exécution sont explicités sur la Figure II- 11 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Tableau II- 6 : Données relatives à l'exécution de l'Algorithme II- 4

ENTREES		RESULTATS INTERMEDIAIRES Itération 1		SORTIES Itération 2		
i	$\{[a_{z,i,j};b_{z,i,j}]/ 1\leq z\leq y_{i,j}\}$	$[a_{z,i,j};b_{z,i,j}]$	$T_{i,j}$	$[a_{z,i,j};b_{z,i,j}]$	$T_{i,j}$	$Z_{i,j}$
1	[0;1], [2;4], [5;9], [11;+∞[	[11;+∞[	4	[2;4]	3	2
2	[0;8], [11;+∞[	[11;+∞[	5	[0;8]	4	1
3	[0;1], [2;4], [5;9], [11;+∞[	[11;+∞[	7	[5;9]	8	3
4	[0;3], [7;+∞[	[7;+∞[	8	[7;+∞[	9	2
m_j=5	[0;1], [2;4], [5;9], [11;+∞[	[11;+∞[	11	[11;+∞[	11	4
m_j+1=6			12		12	

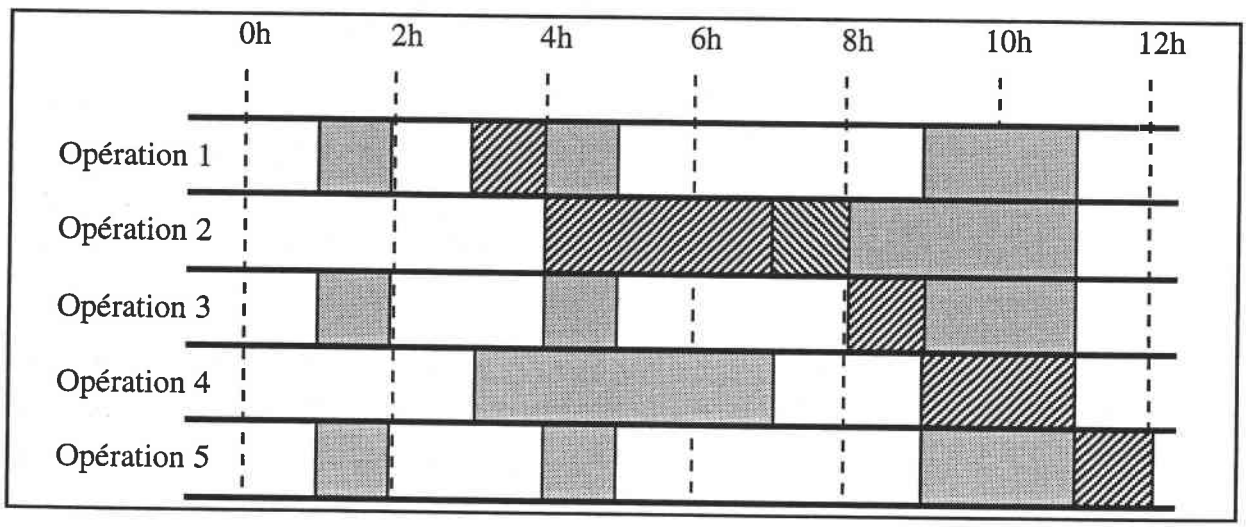

Figure II- 11 : Résultats fournis par l'Algorithme II- 4

Si l'Algorithme II- 4 s'achève au pas 2.3.1.1.2.1, la solution courante n'est pas réalisable et même, il n'existe aucune solution réalisable (comme nous le démontrons dans la suite). Cela signifie qu'il est impossible de fabriquer le produit avec les contraintes imposées au système de production. Ceci ne se produit jamais si la date de livraison d_j est supérieure à la date de fin au plus tôt calculée dans la section précédente.

Théorème II- 4

Si le Problème II- 4 a une solution réalisable, l'Algorithme II- 4 détermine les fenêtres et les dates d'exécution qui permettent de finir au plus tard toutes les opérations effectuées sur le produit. Par conséquent, la solution fournie par l'Algorithme II- 4 est optimale pour le Problème II- 4.

Démonstration

a) Nous montrons tout d'abord que si l'Algorithme II- 4 fournit une solution $\{\{T_{i,j}\}_{1\le i\le m_j+1}\ ;\ \{Z_{i,j}\}_{1\le i\le m_j}\}$, cette solution est réalisable pour le Problème II- 4.

Les dates $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournies par l'Algorithme II- 4 sont déterminées au pas 2.1 par l'Algorithme II- 3 pour les fenêtres retenues $\{Z_{i,j}\}_{1\le i\le m_j}$. D'après le Théorème II- 3, cette solution vérifie les contraintes (II-15), (II-2), (II-4) et (II-6). Puisque tout au long de l'Algorithme II- 4, la variable $Z_{i,j}$ prend des valeurs dans $\{1, 2, \ldots, y_{i,j}\}$, les contraintes (II-5) et (II-7) sont vérifiées par la solution. Au pas 2.2 de l'Algorithme II- 4, on s'assure que toutes les contraintes (II-3) (i.e. $a_{Z_{i,j},i,j}\le T_{i,j}$, pour $i\in I_j$) sont vérifiées avant de déclarer qu'une solution est réalisable. Par conséquent, la solution fournie par l'Algorithme II- 4 est réalisable.

b) Nous montrons ensuite que la première solution réalisable rencontrée par l'Algorithme II- 4 est optimale pour le Problème II- 4.

D'après le Théorème II- 3, pour les fenêtres $\{Z_{i,j}\}_{1\le i\le m_j}$, les dates $\{T_{i,j}\}_{1\le i\le m_j+1}$ fournies par l'Algorithme II- 3 au pas 2.2 de l'Algorithme II- 4, sont fixées au plus tard pour toutes les opérations. Par conséquent, si une des contraintes (II-3) n'est pas vérifiée (i.e. $a_{Z_{i_0,j},i_0,j}>T_{i_0,j}$, où $i_0\in I_j$), aucune solution n'existe pour cet ensemble de fenêtres $\{Z_{i,j}\}_{1\le i\le m_j}$.

Puisque les fenêtres sont classées au pas 1.1 dans l'ordre des $b_{z,i,j}$ croissants, pour $1\le z\le y_{i,j}$, nous avons $b_{Z^*_{i,j},i,j}\le b_{Z_{i,j},i,j}$ si $Z^*_{i,j}\le Z_{i,j}$ pour tout $1\le i\le m_j$. Pour les fenêtres correspondant à $\{Z^*_{i,j}\}_{1\le i\le m_j}$ et $\{Z_{i,j}\}_{1\le i\le m_j}$, nous désignons respectivement par $\{T^*_{i,j}\}_{1\le i\le m_j+1}$ et $\{T_{i,j}\}_{1\le i\le m_j+1}$ les dates fournies par l'Algorithme II- 3 au pas 2.2 de l'Algorithme II- 4. D'après le Théorème II- 3, ces dates sont une solution au Problème II- 5 et donc vérifient les contraintes (II-4), i.e. $T_{i+1,j}\le b_{Z_{i,j},i,j}$ pour $1\le i\le m_j$. De plus, ces dates sont fixées au plus tard et puisque $b_{Z^*_{i,j},i,j}\le b_{Z_{i,j},i,j}$ pour $1\le i\le m_j$, on a $T^*_{i,j}\le T_{i,j}$ pour tout $1\le i\le m_j$. Par conséquent, si la contrainte (II-3) associée à l'opération i_0 n'est pas vérifiée pour la fenêtre $Z_{i_0,j}$ (i.e. $T_{i_0,j}<a_{Z_{i_0,j},i_0,j}$), alors on a $T^*_{i_0,j}<a_{Z_{i_0,j},i_0,j}$. Cela signifie que pour tout ensemble $\{Z^*_{i,j}\}_{1\le i\le m_j}$ tel que $Z^*_{i,j}\le Z_{i,j}$ où $1\le i\le m_j$ et $Z^*_{i_0,j}=Z_{i_0,j}$, la contrainte (II-3) ne peut être vérifiée. Ainsi, il est inutile de reconsidérer une fenêtre $Z_{i_0,j}$ d'une

opération i_0 pour laquelle cette contrainte n'est pas vérifiée. Cette fenêtre est supprimée au pas 2.3.1.1.1 de l'Algorithme II- 4 en décrémentant $Z_{i_0,j}$ de 1 et on ne considère dans la suite de l'algorithme que les fenêtres jusqu'à $Z_{i_0,j}$ pour l'opération i_0. Sont conservées les seules fenêtres susceptibles de fournir une solution réalisable. **Si plus aucune fenêtre n'existe pour une opération, c'est ce qui se passe au pas 2.3.1.1.2.1, alors le problème n'a pas de solution réalisable.** De plus, si une solution est fournie au pas 2.2.1, cette solution est déterminée pour l'ensemble $\{Z_{i,j}\}_{1 \le i \le m_j}$ correspondant aux fenêtres susceptibles de fournir une solution réalisable et de $b_{z_{i,j},i,j}$ maximal. Par conséquent, aucune autre solution réalisable ne pourrait être meilleure. On en déduit que la première solution réalisable rencontrée par l'Algorithme II- 4 est optimale pour le Problème II- 4. CQFD

Complexité

La structure de l'Algorithme II- 4 est similaire à celle de l'Algorithme II- 2. Ainsi, l'étude de complexité qui a été faite est applicable à l'Algorithme II- 4. Par conséquent, **la complexité de l'Algorithme II- 4 est en $O(y_j\ m_j\ +\ y_j\ \log_2\ y_j)$.**

Remarques

Les fenêtres sont triées dans l'ordre croissant des $b_{z,i,j}$ au pas 1.1 de l'Algorithme II- 4. Dans le cas où il n'existe pas de fenêtres incluses pour une opération, comme $b_{z,i,j} \le b_{z+1,i,j}$, on a $a_{z,i,j} \le a_{z+1,i,j}$. Ainsi, le tri du pas 1.1 de l'Algorithme II- 4 est équivalent à celui exécuté du pas 1.1 de l'Algorithme II- 2.

II.4.5. Conclusion

Dans cette section, une méthode de gestion en temps réel est décrite. Elle ordonnance chaque produit au plus tard, sans dépasser la date de livraison fixée au plus tôt (calculée dans la section précédente). De plus, elle permet de donner, dès qu'une commande arrive, la marge de temps restant avant de commencer la fabrication du produit tout en respectant le délai de livraison. En retardant le début de fabrication du produit, nous limitons le plus possible le temps d'occupation des ressources et espérons ainsi laisser plus de liberté pour ordonnancer les produits à venir.

Les dates d'exécution des opérations sont définies au plus tard et sont compatibles avec leurs durées opératoires admissibles et les fenêtres de temps dans lesquelles les opérations peuvent être exécutées. Pour résoudre ce problème formalisé par le Problème II- 4, nous avons proposé l'Algorithme II- 4 de complexité $O(y_j\ m_j\ +\ y_j\ \log_2\ y_j)$, où y_j est le nombre total de fenêtres dans lesquelles les m_j opérations peuvent être réalisées sur le produit. Nous avons prouvé que cet algorithme détermine les dates d'exécution au plus tard des opérations effectuées sur le produit (voir Théorème II- 4), et ainsi maximise la date de début de fabrication du produit.

II.5. MINIMISATION DU COUT D'UTILISATION DES RESSOURCES

II.5.1. Introduction

Dès qu'une commande arrive, nous sommes capables de fournir rapidement un délai minimum de livraison réalisable ainsi que la marge de temps restant avant de commencer la fabrication. Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit de manière à minimiser le coût d'utilisation des ressources. Cet ordonnancement doit également respecter le délai de livraison fixé au plus tôt (voir section 3). Nous pouvons également affecter aux ressources goulots un coût important afin d'inciter le système à minimiser les temps opératoires sur ces machines et, ainsi, espérer augmenter le nombre d'opérations qui peuvent être effectuées sur ces ressources. A notre connaissance, aucune étude n'existe sur ce type de problème. Nous proposons de formaliser le problème, puis de le résoudre grâce à deux algorithmes originaux.

II.5.2. Formalisation

Nous voulons déterminer quand exécuter les opérations sur le produit j sachant que la date de disponibilité est r_j et la date de livraison est d_j. $T_{i,j}$ et $T_{i+1,j}$ définissent les instants de début et de fin de chaque opération i, $1 \le i \le m_j$. Nous devons déterminer la durée d'exécution $P_{i,j}=T_{i+1,j}-T_{i,j}$ de chaque opération i, durée comprise dans l'intervalle $[l_{i,j} ; u_{i,j}]$. Nous devons décider dans quelle fenêtre de temps de $W_{i,j}=\{[a_{z,i,j} ; b_{z,i,j}] / 1 \le z \le y_{i,j}\}$ exécuter l'opération i. La fenêtre retenue est la fenêtre d'indice $Z_{i,j}$. Notre objectif est de minimiser le coût d'utilisation des ressources sachant que le coût induit en fabrication par l'opération i est $s_{i,j}$. Les contraintes sont similaires au Problème II- 1 et peuvent se formaliser ainsi :

Problème II- 6 : Minimisation du coût d'utilisation des ressources

$$\min \quad \sum_{1 \le i \le m_j} s_{i,j}\left(T_{i+1,j} - T_{i,j}\right)$$

$$\text{tel que} \begin{cases} r_j \le T_{1,j} & & \text{(II-1)} \\ T_{m_j+1,j} \le d_j & & \text{(II-15)} \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & \text{(II-2)} \\ a_{Z_{i,j},i,j} \le T_{i,j} & \forall 1 \le i \le m_j & \text{(II-3)} \\ T_{i+1,j} \le b_{Z_{i,j},i,j} & \forall 1 \le i \le m_j & \text{(II-4)} \\ 1 \le Z_{i,j} \le y_{i,j} & \forall 1 \le i \le m_j & \text{(II-5)} \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & \text{(II-6)} \\ Z_{i,j} \in \mathrm{N} & \forall 1 \le i \le m_j & \text{(II-7)} \end{cases}$$

La section suivante définit une approche de résolution permettant de fournir, en un temps polynomial, une solution dans le cas où la période pendant laquelle chaque opération doit être exécutée est donnée. Dans la section II.6.4, nous proposons un algorithme pseudo-polynomial résolvant le problème formulé ici, dans le cas général. Mais, nous ne savons pas si ce problème est NP-difficile. L'algorithme qui résout le cas le plus général n'utilise pas l'algorithme présenté dans la section suivante.

II.5.3.Sélection des durées opératoires

L'objectif de cette section est de déterminer $T_{i,j}$ et $T_{i+1,j}$, dates de début et de fin d'exécution de chaque opération, de manière à minimiser le coût d'utilisation des ressources, si chaque opération i est exécutée sur le produit j dans la fenêtre donnée $Z_{i,j}$, $i \in I_j$. Le problème s'écrit :

Problème II- 7 : Ordonnancement de coût minimal dans le cas où les fenêtres d'exécution des opérations sont données

$$\min \quad \sum_{1 \le i \le m_j} s_{i,j}\left(T_{i+1,j} - T_{i,j}\right)$$

$$\text{tel que} \begin{cases} r_j \le T_{1,j} & & (\text{II-}1) \\ T_{m_j+1,j} \le d_j & & (\text{II-}15) \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & (\text{II-}2) \\ a_{z_{i,j},i,j} \le T_{i,j} & \forall 1 \le i \le m_j & (\text{II-}3) \\ T_{i+1,j} \le b_{z_{i,j},i,j} & \forall 1 \le i \le m_j & (\text{II-}4) \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & (\text{II-}6) \end{cases}$$

Nous désignons par $\underline{T}_{i,j}$ et $\overline{T}_{i,j}$ la date de début de l'opération i effectuée sur le produit j calée respectivement au plus tôt et au plus tard dans la fenêtre donnée $Z_{i,j}$, $i \in I_j$. L'ensemble des dates calées au plus tôt est la solution optimale au Problème II- 3, donnée par l'Algorithme II- 1. De même, l'ensemble des dates calées au plus tard est la solution optimale au Problème II- 5, donnée par l'Algorithme II- 3. En utilisant ces notations, nous reformulons le Problème II- 7 :

Problème II- 8 : Ordonnancement de coût minimal dans le cas où les fenêtres d'exécution des opérations sont données

$$\min \quad \sum_{1 \le i \le m_j} s_{i,j}\left(T_{i+1,j} - T_{i,j}\right)$$

$$\text{tel que} \begin{cases} \underline{T}_{i,j} \le T_{i,j} & \forall 1 \le i \le m_j + 1 & (\text{II-}16) \\ T_{i,j} \le \overline{T}_{i,j} & \forall 1 \le i \le m_j + 1 & (\text{II-}17) \\ l_{i,j} \le T_{i+1,j} - T_{i,j} \le u_{i,j} & \forall 1 \le i \le m_j & (\text{II-}2) \\ T_{i,j} \in \Re^+ & \forall 1 \le i \le m_j + 1 & (\text{II-}6) \end{cases}$$

Théorème II- 5

Le Problème II- 7 et le Problème II- 8 sont équivalents.

Démonstration

Puisque le Problème II- 7 et le Problème II- 8 ont la même fonction objectif, il suffit de montrer qu'une solution réalisable pour l'un des problèmes est réalisable pour l'autre.

a) Nous montrons tout d'abord qu'une solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ réalisable pour le Problème II- 8, est réalisable pour le Problème II- 7.

Soit $\{T_{i,j}\}_{1\le i\le m_j+1}$ une solution réalisable pour le Problème II- 8. Par définition, cette solution vérifie les contraintes (II-2) et (II-6).

D'après la contrainte (II-16), on a $\underline{T}_{i,j}\le T_{i,j}$, pour $1\le i\le m_j$. Or, la solution $\{\underline{T}_{i,j}\}_{1\le i\le m_j+1}$ est solution optimale du Problème II- 3 et vérifie les contraintes (II-1) (i.e. $r_j\le\underline{T}_{1,j}$) et (II-3) (i.e. $a_{z_{i,j},i,j}\le\underline{T}_{i,j}$, pour $1\le i\le m_j$). Par transitivité, la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ vérifie également les contraintes (II-1) et (II-3).

D'après la contrainte (II-17), on a $T_{i,j}\le\overline{T}_{i,j}$, pour $2\le i\le m_j+1$. Or, la solution $\{\overline{T}_{i,j}\}_{1\le i\le m_j+1}$ est solution optimale du Problème II- 5 et vérifie les contraintes (II-15) (i.e. $\overline{T}_{m_j+1,j}\le d_j$) et (II-4) (i.e. $\overline{T}_{i,j}\le b_{z_{i,j},i,j}$, pour $1\le i\le m_j$). Par transitivité, la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ vérifie également les contraintes (II-15) et (II-4).

La solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ vérifie les contraintes (II-1), (II-15), (II-2), (II-3), (II-4) et (II-6), et donc est réalisable pour le Problème II- 7.

b) Nous montrons ensuite qu'une solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ réalisable pour le Problème II- 7, est réalisable pour le Problème II- 8.

Soit $\{T_{i,j}\}_{1\le i\le m_j+1}$ une solution réalisable pour le Problème II- 7. Par définition, cette solution vérifie les contraintes (II-2) et (II-6).

Toute solution du Problème II- 7 vérifie les contraintes (II-1), (II-2), (II-3) et (II-6), et est donc réalisable pour le Problème II- 3. La solution $\{\underline{T}_{i,j}\}_{1\le i\le m_j+1}$ est optimale pour le Problème II-3, c'est-à-dire qu'aucune date $\underline{T}_{i,j}$ ne peut être réduite, pour $1\le i\le m_j+1$. Par conséquent, toute solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ étant réalisable pour le Problème II- 3, vérifie la contrainte (II-16) (i.e. $\underline{T}_{i,j}\le T_{i,j}$, pour $1\le i\le m_j+1$).

Toute solution du Problème II- 7 vérifie les contraintes (II-15), (II-2), (II-4) et (II-6), et est donc réalisable pour le Problème II- 5. La solution $\{\overline{T}_{i,j}\}_{1\le i\le m_j+1}$ est optimale pour le Problème II- 5, c'est-à-dire qu'aucune date $\overline{T}_{i,j}$ ne peut être augmentée, pour $1\le i\le m_j+1$. Par conséquent, toute solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ étant réalisable pour le Problème II- 5, vérifie la contrainte (II-17) (i.e. $T_{i,j}\le\overline{T}_{i,j}$ pour $1\le i\le m_j+1$).

La solution $\{T_{i,j}\}_{1 \le i \le m_j+1}$ vérifie les contraintes (II-16), (II-17), (II-2) et (II-6), et donc est réalisable pour le Problème II- 8.

CQFD

Algorithme II- 5 : Minimisation du coût d'utilisation des ressources

Procédure Définition des dates d'exécution des opérations sur le produit j induisant un coût d'utilisation des ressources minimal.

Entrée : j est l'indice du produit considéré.

r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.

d_j est la date avant laquelle on doit livrer le produit j.

m_j est le nombre d'opérations à réaliser sur le produit j.

I_j est l'ensemble des opérations à réaliser sur le produit j.

$s_{i,j}$ est le coût d'utilisation des ressources induit à chaque unité de temps par l'opération i effectuée sur le produit j, pour $i \in I_j$.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$Z_{i,j}$ est la fenêtre dans laquelle on veut réaliser l'opération i sur le produit j, pour $i \in I_j$.

$a_{Z_{i,j},i,j}$ est la date de début de la période durant laquelle on réalise l'opération i, pour $i \in I_j$.

$b_{Z_{i,j},i,j}$ est la date de fin de la période durant laquelle on réalise l'opération i, pour $i \in I_j$.

Sortie : $T_{i,j}$ est la date de début de l'opération i effectuée sur le produit j, où $i \in I_j$.

$T_{m_j+1,j}$ est la date de fin du produit j.

1. **Poser** $Test$:=1. *On note $Test$ la variable qui vaut 1 jusqu'à ce qu'un test sur l'existence d'une solution réalisable soit réalisé.*
2. **Trier** l'ensemble des opérations $I_j=\{1, 2, \ldots, m_j\}$, dans l'ordre décroissant des $s_{i,j}$.
   On désigne par $\sigma(k)$, la k-ième opération suivant cet ordre. On a $s_{\sigma(k),j} \ge s_{\sigma(k+1),j}$.
3. **Pour** i allant de 1 à m_j : *On note i l'indice de l'opération sélectionnée.*
   3.1. **Poser** $l^*_i := l_{i,j}$. *On note l^*_i la durée minimale établie pour l'opération i.*
   3.2. **Poser** $u^*_i := u_{i,j}$. *On note u^*_i la durée maximale établie pour l'opération i.*
4. **Pour** k allant de 1 à m_j : *On note $\sigma(k)$ l'indice de l'opération sélectionnée.*
   4.1. **Si** $(l^*_{\sigma(k)} \ne u^*_{\sigma(k)})$ **alors** : *La durée opératoire de l'opération $\sigma(k)$ n'est pas fixée.*
      4.1.1. **Appliquer** l'Algorithme II- 1.
      Entrée : $j, r_j, m_j, I_j, \{l^*_i\}_i, \{u^*_i\}_i, \{Z_{i,j}\}_i, \{a_{Z_{i,j},i,j}\}_{i \in I_j}$.
      Sortie : $\{\underline{T}_{i,j}\}_{i \in I_j}, \underline{T}_{m_j+1,j}$. *On note $\underline{T}_{i,j}$ les dates au plus tôt des opérations.*
      4.1.2. **Appliquer** l'Algorithme II- 3.
      Entrée : $j, d_j, m_j, I_j, \{l^*_i\}_i, \{u^*_i\}_i, \{Z_{i,j}\}_i, \{b_{Z_{i,j},i,j}\}_{i \in I_j}$.
      Sortie : $\{\overline{T}_{i,j}\}_{i \in I_j}, \overline{T}_{m_j+1,j}$. *On note $\overline{T}_{i,j}$ les dates au plus tôt des opérations.*
      4.1.3. **Si** $(Test=1)$ **alors** : *Il s'agit de la première itération.*
         4.1.3.1. **Poser** $Test$:=0. *On s'interdit de refaire ce test.*
         4.1.3.2. **Si** (il existe $i_0 \in \{1, 2, \ldots, m_j+1\}$, $\underline{T}_{i_0,j} > \overline{T}_{i_0,j}$) **alors** :
            4.1.3.2.1. **Quitter**. *Aucune solution réalisable n'existe.*
      4.1.4. **Poser** $l^*_{\sigma(k)} := \max\{l_{\sigma(k),j} \; ; \; \underline{T}_{\sigma(k)+1,j} - \overline{T}_{\sigma(k),j}\}$.
      4.1.5. **Poser** $u^*_{\sigma(k)} := l^*_{\sigma(k)}$.
5. **Appliquer** l'Algorithme II- 1.
   Entrée : $j, r_j, m_j, I_j, \{l^*_i\}_i, \{u^*_i\}_i, \{Z_{i,j}\}_i, \{a_{Z_{i,j},i,j}\}_{i \in I_j}$.
   Sortie : $\{\underline{T}_{i,j}\}_{i \in I_j}, T_{m_j+1,j}$.

L'Algorithme II- 5 permet de résoudre le Problème II- 8 et son équivalent, le Problème II- 7, par une méthode itérative. A chaque étape, on fixe la durée de l'opération de coût le plus élevé parmi les opérations dont la durée n'est pas encore fixée. La durée choisie pour cette opération d'indice σ est la plus petite des durées admissibles, c'est-à-dire une durée plus grande que sa durée minimale $l_{\sigma,j}$ et que la durée séparant la date $\overline{T}_{\sigma,j}$ de la date $\underline{T}_{\sigma+1,j}$. Le Théorème II- 6 montre qu'en faisant un tel choix à chaque étape on construit une solution optimale.

Exemple

Nous proposons un exemple d'application de l'Algorithme II- 5. Les données relatives au produit j à réaliser sont fournies dans le Tableau II- 7. Le produit peut être réalisé à partir de la date r_j=0 et avant la date d_j=12.

Tableau II- 7 : Données relatives à l'exécution de l'Algorithme II- 5

ENTREES					RESULTATS INTERMEDIAIRES												SORTIE
					Itération 1			Itération 2			Itération 3			Itération 4			
i	$l_{i,j}$	$u_{i,j}$	$s_{i,j}$	$[a_{z_{i,j},i,j}; b_{z_{i,j},i,j}]$	$\underline{T}_{i,j}$	$\overline{T}_{i,j}$	$l_{i,i}$ = $u_{i,j}$	$\underline{T}_{i,j}$	$\overline{T}_{i,j}$	$l_{i,i}$ = $u_{i,j}$	$\underline{T}_{i,j}$	$\overline{T}_{i,j}$	$l_{i,i}$ = $u_{i,j}$	$\underline{T}_{i,j}$	$\overline{T}_{i,j}$	$l_{i,i}$ = $u_{i,j}$	$T_{i,j}$
1	1	+∞	3	[2;4]	2	**3**	**1**	2	3	1	2	3	1	2	3	1	0
2	3	4	2	[0;8]	**3**	4		3	4		3	**4**	**3**	3	4	3	2
3	1	1	3	[5;9]	6	8	1	6	8	1	**6**	8	1	6	7	1	6
4	2	4	1	[7;+∞[	7	9		7	9		7	9		**7**	**8**		7
m_j=5	1	+∞	3	[11;+∞[	11	11		11	**11**	**1**	11	11	1	**11**	11	**1**	11
6					12	12		**12**	12		12	12		12	12		12

Les résultats de l'exécution sont explicités sur la Figure II- 12. Les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

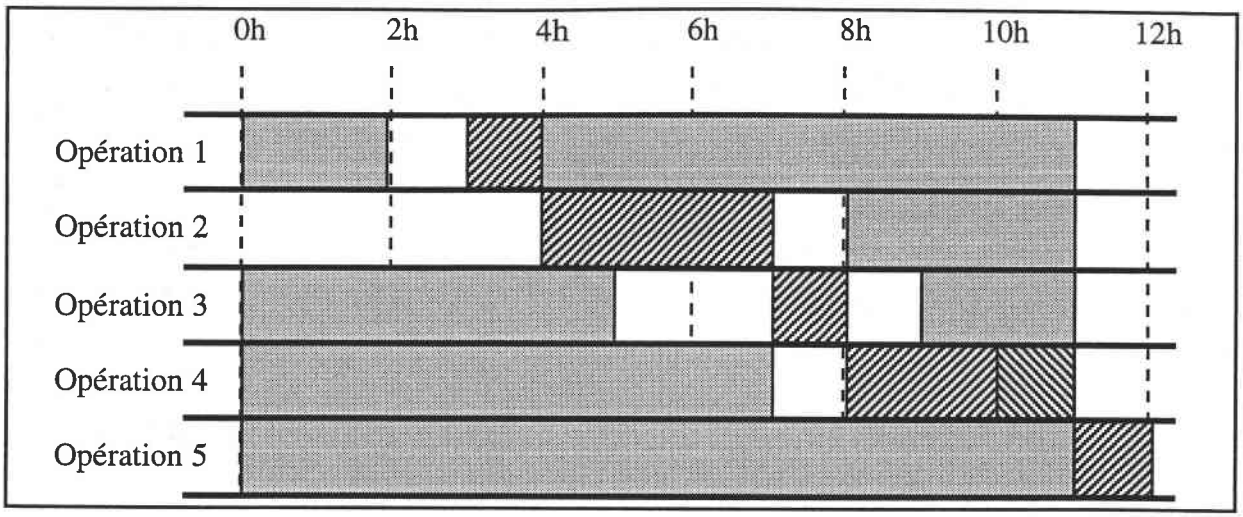

Figure II- 12 : Résultats fournis par l'Algorithme II- 5

Soit σ l'opération dont le coût d'utilisation des ressources est le plus élevé pour le produit j considéré parmi celles dont la durée opératoire peut varier, c'est-à-dire $s_{\sigma,j}=\max\{s_{i,j} \ / \ l_{i,j}\neq u_{i,j},\ 1\leq i\leq m_j\}$. Nous posons $P_\sigma=\max\{l_{\sigma,j};\underline{T}_{\sigma+1,j}-\overline{T}_{\sigma,j}\}$. Nous en déduisons immédiatement $-P_\sigma=\min\{-l_{\sigma,j};\overline{T}_{\sigma,j}-\underline{T}_{\sigma+1,j}\}$. Nous supposons qu'il existe une solution réalisable au Problème II- 8, et nous notons $\{T^*_{i,j}\}_{1\leq i\leq m_j+1}$ la solution optimale de ce problème. A partir de cette solution optimale, nous définissons la solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$:

- $T_{\sigma,j}=\min\{T^*_{\sigma+1,j}-P_\sigma;\overline{T}_{\sigma,j}\}$,
  i.e. $T_{\sigma,j}=\min\{T^*_{\sigma+1,j}-l_{\sigma,j};T^*_{\sigma+1,j}+\overline{T}_{\sigma,j}-\underline{T}_{\sigma+1,j};\overline{T}_{\sigma,j}\}$, (II-18)
- $T_{\sigma+1,j}=T_{\sigma,j}+P_\sigma$,
  i.e. $T_{\sigma+1,j}=\min\{T^*_{\sigma+1,j};\overline{T}_{\sigma,j}+P_\sigma\}$, (II-19)
- $T_{i,j}=\min\{T_{i+1,j}+T^*_{i,j}-T^*_{i+1,j};\overline{T}_{i,j}\}$, pour $1\leq i<\sigma$, (II-20)
- $T_{i,j}=\max\{T_{i-1,j}+T^*_{i,j}-T^*_{i-1,j};\underline{T}_{i,j}\}$, pour $\sigma+1<i\leq m_j+1$. (II-21)

Cette solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ vérifie $T_{\sigma+1,j}-T_{\sigma,j}=P_\sigma=\max\{l_{\sigma,j};\underline{T}_{\sigma+1,j}-\overline{T}_{\sigma,j}\}$.

Théorème II- 6

La solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ définie par les relations (II-18) à (II-21) est optimale pour le Problème II- 8 et le Problème II- 7.

Démonstration

a) Nous montrons tout d'abord la relation (II-22) : $T_{i,j}\geq T^*_{i,j}$, pour $1\leq i\leq\sigma$.

On a $T^*_{\sigma+1,j}-l_{\sigma,j}\geq T^*_{\sigma,j}$, car la contrainte (II-2) est vérifiée par la solution $\{T^*_{i,j}\}_{1\leq i\leq m_j+1}$. De plus, selon (II-16), on a $T^*_{\sigma+1,j}-\underline{T}_{\sigma+1,j}\geq 0$. Par conséquent, $T_{\sigma,j}=\min\{T^*_{\sigma+1,j}-l_{\sigma,j}\ ;\ \overline{T}_{\sigma,j}+(T^*_{\sigma+1,j}-\underline{T}_{\sigma+1,j})\ ;\ \overline{T}_{\sigma(i),j}\}\geq\min\{T^*_{\sigma,j}\ ;\ \overline{T}_{\sigma,j}\}$. Puisque $\overline{T}_{\sigma,j}\geq T^*_{\sigma,j}$ d'après la contrainte (II-17), on en déduit $T_{\sigma,j}\geq T^*_{\sigma,j}$.

Supposons que $T_{i+1,j} \geq T^*_{i+1,j}$, pour $1 \leq i < \sigma$. Sous cette hypothèse, on a $T_{i,j} = \min\{T^*_{i,j} + (T_{i+1,j} - T^*_{i+1,j}); \overline{T}_{i,j}\} \geq \min\{T^*_{i,j}; \overline{T}_{i,j}\}$. Puisque $\overline{T}_{i,j} \geq T^*_{\sigma,j}$ d'après la contrainte (II-17), on a $T_{i,j} \geq T^*_{i,j}$.

Par conséquent, $T_{i,j} \geq T^*_{i,j}$, pour $1 \leq i \leq \sigma$. (II-22)

b) Nous montrons la relation (II-23) : $T_{i,j} \leq T^*_{i,j}$, pour $\sigma+1 \leq i \leq m_j+1$.

On a $T_{\sigma+1,j} = \min\{T^*_{\sigma+1,j}; \overline{T}_{\sigma,j} + P_\sigma\} \leq T^*_{\sigma+1,j}$.

Supposons que $T_{i-1,j} \leq T^*_{i-1,j}$, pour $\sigma+1 < i \leq m_j+1$. Sous cette hypothèse, on a $T_{i,j} = \max\{T^*_{i,j} + (T_{i-1,j} - T^*_{i-1,j}); \underline{T}_{i,j}\} \leq \max\{T^*_{i,j}; \underline{T}_{i,j}\}$. Puisque $\underline{T}_{i,j} \leq T^*_{\sigma,j}$ d'après la contrainte (II-16), on a $T_{i,j} \leq T^*_{i,j}$.

Par conséquent, $T_{i,j} \leq T^*_{i,j}$, pour $\sigma+1 \leq i \leq m_j+1$ (II-23)

c) Nous montrons ensuite que $\{T_{i,j}\}_{1 \leq i \leq m_j+1}$ vérifie les relations (II-16).

D'après (II-22), on a $T_{i,j} \geq T^*_{i,j}$, pour $1 \leq i \leq \sigma$. Comme la contrainte (II-16) est vérifiée par la solution $\{T^*_{i,j}\}_{1 \leq i \leq m_j+1}$ (i.e. $T^*_{i,j} \geq \underline{T}_{i,j}$), on en déduit $T_{i,j} \geq \underline{T}_{i,j}$, pour $1 \leq i \leq \sigma$.

Puisque $P_\sigma \geq \underline{T}_{\sigma+1,j} - \overline{T}_{\sigma,j}$, on a $T_{\sigma+1,j} \geq \min\{T^*_{\sigma+1,j}; \overline{T}_{\sigma,j} + (\underline{T}_{\sigma+1,j} - \overline{T}_{\sigma,j})\}$ i.e. $T_{\sigma+1,j} \geq \min\{T^*_{\sigma+1,j}; \underline{T}_{\sigma+1,j}\}$. Comme la contrainte (II-6) est vérifiée par la solution $\{T^*_{i,j}\}_{1 \leq i \leq m_j+1}$ (i.e. $T^*_{\sigma+1,j} \geq \underline{T}_{\sigma+1,j}$), on en déduit $T_{\sigma+1,j} \geq \underline{T}_{\sigma+1,j}$.

Par définition, on a $T_{i,j} = \max\{T_{i-1,j} + T^*_{i,j} - T^*_{i-1,j}; \underline{T}_{i,j}\} \geq \underline{T}_{i,j}$, pour $\sigma+1 < i \leq m_j+1$.

Par conséquent, la solution $\{T_{i,j}\}_{1 \leq i \leq m_j+1}$ vérifie la contrainte (II-16), i.e. $T_{i,j} \geq \underline{T}_{i,j}$, pour $1 \leq i \leq m_j+1$.

d) Nous montrons que $\{T_{i,j}\}_{1 \leq i \leq m_j+1}$ vérifie les relations (II-17).

Par définition, on a $T_{i,j} = \min\{T_{i+1,j} + T^*_{i,j} - T^*_{i+1,j}; \overline{T}_{i,j}\} \leq \overline{T}_{i,j}$, pour $1 \leq i < \sigma$.

Par définition, on a $T_{\sigma,j} = \min\{T^*_{\sigma+1,j} - P_\sigma; \overline{T}_{\sigma,j}\} \leq \overline{T}_{\sigma,j}$.

D'après (II-23), on a $T_{i,j} \leq T^*_{i,j}$, pour $\sigma+1 \leq i \leq m_j$. Comme la contrainte (II-17) est vérifiée par la solution $\{T^*_{i,j}\}_{1 \leq i \leq m_j+1}$ (i.e. $T^*_{i,j} \leq \overline{T}_{i,j}$), on en déduit $T_{i,j} \leq \overline{T}_{i,j}$, pour $\sigma+1 \leq i \leq m_j$.

Par conséquent, la solution $\{T_{i,j}\}_{1 \leq i \leq m_j+1}$ vérifie la contrainte (II-17), i.e. $T_{i,j} \leq \overline{T}_{i,j}$, pour $1 \leq i \leq m_j+1$.

e) Nous montrons que $\{T_{i,j}\}_{1\le i\le m_j+1}$ vérifie les relations (II-2).

Pour $1\le i<\sigma$, $T_{i+1,j}-T_{i,j}=\max\{T^*_{i+1,j}-T^*_{i,j};T_{i+1,j}-\overline{T}_{i,j}\}$. Comme la contrainte (II-2) est vérifiée par la solution $\{T^*_{i,j}\}_{1\le i\le m_j+1}$ (i.e. $l_{i,j}\le T^*_{i+1,j}-T^*_{i,j}\le u_{i,j}$), on en déduit $T_{i+1,j}-T_{i,j}\ge T^*_{i+1,j}-T^*_{i,j}\ge l_{i,j}$, pour $1\le i<\sigma$.

Puisque la contrainte (II-2) est vérifiée pour les solutions $\{T^*_{i,j}\}_{1\le i\le m_j+1}$ et $\{\overline{T}_{i,j}\}_{1\le i\le m_j+1}$, on a $u_{i,j}\ge\max\{T^*_{i+1,j}-T^*_{i,j};\overline{T}_{i+1,j}-\overline{T}_{i,j}\}$. D'après (II-17), on a $\overline{T}_{i+1,j}\ge T_{i+1,j}$ ce qui conduit à $u_{i,j}\ge\max\{T^*_{i+1,j}-T^*_{i,j};T_{i+1,j}-\overline{T}_{i,j}\}=T_{i+1,j}-T_{i,j}$, pour $1\le i<\sigma$.

On a $T_{\sigma+1,j}-T_{\sigma,j}=P_\sigma\le l_{\sigma,j}$.

D'après (II-22), on a $T_{\sigma,j}\ge T^*_{\sigma,j}$ et d'après (II-23), on a $T_{\sigma+1,j}\le T^*_{\sigma+1,j}$. Par conséquent, on en déduit $T^*_{\sigma+1,j}-T^*_{\sigma,j}\ge T_{\sigma+1,j}-T_{\sigma,j}$. La contrainte (II-2) étant vérifiée pour la solution $\{T^*_{i,j}\}_{1\le i\le m_j+1}$, $u_{\sigma,j}\ge T^*_{\sigma+1,j}-T^*_{\sigma,j}\ge T_{\sigma+1,j}-T_{\sigma,j}$.

Pour $\sigma+1\le i\le m_j$, $T_{i+1,j}-T_{i,j}=\max\{T^*_{i+1,j}-T^*_{i,j};\underline{T}_{i+1,j}-T_{i,j}\}$. Comme la contrainte (II-2) est vérifiée par la solution $\{T^*_{i,j}\}_{1\le i\le m_j+1}$ (i.e. $l_{i,j}\le T^*_{i+1,j}-T^*_{i,j}\le u_{i,j}$), on en déduit $T_{i+1,j}-T_{i,j}\ge T^*_{i+1,j}-T^*_{i,j}\ge l_{i,j}$, pour $1\le i<\sigma$.

Puisque la contrainte (II-2) est vérifiée pour les solutions $\{T^*_{i,j}\}_{1\le i\le m_j+1}$ et $\{\underline{T}_{i,j}\}_{1\le i\le m_j+1}$, on a $u_{i,j}\ge\max\{T^*_{i+1,j}-T^*_{i,j};\underline{T}_{i+1,j}-\underline{T}_{i,j}\}$. D'après (II-17), on a $\overline{T}_{i+1,j}\ge T_{i+1,j}$ ce qui conduit à $u_{i,j}\ge\max\{T^*_{i+1,j}-T^*_{i,j};T_{i+1,j}-\overline{T}_{i,j}\}=T_{i+1,j}-T_{i,j}$, pour $1\le i<\sigma$.

Par conséquent, la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ vérifie la contrainte (II-2), c'est-à-dire $l_{i,j}\le T_{i+1,j}-T_{i,j}\le u_{i,j}$, pour $1\le i\le m_j$.

f) Nous montrons enfin que la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ est optimale.

Pour cela, nous comparons la valeur de la fonction objectif de la solution $\{T_{i,j}\}_{1\le i\le m_j+1}$ à celle de la solution optimale $\{T^*_{i,j}\}_{1\le i\le m_j+1}$. Nous notons Δ cette différence :

$$\Delta=\sum_{1\le i\le m_j}s_{i,j}\left(T_{i+1,j}-T_{i,j}\right)-\sum_{1\le i\le m_j}s_{i,j}\left(T^*_{i+1,j}-T^*_{i,j}\right),$$

$$\Delta=\sum_{\left\{\begin{array}{l}1\le i\le m_j\ \text{t.q.}\\ i\ne\sigma\\ s_{i,j}>s_{\sigma,j}\end{array}\right\}\cup\{i=\sigma\}\cup\left\{\begin{array}{l}1\le i\le m_j\ \text{t.q.}\\ i\ne\sigma\\ s_{i,j}\le s_{\sigma,j}\end{array}\right\}}\left[s_{i,j}\left[\left(T_{i+1,j}-T_{i,j}\right)-\left(T^*_{i+1,j}-T^*_{i,j}\right)\right]\right].$$

Or, puisque $s_{\sigma,j}=\max\{s_{i,j}\ /\ l_{i,j}\ne u_{i,j},\ 1\le i\le m_j\}$, pour $i\ne\sigma$ et $s_{i,j}>s_{\sigma,j}$, nous avons $\left[\left(T_{i+1,j}-T_{i,j}\right)-\left(T^*_{i+1,j}-T^*_{i,j}\right)\right]=[l_{i,j}-l_{i,j}]=0$. De plus, d'après (II-20) et (II-21), nous avons $\left[\left(T_{i+1,j}-T_{i,j}\right)-\left(T^*_{i+1,j}-T^*_{i,j}\right)\right]\ge 0$ pour $i\ne\sigma$. Par conséquent, nous en déduisons

$$\Delta\le s_{\sigma,j}\sum_{\{i=\sigma\}\cup\left\{\begin{array}{l}1\le i\le m_j\ \text{t.q.}\\ i\ne\sigma\\ s_{i,j}\le s_{\sigma,j}\end{array}\right\}}\left[\left(T_{i+1,j}-T_{i,j}\right)-\left(T^*_{i+1,j}-T^*_{i,j}\right)\right]\le s_{\sigma,j}\sum_{1\le i\le m_j}\left[\left(T_{i+1,j}-T_{i,j}\right)-\left(T^*_{i+1,j}-T^*_{i,j}\right)\right],$$

et $\Delta\le s_{\sigma(i),j}\left(T_{m_j+1,j}-T^*_{m_j+1,j}-T_{1,j}+T^*_{1,j}\right)$. Selon la relation (II-22), $-T_{1,j}+T^*_{1,j}\le 0$ et selon la

relation (II-23), $T_{m_j+1,j}-T^*_{m_j+1,j}\leq 0$, par conséquent on a $\Delta\leq 0$. On en déduit que la solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ n'est pas moins bonne que la solution optimale $\{T^*_{i,j}\}_{1\leq i\leq m_j+1}$.

Par conséquent, la solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ vérifiant les contraintes (II-16), (II-17), (II-2) et (II-6), est réalisable pour le Problème II- 8. De plus, sa fonction objectif est égale à celle de la solution optimale, elle est donc également optimale pour le Problème II- 8 et son équivalent le Problème II- 7.

CQFD

Théorème II- 7

L'Algorithme II- 5 détermine les dates de début et de fin d'exécution de toutes les opérations induisant un coût minimal d'utilisation des ressources, i.e. il fournit une solution optimale au Problème II- 7 et au Problème II- 8.

Démonstration

a) Nous montrons que s'il existe une solution au Problème II- 7, l'Algorithme II- 5 fournit une solution optimale de ces problèmes.

A chaque itération du pas 4, nous fixons la durée d'une opération. Nous choisissons l'opération de coût le plus élevé parmi les opérations dont la durée n'est pas encore fixée. L'indice de cette opération est σ, et on a $s_{\sigma,j}=\max\{s_{i,j} \;/\; l_{i,j}\neq u_{i,j},\; 1\leq i\leq m_j\}$.

D'après le Théorème II- 6, s'il existe une solution au Problème II- 7 (et au Problème II- 8), la solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ définie par les relations (II-18) à (II-21) est optimale pour ces problèmes. Cela signifie qu'il existe une solution optimale de ces problèmes vérifiant $T_{\sigma+1,j}-T_{\sigma,j}=P_\sigma=\max\{l_{\sigma,j};\underline{T}_{\sigma+1,j}-\overline{T}_{\sigma,j}\}$. Cette valeur est donc choisie comme durée de l'opération σ.

Pour ce faire, nous posons $u_\sigma=l_\sigma=\max\{l_{\sigma,j};\underline{T}_{\sigma+1,j}-\overline{T}_{\sigma,j}\}$ aux pas 4.1.4 et 4.1.5. Ainsi, prenant en compte la contrainte (II-2), nous avons dans la suite $T_{\sigma+1,j}-T_{\sigma,j}=u_\sigma=l_\sigma$. La durée de l'opération σ est par conséquent optimale dans toute la suite de l'algorithme.

b) Nous montrons que l'Algorithme II- 5 s'arrête sans fournir de solution si et seulement si le Problème II- 7 n'a pas de solution réalisable.

Toute solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ réalisable pour un des Problème II- 7 et Problème II- 8 est réalisable pour l'autre. Une telle solution réalisable vérifie $\underline{T}_{i,j}\leq T_{i,j}\leq\overline{T}_{i,j}$ pour $1\leq i\leq m_j+1$. Par conséquent, s'il existe $i_0\in\{1, 2, \ldots, m_j+1\}$ tel que $\underline{T}_{i_0,j}>\overline{T}_{i_0,j}$ (condition d'arrêt au pas 4.1.3.2), alors il n'existe pas de solution réalisable pour les deux problèmes.

Inversement, si $\underline{T}_{i,j} \leq \overline{T}_{i,j}$ pour $1 \leq i \leq m_j+1$, alors les solutions $\{\underline{T}_{i,j}\}_{1 \leq i \leq m_j+1}$ et $\{\overline{T}_{i,j}\}_{1 \leq i \leq m_j+1}$ sont réalisables pour le Problème II- 7 et le Problème II- 8. Par conséquent, il existe au moins une solution réalisable pour ces problèmes.

CQFD

Complexité

Le tri au pas 2 de l'Algorithme II- 5 requiert $O(m_j \log_2 m_j)$ [Beauquier et *al.*, 1992] où m_j est le nombre d'opérations à exécuter sur le produit j. Les pas 3.1 à 3.2 nécessitent $2m_j$ affectations. Le pas 4 est itéré m_j fois. Par conséquent, les pas 4.1.1, 4.1.2, 4.1.3, 4.1.4 et 4.1.5 sont exécutés au plus m_j fois. A chaque nouvelle itération des pas 4.1.1 et 4.1.2, l'Algorithme II- 1 et l'Algorithme II- 3 sont appliqués. Chacun d'eux nécessite $O(m_j)$ opérations. Les pas 4.1.3 à 4.1.3.2.1 sont exécutés au plus une fois et requièrent de l'ordre de $O(m_j)$ opérations. Les pas 4.1.4 et 4.1.5 sont immédiats. Enfin, l'Algorithme II- 1 est appliqué au pas 5, une dernière fois. Par conséquent, **la complexité de l'Algorithme II- 5 est en $O(m_j^2)$.**

Si, au pas 4.1.4 de l'Algorithme II- 5, on a $\underline{T}_{\sigma+1,j} - \underline{T}_{\sigma,j} = \max\{l_{\sigma,j} \; ; \; \underline{T}_{\sigma+1,j} - \overline{T}_{\sigma,j}\}$, il est alors inutile de remettre à jour les valeurs $\underline{T}_{i,j}$. En effet, dans ce cas, fixer la durée de l'opération σ ne modifie pas les dates au plus tôt. Par conséquent, il ne sert à rien d'exécuter l'Algorithme II-1 au prochain passage au pas 4.1.1. De manière analogue, si les dates au plus tard vérifient $\overline{T}_{\sigma+1,j} - \overline{T}_{\sigma,j} = \max\{l_{\sigma,j} \; ; \; \underline{T}_{\sigma+1,j} - \overline{T}_{\sigma,j}\}$, il ne sert à rien d'exécuter l'Algorithme II- 3 au prochain passage au pas 4.1.2. Ces deux modifications permettent de réduire la durée d'exécution de l'Algorithme II- 5.

Remarque

Au pas 2 de l'Algorithme II- 5, nous trions les opérations dans l'ordre décroissant des coûts d'utilisation des ressources $s_{i,j}$. L'algorithme fixe ensuite les durées opératoires à partir de cet ordre. Dans cet ordre, les premières opérations ont des durées les plus petites possibles au détriment des opérations ultérieures. En cas d'égalité des coûts d'utilisation des ressources entre deux opérations, les opérations sont placées dans un ordre arbitraire.

II.5.4. Sélection des fenêtres

L'objectif de cette section est de résoudre le Problème II- 6. Autrement dit, nous allons déterminer dans quelle fenêtre $Z_{i,j}$ et à quelle date $T_{i,j}$ l'opération i doit être effectuée de manière à ce que le coût d'utilisation des ressources pour fabriquer le produit j soit minimal.

être exécutées est réduit, c'est pourquoi nous proposons une autre approche pour résoudre le Problème II- 6 requiérant une discrétisation du temps.

Nous supposons dans cette approche que les durées des opérations ainsi que les dates de début et de fin des fenêtres sont des valeurs rationnelles, c'est-à-dire pouvant s'écrire comme un rapport entre deux entiers. Tel est le cas dans la pratique car les ordinateurs ne permettent pas de prendre en compte des valeurs irrationnelles. Les valeurs r_j, d_j, $l_{i,j}$, et $a_{z,i,j}$ sont éléments de Q^+, tandis que $u_{i,j}$ et $b_{z,i,j}$ appartiennent à $Q^+ \cup \{+\infty\}$, où Q^+ est l'ensemble des nombres rationnels positifs. Nous multiplions toutes ces valeurs par le plus petit multiple commun de leur dénominateur (nombre qui peut éventuellement être grand). Nous pouvons donc transformer notre problème de manière à n'avoir que des dates et des durées entières. Sans perte de généralité, et pour simplifier la présentation, nous admettons donc que ces valeurs sont entières.

L'approche que nous proposons est basée sur la programmation dynamique. A chaque opération à exécuter sur le produit correspond une étape de la programmation dynamique. Nous définissons $(m_j + 1)$ étapes correspondant à une étape d'initialisation et aux m_j étapes marquant la fin de chacune des opérations à réaliser sur le produit j.

Les états du système à chaque étape correspondent à toutes les valeurs possibles pour le début de l'opération considérée. L'état $T_{i,j}$ correspond donc instant de début d'exécution de l'opération i sur le produit j. Puisque le produit ne peut commencer avant la date r_j et doit être terminé avant la date d_j (voir contraintes (II-1) et (II-15)), nous avons :

$r_j \leq T_{i,j} \leq d_j$, pour $1 \leq i \leq m_j$. (II-24)

Comme les données sont entières, le nombre de dates possibles (i.e. d'états à chaque étape) est égal à $d_j - r_j + 1$.

La durée de chaque opération i est contrôlable et peut être choisie dans l'intervalle $[l_{i,j};u_{i,j}]$ (voir contrainte (II-2)). Ainsi, les dates de début de l'opération i qui autorisent une fin à la date $T_{i+1,j}$ sont comprises entre $T_{i+1,j} - u_{i,j}$ et $T_{i+1,j} - l_{i,j}$. Si nous décidons de commencer l'opération i à l'instant $T_{i,j}$ et de la terminer à l'instant $T_{i+1,j}$, deux cas peuvent se présenter :

a) l'intervalle $[T_{i,j} ; T_{i+1,j}]$ est inclus dans une période de disponibilité des ressources, $[a_{z,i,j};b_{z,i,j}]$ où $1 \leq z \leq y_{i,j}$; dans ce cas, nous associons à l'état $T_{i+1,j}$ un coût correspondant à l'utilisation des ressources durant cette période.

b) l'intervalle $[T_{i,j} ; T_{i+1,j}]$ n'est pas inclus dans une période de disponibilité des ressources ; dans ce cas, nous associons à l'état $T_{i+1,j}$ un coût infini.

Parmi toutes les dates possibles $T_{i,j}$, nous retenons la meilleure date (au sens du coût qu'elle induit) permettant de terminer l'opération i à la date $T_{i+1,j}$. Nous notons $S(T_{i+1,j})$ l'ensemble des dates permettant de terminer l'opération i à la date $T_{i+1,j}$. Formellement, nous avons :

$S(T_{i+1,j}) = \{t \,/\, r_j \leq t \leq d_j,\ T_{i+1,j} - l_{i,j} \leq t \leq T_{i+1,j} - u_{i,j}$, et $\exists 1 \leq z \leq y_{i,j}$ tq. $[t;T_{i+1,j}] \subseteq [a_{z,i,j};b_{z,i,j}]\}$. (II-25)

Dans la Figure II- 13, chaque état est représenté par un disque noir. Les états d'une même étape sont alignés horizontalement. Nous donnons sur cette figure les états de deux étapes successives. Deux états, $T_{i,j}$=4 et $T_{i+1,j}$=7, sont distingués. Ils correspondent au début et à la fin d'exécution de l'opération i. $S(T_{i+1,j})$ est l'ensemble des états qui permettent d'atteindre l'état $T_{i+1,j}$.

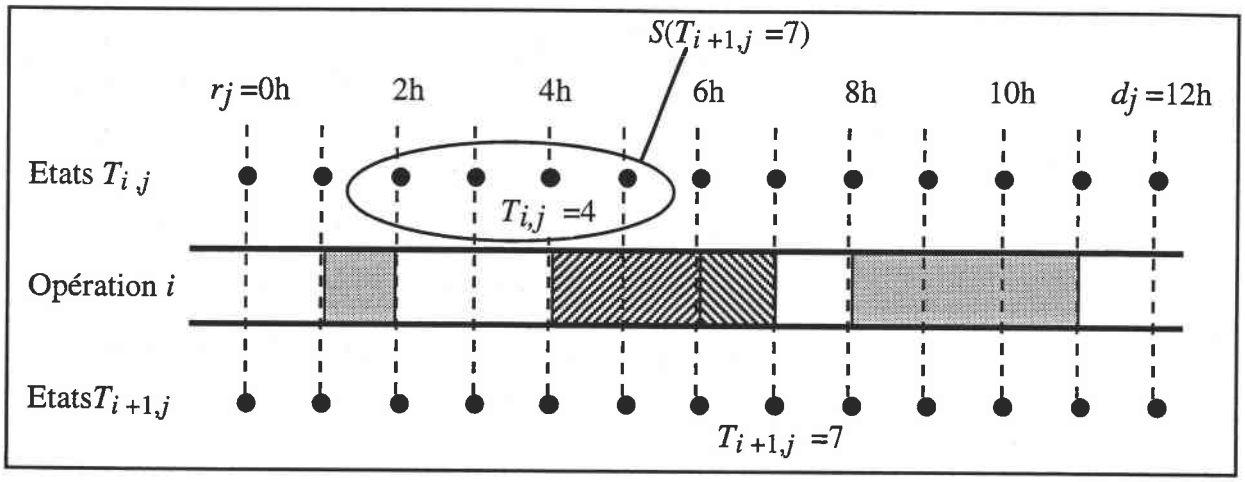

Figure II- 13 : Représentation des états de deux étapes successives

A partir des états de $S(T_{i+1,j})$, nous pouvons déterminer le coût d'utilisation des ressources si l'opération i se termine à la date $T_{i+1,j}$. Nous évaluons chaque état $T_{i+1,j}$ par le coût, noté $C(T_{i+1,j})$. Nous proposons la relation de récurrence suivante :

$$C(T_{1,j})=0, \text{ pour } r_j \le T_{1,j} \le d_j.$$

$$C\left(T_{i+1,j}\right)=\begin{cases} +\infty & \text{si } S\left(T_{i+1,j}\right)=\varnothing \\ \min\limits_{T_{i,j}\in S\left(T_{i+1,j}\right)}\left\{C\left(T_{i,j}\right)+s_{i,j}\left(T_{i+1,j}-T_{i,j}\right)\right\} & \text{sinon} \end{cases}$$

$$\text{pour } r_j \le T_{i+1,j} \le d_j,\ 1 \le i \le m_j. \qquad \text{(II-26)}$$

Un état $T_{i,j}$ qui ne correspond pas à une solution réalisable (i.e. $S(T_{i,j})\neq\varnothing$) a un coût infini. Pour chaque état $T_{i,j}$, l'évaluation $C(T_{i,j})$ donne le coût minimal d'utilisation des ressources permettant de réaliser les $(i-1)$ premières opérations et terminer l'opération $(i-1)$ à la date $T_{i,j}$. La date $T_{m_j+1,j}$ pour laquelle $C(T_{m_j+1,j})$ est minimal est la date de fin de fabrication du produit permettant de fabriquer le produit à moindre coût.

La Figure II- 14 donne les évaluations $C(T_{i,j})$ et $C(T_{i+1,j})$ des états de deux étapes successives. Le coût d'utilisation de la ressource pour l'opération i est $s_{i,j}$=0,1. Cette opération peut être exécutée dans l'une des fenêtres {[0;5], [2;7], [6;8]}. Sa durée minimale est $l_{i,j}$=2 et sa durée maximale est $u_{i,j}$=+∞.

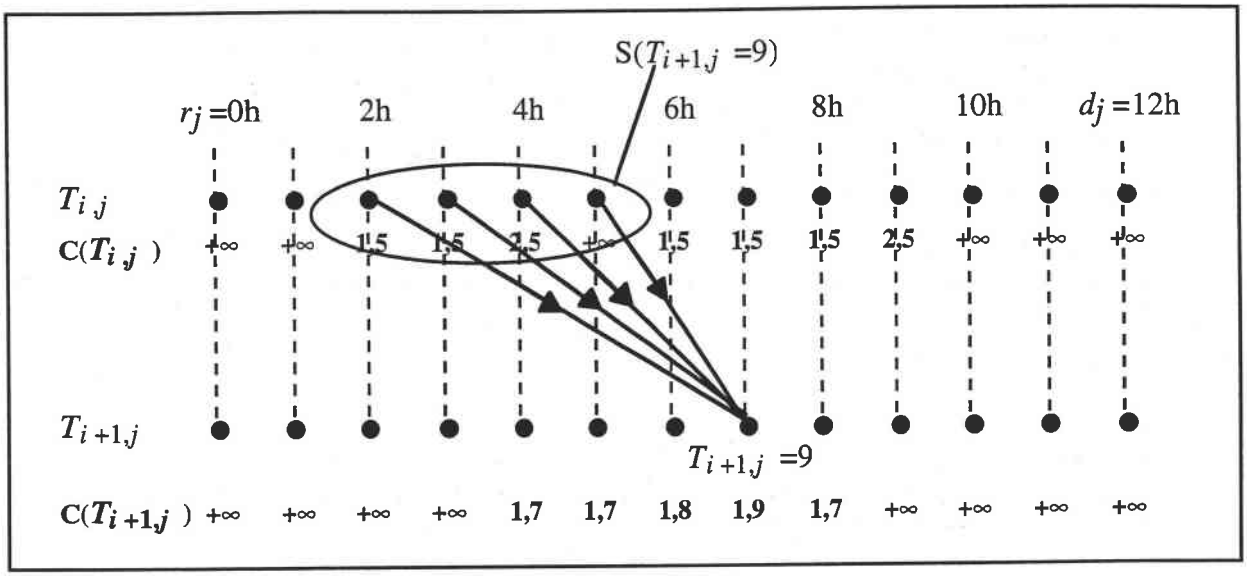

Figure II- 14 : Représentation des états de deux étapes successives

Si $\{T_{i,j}\}_{1 \le i \le m_j+1}$ est la solution optimale du Problème II- 6, on a $\sum_{1 \le i \le m_j} s_{i,j}\left(T_{i+1,j} - T_{i,j}\right) = \min_{r_j \le T_{m_j+1,j} \le d_j} \left\{C(T_{m_j+1,j})\right\}$. L'Algorithme II- 6 découle des relations (II-24) à (II-26) et permet de résoudre le Problème II- 6.

Algorithme II- 6 : Choix des fenêtres pour un coût d'utilisation minimal

Procédure Définition des fenêtres induisant un coût d'utilisation des ressources minimal pour le produit j.

Entrée : j est l'indice du produit considéré.

r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.

d_j est la date avant laquelle on doit livrer le produit j.

m_j est le nombre d'opérations à réaliser sur le produit j.

I_j est l'ensemble des opérations à réaliser sur le produit j.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$y_{i,j}$ est le nombre de fenêtres dans lesquelles on peut réaliser l'opération i sur le produit j, pour $i \in I_j$.

$a_{z,i,j}$ est la date de début de la z-ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \le z \le y_{i,j}$, $i \in I_j$.

$b_{z,i,j}$ est la date de fin de la z-ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \le z \le y_{i,j}$, $i \in I_j$.

Sortie : $T_{i,j}$ est la date de début de l'opération i effectuée sur le produit j, où $i \in I_j$.

$T_{m_j+1,j}$ est la date de fin du produit j.

$Z_{i,j}$ est la fenêtre dans laquelle on doit réaliser l'opération i sur le produit j, où $i \in I_j$.

1. **Initialisation.**

1.1. **Trier** les fenêtres $[a_{z,i,j}\ ;\ b_{z,i,j}]$ pour $1 \le z \le y_{i,j}$, $i \in I_j$, dans l'ordre croissant des $a_{z,i,j}$ et, en cas d'égalité, dans l'ordre croissant des $b_{z,i,j}$.
Dans la suite de l'algorithme, la z-ième fenêtre est le z-ième élément dans cet ordre. $1 \le z \le y_{i,j}$.

1.2. **Pour** $T_{1,j}$ allant de r_j à d_j : *On énumère chaque état $T_{1,j}$ de la 1ère étape.*

1.2.1. **Poser** $C(T_{1,j}):=0$. *On fixe le coût de chaque état $T_{1,j}$ à 0.*

2. **Pour** i allant de 1 à m_j : *On note i l'indice de l'étape examinée.*

2.1. **Pour** $T_{i+1,j}$ allant de r_j à d_j : *On énumère chaque état $T_{i+1,j}$ de l'étape i.*

2.1.1. **Poser** $C(T_{i+1,j}):=+\infty$. *On initialise le coût de chaque état $T_{i+1,j}$ à $+\infty$.*

2.1.2. **Poser** $z:=\min\{z \ / \ 1\leq z\leq y_{i,j},\ b_{z,i,j}\geq T_{i+1,j}\}$. *On note z la fenêtre commençant le plus tôt possible et permettant de réaliser une opération qui se terminerait à la date $T_{i+1,j}$.*

2.1.3. **Pour** $T_{i,j}$ allant de $\max\{r_j \ ; \ a_{z,i,j} \ ; \ T_{i+1,j}-u_{i,j}\}$ à $\min\{d_j \ ; \ T_{i+1,j}-l_{i,j}\}$: *On examine chaque état $T_{i,j}$ de $S(T_{i+1,j})$.*

2.1.3.1. **Si** ($C(T_{i+1,j})>C(T_{i,j})+(T_{i+1,j}-T_{i,j})\ s_{i,j}$) **alors** : *L'état $T_{i,j}$ permet de diminuer le coût.*

2.1.3.1.1. **Poser** $C(T_{i+1,j}):=C(T_{i,j})+(T_{i+1,j}-T_{i,j})\ s_{i,j}$.

2.1.3.1.2. **Poser** $Sopt(T_{i+1,j}):=T_{i,j}$. *On désigne par $Sopt(T_{i+1,j})$ l'état de $S(T_{i+1,j})$ qui permet d'obtenir le coût $C(T_{i+1,j})$ le plus petit.*

3. **Poser** $Copt:=+\infty$. *On note $Copt$ le meilleur coût trouvé pour un état $T_{m_j+1,j}$ de la dernière étape.*

4. **Pour** $T_{m_j+1,j}$ allant de r_j à d_j : *On énumère chaque état $T_{m_j+1,j}$.*

4.1. **Si** $(Copt>C(T_{m_j+1,j}))$ **alors** : *L'état $T_{m_j+1,j}$ a un meilleur coût.*

4.1.1. **Poser** $Copt:=C(T_{m_j+1,j})$.

4.1.2. **Poser** $Topt:=T_{m_j+1,j}$. *On choisit comme date de fin du produit celle de coût minimal.*

5. **Reconstitution de la solution.**

5.1. **Si** $(Copt=+\infty)$ **alors** : *Aucun état de la dernière étape n'est atteignable.*

5.1.1. **Quitter**. *Il n'existe aucune solution réalisable.*

5.2. **Sinon** :

5.2.1. **Poser** $T_{m_j+1,j}:=Topt$.

5.2.2. **Pour** i allant de m_j à 1 : *On note i l'indice de l'étape considérée.*

5.2.2.1. **Poser** $T_{i,j}:=Sopt(T_{i+1,j})$.

5.2.2.2. **Poser** $Z_{i,j}:=\min\{z \ / \ 1\leq z\leq y_{i,j},\ b_{z,i,j}\geq T_{i+1,j}\}$.

Exemple

Tableau II- 8 : Données fournies à l'Algorithme II- 6

ENTREES				
Indice de l'opération i	Durée minimale $l_{i,j}$	Durée maximale $u_{i,j}$	Coût opératoire $s_{i,j}$	Périodes durant lesquelles on peut réaliser l'opération i $\{[a_{z,i,j};b_{z,i,j}] \ / \ 1\leq z\leq y_{i,j}\}$
1	1	$+\infty$	3	[0;1], [2;4], [5;9], [11;$+\infty$[
2	3	4	2	[0;8], [11;$+\infty$[
3	1	1	3	[0;1], [2;4], [5;9], [11;$+\infty$[
4	2	4	1	[0;3], [7;$+\infty$[
m_j=5	1	$+\infty$	3	[0;1], [2;4], [5;9], [11;$+\infty$[
m_j+1=6				

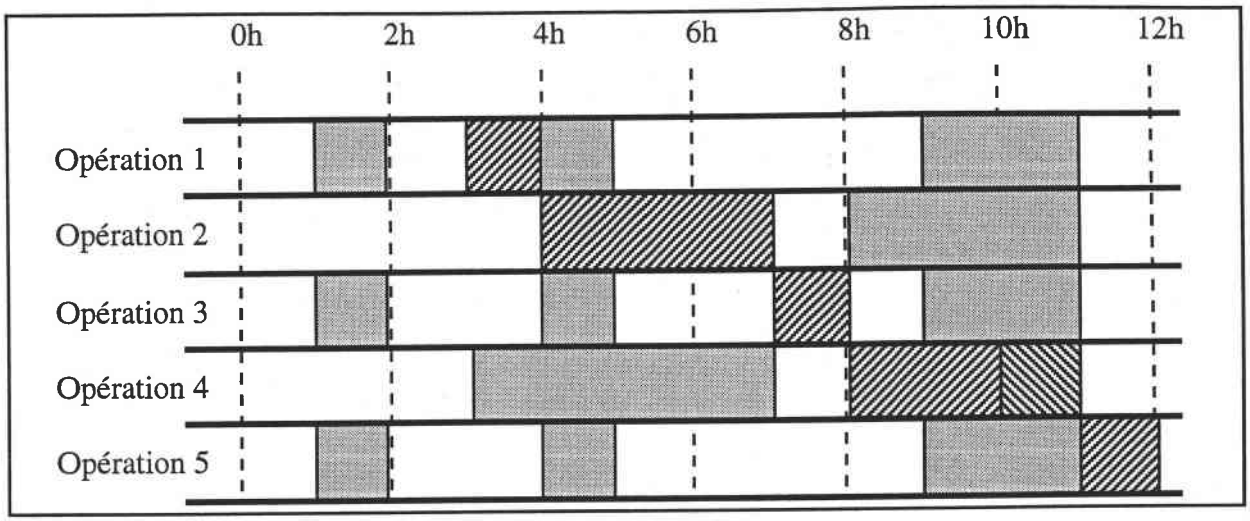

Figure II- 15 : Résultats fournis par l'Algorithme II- 6

Tableau II- 9 : Résultats de l'exécution de l'Algorithme II- 6

RESULTATS INTERMEDIAIRES												SORTIE
Init. Etat $T_{1,j}$		Itération 1 Etat $T_{2,j}$		Itération 2 Etat $T_{3,j}$		Itération 3 Etat $T_{4,j}$		Itération 4 Etat $T_{5,j}$		Itération 5 Etat $T_{6,j}$		
T	C	C	*Sopt*	C	*Sopt*	C	*Sopt*	C	*Sopt*	C	*Sopt*	$T_{i,j}$
r_j=0	0	+∞		+∞		+∞		+∞		+∞		
1	0	3	0	+∞		+∞		+∞		+∞		
2	0	+∞		+∞		+∞		+∞		+∞		
3	**0**	3	2	+∞		+∞		+∞		+∞		$T_{1,j}$
4	0	**3**	**3**	9	1	+∞		+∞		+∞		$T_{2,j}$
5	0	+∞		11	1	+∞		+∞		+∞		
6	0	3	5	9	3	14	5	+∞		+∞		
7	0	3	6	9	4	12	6	+∞		+∞		
8	0	3	7	**11**	**4**	12	7	+∞		+∞		$T_{3,j}$
9	0	3	8	+∞		**14**	**8**	14	7	+∞		$T_{4,j}$
10	0	+∞		+∞		+∞		14	8	+∞		
11	0	+∞		+∞		+∞		**16**	**9**	+∞		$T_{5,j}$
d_j=12	0	3	11	+∞		+∞		+∞		**19**	**11**	$T_{6,j}$

Nous proposons un exemple d'application de l'Algorithme II- 6. Les données relatives au produit j à réaliser sont fournies dans le Tableau II- 8. Le produit peut être réalisé à partir de la date r_j=0 et avant la date d_j=12. Les résultats de l'exécution sont donné dans le explicités sur la Figure II- 15 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées. Si l'Algorithme II- 6 s'achève au pas 5.1.1, il n'existe aucune solution réalisable. Cela signifie qu'il est impossible de fabriquer le produit avec les contraintes imposées au système de production.

Théorème II- 8

Si le Problème II- 6 a une solution réalisable, l'Algorithme II- 6 détermine les fenêtres et les dates d'exécution des opérations qui permettent de réaliser le produit avec un coût minimal, i.e. la solution fournie par l'Algorithme II- 6 est optimale pour le Problème II- 6.

Démonstration

La formulation du Problème II- 6 est équivalente aux équations de programmation dynamique (II-24) à (II-26). Il en découle que l'Algorithme II- 6 fournit la solution optimale au Problème II- 6.

CQFD

Complexité

Le tri au pas 1.1 de l'Algorithme II- 6 requiert $O(\sum_{1\le i\le m_j} y_{i,j}\log_2 y_{i,j}) \le O(y_j \log_2 y_j)$ [Beauquier et *al.*, 1992] où $y_j = \sum_{1\le i\le m_j} y_{i,j}$ est le nombre de fenêtres dans lesquelles chaque opération i peut être exécutée sur le produit j. Les pas 1.2 et 1.2.1 nécessitent $O(d_j - r_j + 1)$ opérations. Le pas 2 est exécuté pour chaque valeur de i, $2 \le i \le m_j + 1$ et pour chaque itération du pas 2, le pas 2.1 est exécuté $(d_j - r_j + 1)$ fois. A l'étape i, on exécute les pas 2.1.1 à 2.1.2 qui nécessitent $O(y_{i-1,j})$ opérations et le pas 2.1.3 qui n'est pas itéré plus de $(d_j - r_j + 1)$ fois. Les pas 2.1.3.1, 2.1.3.1.1 et 2.1.3.1.2 sont immédiats. Par conséquent, les pas 1 à 3 coûtent $O(y_j(\log_2 y_j + m_j(d_j - r_j + 1)) + m_j(d_j - r_j + 1)^2)$ opérations. Les pas 4 et 5.2.1 sont exécutés respectivement $O(d_j - r_j + 1)$ fois et m_j fois. Puisque tous les autres pas sont immédiats (i.e. s'exécutent en $O(1)$), **la complexité de l'Algorithme II- 6 est en $O(y_j \log_2 y_j + y_j \; m_j(d_j - r_j + 1) + m_j(d_j - r_j + 1)^2)$.**

Comme la complexité dépend de la durée $d_j - r_j$, l'algorithme est pseudo-polynomial.

Il est possible de réduire le temps de calcul nécessaire à l'Algorithme II- 6. Nous désignons par $\underline{\underline{T}}_{i,j}$ et $\overline{\overline{T}}_{i,j}$ la date de début de l'opération i effectuée sur le produit j calée respectivement au plus tôt et au plus tard dans l'ensemble des fenêtres possibles pour chaque opération, $i \in I_j$. Les dates calées au plus tôt sont données par l'Algorithme II- 2. De même, les dates calées au plus tard sont données par l'Algorithme II- 4. Evidemment, toute solution $\{T_{i,j}\}_{1 \le i \le m_j+1}$ réalisable pour le Problème II- 6 vérifie : $r_j \le \underline{\underline{T}}_{i,j} \le T_{i,j} \le \overline{\overline{T}}_{i,j} \le d_j$. Au lieu de considérer à chaque étape tous les états r_j à d_j, nous pouvons limiter le nombre d'états à chaque étape i aux états $T_{i,j}$ tels que $\underline{\underline{T}}_{i,j} \le T_{i,j} \le \overline{\overline{T}}_{i,j}$, pour $1 \le i \le m_j + 1$. (II-27)

Nous pouvons aussi nous limiter aux états pour lesquels il existe une fenêtre dans laquelle sont exécutées les opérations précédentes et suivantes.

Nous avons supposé, pour la présentation, que les durées et les dates étaient entières. Dans le cas où les durées et les dates sont rationnelles, nous multiplions toutes ces valeurs par le plus petit multiple commun de leur dénominateur que l'on note Q. Plus la valeur de Q est importante, plus les dates et les durées sont précises. L'Algorithme II- 6 peut être appliqué et sa complexité est alors égale à $O(y_j \log_2 y_j + y_j m_j Q(d_j - r_j + 1) + m_j Q^2 (d_j - r_j + 1)^2)$ et dépend directement de la précision des données.

Si le temps de calcul reste trop important, nous nous contentons d'une solution réalisable de coût proche de l'optimum. Nous pouvons obtenir une telle solution à partir de l'Algorithme II- 6. Pour cela, nous commençons par réduire la précision des données attendue (i.e. $1/Q$). Pour cette précision, nous calculons à l'aide de l'Algorithme II- 6 la solution optimale. L'Algorithme II- 6 est alors relancé avec une précision accrue et, plutôt que d'explorer tous les états de $S(T_{i+1,j})$ qui permettent d'atteindre l'état $T_{i+1,j}$, il se cantonne aux états proches de la solution optimale obtenue pour une précision moindre. Des raffinements successifs conduisent à une solution réalisable que l'on espère de valeur proche de l'optimum.

D'autres approches heuristiques basées sur la programmation dynamique sont possibles. Citons, par exemple, celle que nous avons mise en œuvre pour des problèmes de découpe [Antonio et *al.*, 1999]. Quelle que soit l'approche heuristique retenue, nous appliquons l'Algorithme II- 5 sur l'ensemble des fenêtres retenues par la méthode approchée. Dans cet ensemble de fenêtres, nous sommes alors certains d'obtenir une solution réalisable et, qui plus est, de coût minimal.

Place mémoire

Les méthodes de type programmation dynamique requièrent une place mémoire importante. Il devient essentiel de connaître l'espace-mémoire nécessaire pour garantir leur bonne exécution. Nous appelons "mot", l'espace-mémoire permettant de stocker une valeur rationnelle. Nous utilisons le mot comme unité de mesure de l'espace mémoire. Généralement, un *octet* suffit à coder un mot.

Dans l'Algorithme II- 6, les évaluations $C(T_{i+1,j})$ des états doivent être maintenues en mémoire. Pour calculer $C(T_{i+1,j})$ à l'étape i, il suffit de connaître les évaluations de l'étape précédente $C(T_{i,j})$. Nous conservons en mémoire, tout au long de l'algorithme, les évaluations des états $T_{i+1,j}$ de l'étape courante i et de $T_{i,j}$ de l'étape précédente. Puisque $r_j \leq T_{i,j} \leq d_j$, $2(d_j - r_j + 1)$ mots sont nécessaires pour garder en mémoire les coûts de l'Algorithme II- 6.

Outre les évaluations $C(T_{i+1,j})$, la seule structure de données qui nécessite une place importante est $Sopt(T_{i+1,j})$. Cette structure maintient en mémoire l'état $T_{i,j}$ de $S(T_{i+1,j})$ permettant de terminer l'opération i à la date $T_{i+1,j}$ et au coût le plus faible. Cette structure $Sopt(T_{i+1,j})$ autorise une reconstitution rapide de la solution optimale au pas 5.2.2.1. Elle nécessite une place mémoire de $m_j(d_j - r_j + 1)$ mots. Globalement, **l'Algorithme II- 6 requiert $(m_j + 2)(d_j - r_j + 1)$ mots.**

Evidemment, si nous restreignons l'ensemble des états à considérer à chaque étape (comme la relation (II-27) le permet), nous diminuons l'espace-mémoire nécessaire à l'algorithme.

Il est possible de réduire davantage l'espace-mémoire requis, mais ceci ne peut se faire qu'au détriment du temps de calcul. Pour cela, nous conservons en mémoire les seules évaluations $C(T_{i,j})$ et $C(T_{i+1,j})$ des états de deux étapes consécutives. A la fin de l'Algorithme II- 6, nous pouvons déduire de $C(T_{m_j+1,j})$ et de $C(T_{m_j,j})$, les dates $T_{m_j+1,j}$ et $T_{m_j,j}$ de la solution optimale. Ensuite, nous exécutons de nouveau l'Algorithme II- 6 pour déterminer successivement $T_{m_j-1,j}$, ..., $T_{1,j}$. Ainsi, nous reconstituons la solution en faisant l'économie de la place requise pour $Sopt(T_{i+1,j})$. Cette version de l'Algorithme II- 6 ne requiert plus que $2(d_j - r_j + 1)$ mots en mémoire, mais sa complexité est multipliée par m_j.

Remarques

Généralement, le coût d'utilisation des ressources $s_{i,j}$ représente les frais financiers d'utilisation des ressources. On peut également prendre en compte, dans ce coût, la charge de travail des ressources. En affectant un coût d'utilisation élevé sur ces machines goulots (i.e. celles qui ont la charge de travail la plus élevée), nous sommes assurés de réduire le plus

possible le temps d'utilisation de ces machines et espérons accroître la productivité du système de production.

Dans la pratique, les machines goulots sont souvent celles de coût d'utilisation le plus important. En effet, les responsables hésitent moins à investir dans l'achat d'une machine de faible coût que dans celle de coût élevé. Il en résulte que si la répartition des commandes par produit reste stable, alors les machines les plus chères sont aussi les plus utilisées.

Nous désignons par $\rho_{i,j}$, le taux d'utilisation des ressources nécessaires à l'opération i réalisée sur le produit j. Si nous voulons estimer plus précisément la charge future de travail des ressources, nous pouvons adapter l'Algorithme II- 6 de manière à minimiser le temps sur les ressources en fonction de leur charge estimée. Nous pouvons alors minimiser le critère suivant qui évalue la charge supplémentaire relative des machines : $\max_{1\le i\le m_j}\{\rho_{i,j}(T_{i+1}-T_i)\}$. Pour cela, il suffit de remplacer l'évaluation de chaque état $T_{i+1,j}$ définie selon les relations (II-26) par la relation de récurrence suivante :

$$C(T_{1,j})=0, \text{ pour } r_j \le T_{1,j} \le d_j.$$

$$C(T_{i+1,j}) = \begin{cases} +\infty & \text{si } S(T_{i+1,j}) = \varnothing \\ \min\limits_{T_{i,j}\in S(T_{i+1,j})} \left\{\max\left\{C(T_{i,j}); \rho_{i,j}(T_{i+1,j}-T_{i,j})\right\}\right\} & \text{sinon} \end{cases}$$

$$\text{pour } r_j \le T_{i+1,j} \le d_j,\ 1 \le i \le m_j.$$

En fait, l'Algorithme II- 6 peut être adapté à une large variété de fonctions objectifs non décroissantes.

Les algorithmes présentés dans ce chapitre déterminent les dates de début $T_{i,j}$ et de fin $T_{i+1,j}$ de chaque opération i, ainsi que la fenêtre $[a_{z_{i,j},i,j}\ ;\ b_{z_{i,j},i,j}]$ dans laquelle chacune doit être réalisée. Les périodes $[a_{z_{i,j},i,j}\ ;\ T_{i,j}]$ et $[T_{i+1,j}\ ;\ b_{z_{i,j},i,j}]$ des ressources correspondantes restent alors disponibles pour l'exécution d'opérations sur des produits commandés ultérieurement. Dans de nombreux cas, on peut dire qu'elle est la durée minimale possible des opérations exécutées à l'aide de ces ressources. Nous appelons $l_{z_{i,j}}$, la durée minimale pour ces ressources. Aussi, nous pouvons évaluer si les fenêtres restant disponibles $[a_{z_{i,j},i,j}\ ;\ T_{i,j}]$ et $[T_{i+1,j}\ ;\ b_{z_{i,j},i,j}]$ seront réellement utilisables, i.e. $l_{z_{i,j}} \le T_{i,j} - a_{z_{i,j},i,j}$ et $l_{z_{i,j}} \le b_{z_{i,j},i,j} - T_{i+1,j}$. Si elles ne peuvent plus être utilisables, elles sont qualifiées de chutes, sinon elles sont appelées tombants (vocabulaire emprunté au problème de découpe [Chauvet et *al.*, 1999b]). L'Algorithme II- 6 peut être adapté pour prendre en compte, dans l'évaluation des solutions, l'utilité future des périodes résiduelles. En gardant les fenêtres des ressources qui sont les plus susceptibles d'être utilisées, nous pouvons espérer augmenter encore la productivité du système.

II.5.5. Conclusion

Dans cette section, la méthode de gestion en temps réel décrite ordonnance chaque produit de manière à minimiser le coût d'utilisation des ressources, ceci sans dépasser la date de livraison fixée au plus tôt. Cette méthode de gestion permet de fabriquer chaque produit au coût le plus bas en tenant compte des produits déjà ordonnancés. En outre, cette méthode peut permettre de réduire l'utilisation des machines goulots et ainsi espérer augmenter la productivité du système.

Dans cette section, nous avons tout d'abord défini les dates d'exécution des opérations compatibles avec leurs durées opératoires admissibles et des fenêtres de temps données dans lesquelles chaque opération doit être réalisée. Pour résoudre ce problème formalisé par le Problème II- 7, nous avons proposé l'Algorithme II- 5 de complexité $O(m_j{}^2)$, où m_j est le nombre total d'opérations à réaliser sur le produit j. Ensuite, nous avons décrit l'Algorithme II-6 qui est basé sur la programmation dynamique. Il détermine en un temps pseudo-polynomial les fenêtres et les dates d'exécution correspondant aux opérations et induisant un coût d'utilisation des ressources minimal pour ce produit. Même s'il nécessite une discrétisation du temps, cet algorithme est tout à fait utilisable sur des problèmes réels. Du point de vue académique, nous nous demandons si le Problème II- 6 est NP-difficile ou s'il est possible de fournir une solution en un temps polynomial. Cette question reste ouverte.

II.6. RESULTATS COMPARATIFS

II.6.1.Introduction

Dans cette section, nous voulons comparer les différents modes de gestion en temps réel présentés dans les sections précédentes. Pour bien comprendre les effets de ces modes de gestion, nous limitons le champ d'étude aux cas où une seule ressource est nécessaire à l'exécution de chaque opération sur chaque produit, et où les ressources sont disponibles en un seul exemplaire.

Les hypothèses restent donc vérifiées, et nous nous plaçons dans une situation où l'Hypothèse II- 7 est plus restreinte. Autrement dit, nous considérons que les produits passent successivement sur un ensemble de m machines dans un ordre propre à chaque produit. Il s'agit du problème d'atelier de type *job shop* $J_m|\text{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$, où $u_{i,j} \in \Re^+ \cup \{+\infty\}$.

Chacune des méthodes de gestion donne l'ordonnancement de chaque produit au moment où sa commande est enregistrée.

La première méthode de gestion consiste **à fabriquer le produit au plus tôt** (voir section 3). De plus, cette méthode permet de fournir au client un délai minimum de livraison réalisable et précis. Comme nous l'avons vu (voir Section II.4.5), cette première méthode permet d'atteindre la productivité maximale à long terme quand tous les produits sont identiques (i.e. elle minimise le *makespan* de l'ensemble des produits).

Les deux autres méthodes proposées utilisent respectant la date de livraison fixée au plus tôt fournie par cet algorithme.

La deuxième méthode de gestion revient **à fabriquer le produit au plus tard**, tout en respectant la date de livraison fixée au plus tôt (voir section 4). Ce mode de gestion permet de minimiser le temps pendant lequel des ressources sont utilisées pour fabriquer le produit.

La troisième méthode de gestion a pour but de **minimiser le coût d'utilisation des ressources**, tout en livrant le produit au plus tôt (voir section 5). Nous l'appliquons ici pour diminuer l'utilisation des ressources goulots et ainsi accroître la productivité du système. Pour cela, nous affectons à chaque ressource un coût d'utilisation égale à l'estimation du taux d'utilisation future de cette ressource.

Ces trois méthodes de gestion garantissent l'optimalité du critère considéré pour le produit à ordonnancer. L'évaluation de ces critères est faite produit par produit puisqu'il est impossible d'attendre que toutes les commandes soient connues pour fixer la gestion du système. Ainsi, nous qualifions ces objectifs de critères locaux.

Notre objectif global est d'obtenir une gestion qui maximise la productivité du système de production à long terme. Nous décrivons différents indicateurs dont la productivité. Ils sont utiles pour comprendre les effets de chaque mode de gestion.

Ensuite, nous analysons les résultats fournis par ces algorithmes dans le pire des cas. Enfin, nous proposons d'appliquer ces trois méthodes de gestion sur une longue période de production. Nous comparons alors la productivité moyenne du système suivant les méthodes de gestion utilisées.

II.6.2.Définition des indicateurs

II.6.2.1.Indicateurs généraux

Le ***makespan* de l'ensemble des produits** est la longueur de la plage de temps qui s'écoule entre la date de début de fabrication du premier produit et la date de fin de fabrication du dernier produit de la séquence de produits.

La **productivité** donne le nombre de produits réalisés par unité de temps. Pour la calculer, nous divisons le nombre de produits fabriqués par le *makespan* de l'ensemble de ces produits. Il est évidemment relatif à l'unité de temps choisie. L'objectif est d'obtenir une gestion qui maximise la productivité du système de production.

II.6.2.2.Indicateurs relatifs aux ressources

L'**utilisation** d'une ressource est la somme des durées des opérations effectuées par cette ressource sur l'ensemble des produits considérés.

Le **taux d'utilisation** d'une ressource est le rapport entre son utilisation et le *makespan*. Pour un même niveau de productivité, nous préférons un taux d'utilisation le plus bas possible qui laisse plus de temps disponible pour des activités extérieures à la production, comme la maintenance.

Nous proposons un indicateur peu sensible aux durées opératoires : le **rendement** d'une ressource. C'est, pour une ressource, le quotient de la charge minimale de travail de la ressource (i.e. la somme des durées minimales de toutes les opérations réalisées par la ressource) par le *makespan* de tous les produits.

Le rendement d'une ressource est compris entre 0 et 1. Bien évidemment, nous recherchons un mode de gestion pour lequel le rendement des ressources est maximal. Les contraintes fortes imposées au système de production, telle l'absence d'attente entre les opérations, ne permettent pas dans la pratique d'atteindre un rendement de 1, quel que soit le mode de gestion choisi. On

peut l'atteindre pour des cas très spécifiques comme lorsque que tous les produits sont identiques et que toutes les durées opératoires sont égales.

II.6.2.3.Indicateurs relatifs aux produits

Le **temps de cycle** d'un produit est la durée qui s'écoule entre le début et la fin de sa fabrication. Plus il est réduit, moins le temps de production et donc l'immobilisation des ressources sont importants. Nous cherchons donc à diminuer le temps de cycle pour réduire les coûts de production.

Une partie de ce temps est incompressible ; il s'agit de la charge de travail induite par le produit (i.e. la somme des durées minimales des opérations à réaliser sur le produit).

Nous désignons l'autre partie de son temps de cycle (sur laquelle le mode de gestion peut influer) par l'**allongement des durées opératoires** du produit.

Le **délai de livraison** d'un produit (appelé *flow time* dans la littérature anglaise) est la durée qui s'écoule entre l'instant où la commande du produit est connue et la date de fin de production. Ce délai prend en compte l'attente du produit (*waiting time* en anglais) avant son début de fabrication et, par conséquent, est plus long que le temps de cycle. Nous souhaitons que le délai de livraison des produits soit le plus court possible, gage de réactivité du système de production.

Le **temps de calcul** de l'ordonnancement d'un produit est la durée nécessaire à l'ordinateur gérant le système pour définir l'ordonnancement du produit. Nous souhaitons qu'il soit aussi faible que possible puisque nous gérons le système en temps réel. Ce temps est lié à l'ordinateur, au langage de programmation et au compilateur utilisés.

II.6.3.Analyse dans le pire des cas

Des études ont été menées pour évaluer la qualité des résultats fournis par les algorithmes en temps réel [Sgall, 1998]. Pour cela, on compare leurs solutions à celle que l'on obtiendrait si au moment d'ordonnancer on savait quels sont les produits à venir. On parle alors d'ordonnancement obtenu en temps différé (*off-line scheduling*, en anglais). Le rapport entre la solution fournie par un algorithme en temps réel et la solution optimale que l'on obtiendrait en temps différé, permet de mesurer la qualité de cet algorithme.

Nous considérons une heuristique H utilisée en temps réel pour déterminer l'ordonnancement de produits successifs. Nous appelons $Opt(J)$, la valeur minimale du critère pour l'ordonnancement d'un ensemble J de produits obtenu sous l'hypothèse que tous les produits sont connus à l'avance. Nous désignons par $H(J)$, la valeur du critère de l'ordonnancement des produits obtenu en appliquant l'heuristique H. Nous évaluons la qualité de la solution de l'algorithme H dans le pire des cas par $R_H = \max_{J\in\Im}\left\{\frac{H(J)}{Opt(J)}\right\}$ où $\Im$ désigne l'ensemble de toutes les configurations possibles de produits. On dit que l'algorithme H est R_H-compétitif.

Soient R_{H_1} et R_{H_2}, les évaluations des solutions de deux algorithmes en temps réel H_1 et H_2 par rapport à l'ordonnancement optimal *off-line*. Si $R_{H_1} > R_{H_2}$ alors nous concluons que l'heuristique H_1 peut donner de plus mauvais résultat que l'heuristique H_2. La seule valeur de R_{H_1} ne nous permet pas de juger de la qualité de l'heuristique H_1.

Comme nous recherchons une productivité maximale du système à long terme, le critère qui nous intéresse ici est le *makespan* de l'ensemble des produits. La plupart des études qui existent concernent le cas de produits sur lesquels une seule opération est réalisée [Sgall, 1998]. Les seuls résultats connus pour le cas où les produits doivent subir plusieurs opérations concernent les ateliers de type *flow shop* et *job shop* à deux machines. En effet, pour $F_2 \| C_{max}$ et $J_2 \| C_{max}$, Chen et Woeginger (1995) ont montré qu'il n'existe pas d'algorithme déterministe en temps réel qui garantisse d'obtenir un *makespan* pour l'ensemble des produits inférieur au double du *makespan* optimal obtenu en connaissant tous les produits à venir.

Dans la suite, nous proposons d'étudier la compétitivité des algorithmes proposés, dans des cas non limités à $m=2$ machines pour des ateliers de type $J_m|nwt, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$, où $u_{i,j} \in \Re^+ \cup \{+\infty\}$.

Théorème II- 9

Les Algorithmes II- 2, II- 4 et II- 6 sont m-compétitifs pour les problèmes d'atelier de type : $F_m|nwt, r_j, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$, $J_m|nwt, r_j, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{max}$, $F_m|r_j, p_{i,j}|C_{max}$ et $J_m|r_j, p_{i,j}|C_{max}$, pour $m \ge 2$.

Démonstration

a) Pour un problème du type $F_m|nwt, l_{i,j} \leq P_{i,j} \leq u_{i,j}|C_{max}$, nous montrons qu'il existe des cas pour lesquels les algorithmes cités fournissent des solutions dont le *makespan* est m-fois supérieur au *makespan* "optimal".

Nous considérons que l'ensemble J_0 des n produits doit passer sur m machines dans le même ordre (comme dans un *flow shop*) et que chacun des produits soit disponible à la date $r_j=0$.

Pour ce type d'atelier, supposons que les algorithmes cités soient (m-δ)-compétitifs, où $\delta>0$ de valeur très proche de zéro.

Considérons le cas où $n=(4m-4)$ produits doivent être réalisés. Un produit sur deux doit subir des opérations dont la durée égale à $l_{i,j}=\delta/5m$ (i.e. dans le cas où j est impair). Les autres produits (i.e. dans le cas où j est pair) doivent subir des opérations de durées égales soit à 1, soit à $\delta/5m$, comme il est indiqué sur la Figure II- 16 pour $m=3$ machines.

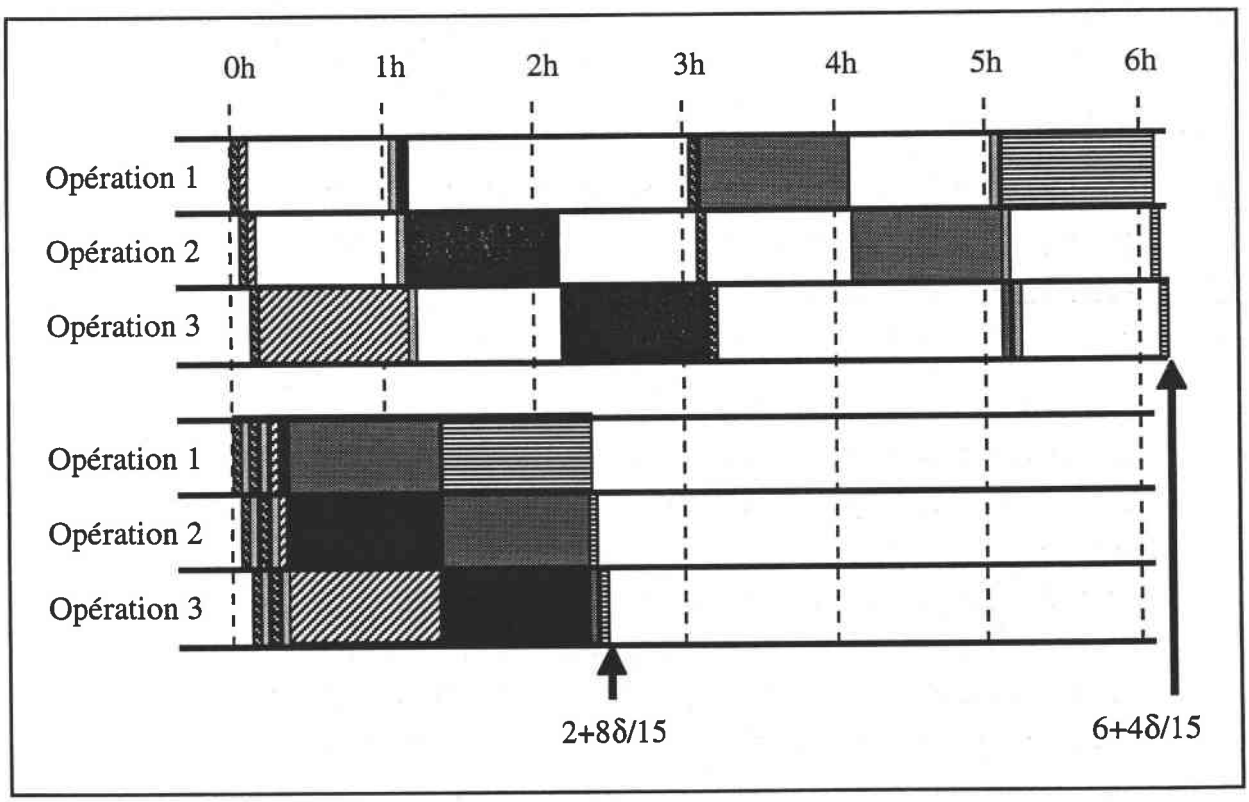

Figure II- 16 : Cas prouvant que les algorithmes ne sont pas (m-δ)-compétitifs

Le premier ordonnancement est celui fourni par les 3 algorithmes qui placent les produits dans l'ordre d'arrivée. En appliquant ces heuristiques, nous obtenons une valeur du *makespan* égale à $H(J_0)=(m-1)m-(m-1)(m-5)(\delta/5m)$. Le deuxième ordonnancement est optimal pour le cas où tous les produits sont connus au départ. Son *makespan* vaut $Opt(J_0)=(m-1)+4(m-1)(\delta/5m)$.

Ce cas nous donne une borne inférieure de la compétitivité des algorithmes :

$$R_H \geq \frac{H(J_0)}{Opt(J_0)} = \frac{m-(m-5)(\delta/5m)}{1+4(\delta/5m)} = \frac{[m+4(\delta/5)]-\delta+\delta/m}{[m+4(\delta/5)]/m} > m-\delta, \text{ pour } m \geq 2.$$

Nous aboutissons à une contradiction avec l'hypothèse faite que les algorithmes proposés sont (m -δ)-compétitifs, où $\delta>0$. Par conséquent, il existe des cas pour lesquels les algorithmes cités fournissent des solutions dont le *makespan* est m -fois supérieur au *makespan* "optimal".

Si des durées opératoires maximales peuvent être différentes des valeurs minimales (i.e. $l_{i,j}<u_{i,j}$), alors l'Algorithme II- 2 donne une solution où certaines durées sont allongées. Mais, ceci ne modifie pas le *makespan* final obtenu.

Nous pouvons étendre ce résultat aux situations où les dates de disponibilité r_j et les gammes des produits ne sont pas nécessairement identiques. Par conséquent, nous en concluons qu'il existe des cas pour lesquels l'Algorithme II- 2, l'Algorithme II- 4 et l'Algorithme II- 6 fournissent des solutions dont le *makespan* est m -fois supérieur au *makespan* "optimal" pour les problèmes de type $F_m|\text{nwt}, r_j, l_{i,j} \leq P_{i,j} \leq u_{i,j}|C_{\max}$ et $J_m|\text{nwt}, r_j, l_{i,j} \leq P_{i,j} \leq u_{i,j}|C_{\max}$, pour $m \geq 2$. De la même manière, ces algorithmes fournissent des solutions dont le *makespan* est m -fois supérieur au *makespan* "optimal" pour les problèmes où l'attente entre les opérations est permise, i.e. de type $F_m|r_j, p_{i,j}|C_{\max}$ et $J_m|r_j, p_{i,j}|C_{\max}$, pour $m \geq 2$ (dans ce cas $p_{i,j}=l_{i,j}$).

b) Nous montrons ensuite que les algorithmes cités sont m -compétitifs pour les problèmes d'atelier de type : $J_m|\text{nwt}, r_j, l_{i,j} \leq P_{i,j} \leq u_{i,j}|C_{\max}$.

Supposons que l'ensemble J des n produits doit passer sur m machines dans un ordre quelconque (comme dans pour un *job shop*) et que chaque produit est disponible à la date r_j. Les produits sont ordonnancés en temps réel dans l'ordre chronologique des dates de r_j. Par conséquent, $r_j \leq r_p$, pour $1 \leq j \leq p \leq n$.

Nous appelons J_p l'ensemble des p premiers produits, pour $1 \leq p \leq n$. Nous désignons par $Opt(J_p)$, le *makespan* optimal des produits de J_p. Le *makespan* optimal de l'ensemble des produits vérifie $Opt(J)=Opt(J_n) \geq Opt(J_p)$, pour $1 \leq p \leq n$. D'autre part, puisque les produits ne peuvent subir qu'une opération à la fois, nous avons : $Opt(J_p) \geq \max_{j=1,\ldots,p} \left\{ r_j + \sum_{i=1,\ldots,m} l_{i,j} \right\}$, où $l_{i,j}$ est la durée minimale de l'opération effectuée par la machine i sur le produit j. De plus, puisque les machines ne peuvent exécuter qu'une opération à la fois, nous avons : $Opt(J_p) \geq \max_{i=1,\ldots,m} \left\{ \max_{j=1,\ldots,p} \left\{ r_j + \sum_{k=j,\ldots,p} l_{i,k} \right\} \right\}$.

Considérons l'algorithme qui, pour chaque produit j, réserve le temps total de fabrication $\sum_{i=1,\ldots,m} l_{i,j}$ sur chaque machine i, au-delà de sa date de disponibilité r_j. Nous désignons cette

heuristique par l'Algorithme II- 7. La date $C_{m,j}$ à laquelle le produit j est terminé est obtenue par la relation de récurrence suivante :

$$\begin{cases} C_{m,0} = r_1 \\ C_{m,j} = \max\left\{ C_{m,j-1} + \sum_{i=1,\ldots,m} l_{i,j} \; ; \; r_j + \sum_{i=1,\ldots,m} l_{i,j} \right\} \quad \forall 1 \le j \le n \end{cases}$$

Par conséquent, $H_{\text{II-7}}(J_p)$, le *makespan* obtenu en appliquant l'Algorithme II- 7 sur les p premiers produits vaut $H_{\text{II-7}}(J_p)=C_{m,p}$, pour $1 \le p \le n$.

Montrons, par récurrence, que $H_{\text{II-7}}(J_p) \le m \max_{i=1,\ldots,m}\left\{ \max_{j=1,\ldots,p}\left\{ r_j + \sum_{k=j,\ldots,p} l_{i,k} \right\}\right\}$.

Cette relation est vérifiée pour $p=1$. Supposons la vérifiée au rang p. Nous avons alors :

$$H_{\text{II-7}}(J_{p+1})=C_{m,p+1}=\max\left\{ C_{m,p} + \sum_{i=1,\ldots,m} l_{i,p+1} \; ; \; r_{p+1} + \sum_{i=1,\ldots,m} l_{i,p+1} \right\}$$

$$H_{\text{II-7}}(J_{p+1}) \le \max\left\{ m \max_{i=1,\ldots,m}\left\{ \max_{j=1,\ldots,p}\left\{ r_j + \sum_{k=j,\ldots,m} l_{i,k} \right\}\right\} + \sum_{i=1,\ldots,m} l_{i,p+1} \; ; \; r_{p+1} + \sum_{i=1,\ldots,m} l_{i,p+1} \right\}$$

$$H_{\text{II-7}}(J_{p+1}) \le \max\left\{ m \max_{i=1,\ldots,m}\left\{ \max_{j=1,\ldots,p}\left\{ r_j + \sum_{k=j,\ldots,m} l_{i,k} \right\}\right\} + m \max_{i=1,\ldots,m}\left\{ l_{i,p+1} \right\} \; ; \; r_{p+1} + \sum_{i=1,\ldots,m} l_{i,p+1} \right\}$$

$$H_{\text{II-7}}(J_{p+1}) \le m \max_{i=1,\ldots,m}\left\{ \max_{j=1,\ldots,p+1}\left\{ r_j + \sum_{k=j,\ldots,m} l_{i,k} \right\}\right\}.$$

Nous obtenons $H_{\text{II-7}}(J_p) \le m \max_{i=1,\ldots,m}\left\{ \max_{j=1,\ldots,p}\left\{ r_j + \sum_{k=j,\ldots,p} l_{i,k} \right\}\right\}$, pour $1 \le p \le m$. Nous en déduisons que $H_{\text{II-7}}(J_n) \le m \max_{i=1,\ldots,m}\left\{ \max_{j=1,\ldots,n}\left\{ r_j + \sum_{k=j,\ldots,n} l_{i,k} \right\}\right\} \le m \; Opt(J_n) = m \; Opt(J)$.

Nous concluons que le *makespan* de l'Algorithme II- 7 n'est pas plus de m fois supérieur au *makespan* optimal obtenu en connaissant tous les produits à venir. Les trois méthodes de gestion que nous proposons placent chaque produit au plus tôt et, donc, utilisent pour chaque produit une plage de temps inférieur à $\sum_{i=1,\ldots,m} l_{i,j}$. Par conséquent, ces 3 algorithmes en temps réel fournissent des solutions dont le *makespan* pour l'ensemble des produits est inférieur au *makespan* obtenu en appliquant l'Algorithme II- 7.

Finalement, l'Algorithme II- 2, l'Algorithme II- 4 et l'Algorithme II- 6 sont m -compétitifs, pour $J_m|\text{nwt}, r_j, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$. Nous pouvons étendre le résultat aux cas particuliers suivants : $F_m|\text{nwt}, r_j, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$, $F_m|\text{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$ ou encore $J_m|\text{nwt}, l_{i,j} \le P_{i,j} \le u_{i,j}|C_{\max}$. Ces résultats restent également vrais pour les problèmes où l'attente est permise, par exemple $J_m|r_j, p_{i,j}|C_{\max}$ (dans ce cas $p_{i,j}=l_{i,j}$).

CQFD

Ce théorème montre que les trois méthodes proposées ont un comportement similaire sur les situations extrêmes. Les résultats obtenus permettent également d'évaluer combien les contraintes de temps réel dégradent les solutions par rapport à un contexte où les calculs se font en temps différé. Les cas particuliers mettent en exergue les faiblesses des algorithmes. Nous voyons que les algorithmes proposés sont moins compétitifs quand les durées des opérations réalisées à l'aide d'une même machine diffèrent d'un produit à l'autre. Dans la plupart des applications réelles, cette hypothèse n'est pas valide. Cependant, nous espérons développer d'autres méthodes en temps réel qui se comportent mieux sur ces cas spécifiques.

Récemment, des algorithmes dits aléatoires (*randomized algorithm*, en anglais) ont été introduits pour des situations où une seule opération doit être exécutée sur chaque produit [Ben-David et *al.*, 1994]. La définition de compétitivité a été élargie à ces algorithmes. Ils donnent généralement des résultats au moins aussi compétitifs que les algorithmes classiques, dits alors déterministes.

Dans les problèmes qui nous intéressent dans cette section, plusieurs opérations doivent être réalisées sur chaque produit et une seule ressource est disponible pour chaque opération. Nous pouvons proposer une analogie du principe des algorithmes aléatoires applicable aux cas qui nous intéressent. Le principe serait d'ordonnancer chaque produit de manière à atteindre un rendement (voir définition dans la section précédente) constant en moyenne. Rappelons que le rendement d'une ressource est le quotient entre la charge minimale de travail de la ressource et le *makespan* de tous les produits obtenu pour une heuristique H donnée. Puisque la valeur de la charge minimale de chaque ressource est inférieure au *makespan* obtenu en connaissant tous les produits à venir, l'inverse du rendement d'une ressource est toujours supérieur au ratio R_H, ratio qui mesure la qualité de la méthode de gestion H dans le pire des cas.

Nous espérons pouvoir développer de tels algorithmes aléatoires à partir de l'Algorithme II- 6 pour des situations qui ne seraient plus limitées à une opération exécutée sur un produit.

II.6.4. Analyse en moyenne

L'objectif de cette section est d'évaluer la qualité des trois modes de gestion proposés en simulant leur application sur différents ateliers. Nous tentons de relier la productivité et le rendement aux caractéristiques des systèmes de production.

Les résultats que nous fournissons sont obtenus à partir de programmes développés sur une station de travail Sun Sparc 20 en langage C. La puissance de calcul de ce type de machine est comparable à celles que nous rencontrons aujourd'hui même dans les petites et moyennes entreprises.

II.6.4.1. Cas d'atelier de type flow shop

Nous considérons un atelier de type *flow shop* à m machines. Nous y fabriquons 1000 produits différents générés au hasard. Chaque produit subit m opérations successives : la première sur la machine 1, la deuxième sur la machine 2, ..., la dernière sur la machine m. La durée minimale des opérations exécutée sur chaque machine est choisie aléatoirement dans l'intervalle [0 ; 50] et la durée opératoire maximale est choisie au hasard et est inférieure à 150.

Globalement, plus le nombre de machines est important, plus le nombre d'opérations est important. L'inégalité des durées opératoires impose des attentes entre les produits, ce qui diminue la performance du système. Le nombre de machines augmentant, les performances (i.e. productivité et rendement) diminue. La Figure II- 17 montre l'évolution de la productivité (courbes épaisses) et du rendement moyen (courbes fines) si la fabrication est lancée au plus tôt (courbe ◊), au plus tard (courbe o), à coût minimal (courbe *). Les courbes des deux derniers modes de gestion sont superposées. D'une manière générale, dans ce contexte de *flow shop*, fabriquer au plus tôt (courbe ◊) est meilleur en termes de productivité et de rendement que fabriquer au plus tard ou à coût minimal. Ceci se comprend si on se souvient que fabriquer au plus tôt est la meilleure politique quand tous les produits sont identiques (voir Section II.3.5). Le rendement moyen est nettement supérieur au rendement dans le pire des cas de $1/m$ calculé dans la section précédente.

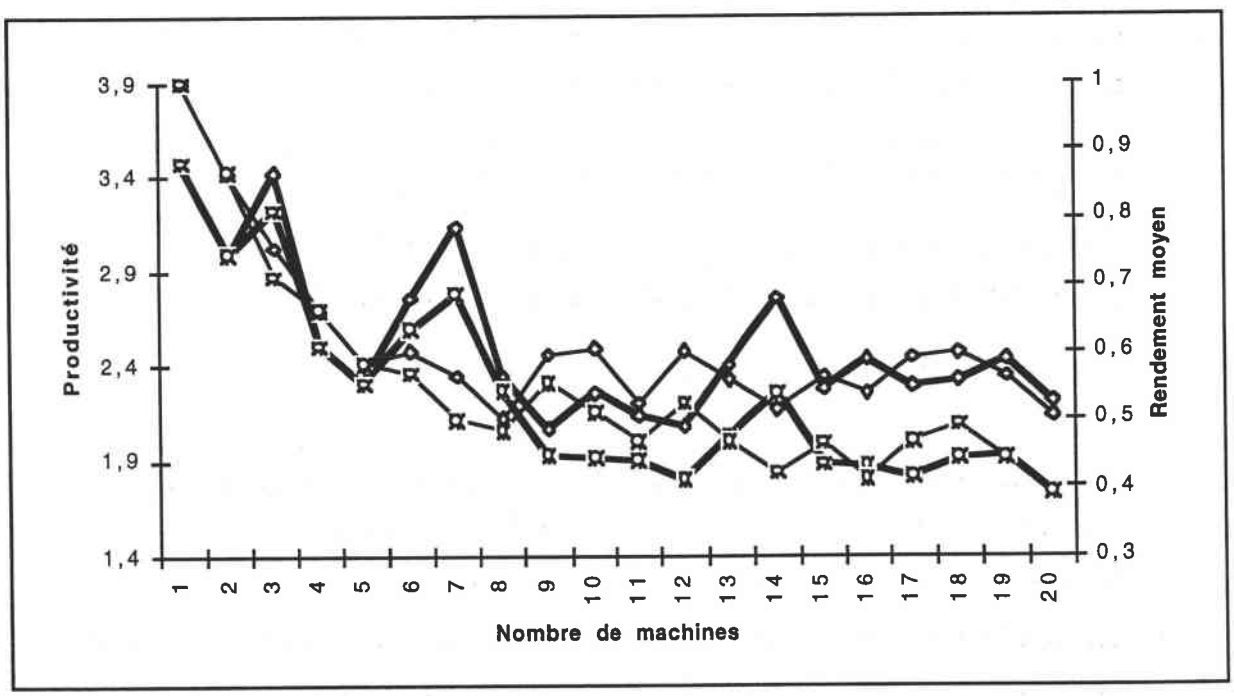

Figure II- 17 : Productivité et rendement des différents modes de gestion

Le nombre de machines augmentant, le nombre d'opérations, le temps de cycle et le délai de livraison s'accroissent. La Figure II- 18 montre l'évolution du temps de cycle moyen (courbes

épaisses) et du délai moyen de livraison (courbes fines) si la fabrication est lancée au plus tôt (courbe ◊), au plus tard (courbe o), à coût minimal (courbe *). Les courbes des deux derniers modes de gestion sont également superposées. D'une manière générale, fabriquer au plus tard ou à coût minimal permet de réduire le temps de cycle. Cependant, dans ce contexte de *flow shop*, fabriquer au plus tôt garantit une meilleure productivité. Ainsi, le délai moyen de livraison est plus court si on fabrique au plus tôt que si on fabrique au plus tard ou à coût minimal.

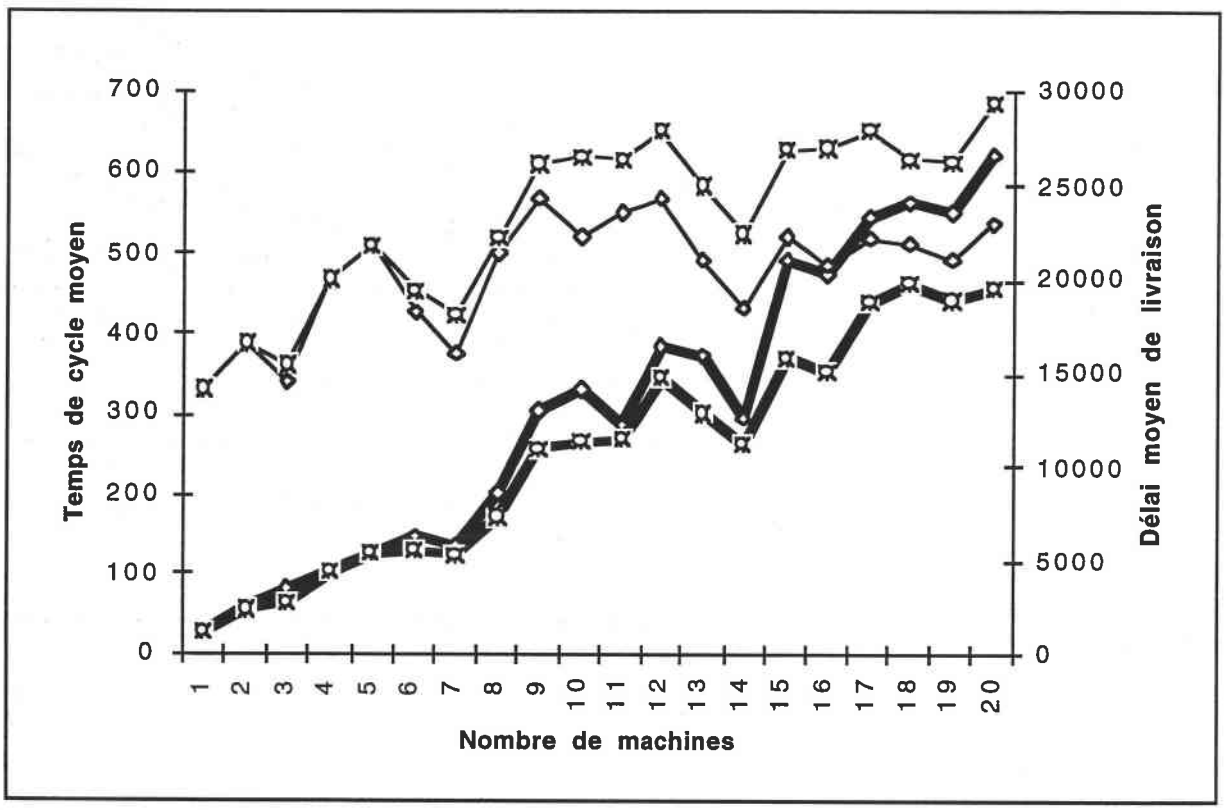

Figure II- 18 : Temps de cycle et délai de livraison des différents modes de gestion

La complexité des algorithmes donnée dans les sections précédentes permet d'évaluer le temps de calcul nécessaire à l'ordonnancement de chaque produit. Ces résultats théoriques sont confirmés par les résultats numériques représentés sur la Figure II- 19. Elle montre l'évolution du temps de calcul moyen (courbes épaisses) et de l'écart moyen à cette moyenne (courbes en pointillé) si la fabrication est lancée au plus tôt (courbe ◊), au plus tard (courbe o), à coût minimal (courbe *). Pour les trois modes de gestion, le temps moyen de calcul est multiplié par 10 quand le nombre de machines est multiplié par 20. Ordonnancer de manière à fabriquer à moindre coût est plus coûteux en temps de calcul (4 à 5 fois plus qu'une gestion au plus tôt). La gestion au plus tard nécessite 2 à 3 fois plus de temps de calcul que la gestion au plus tôt.

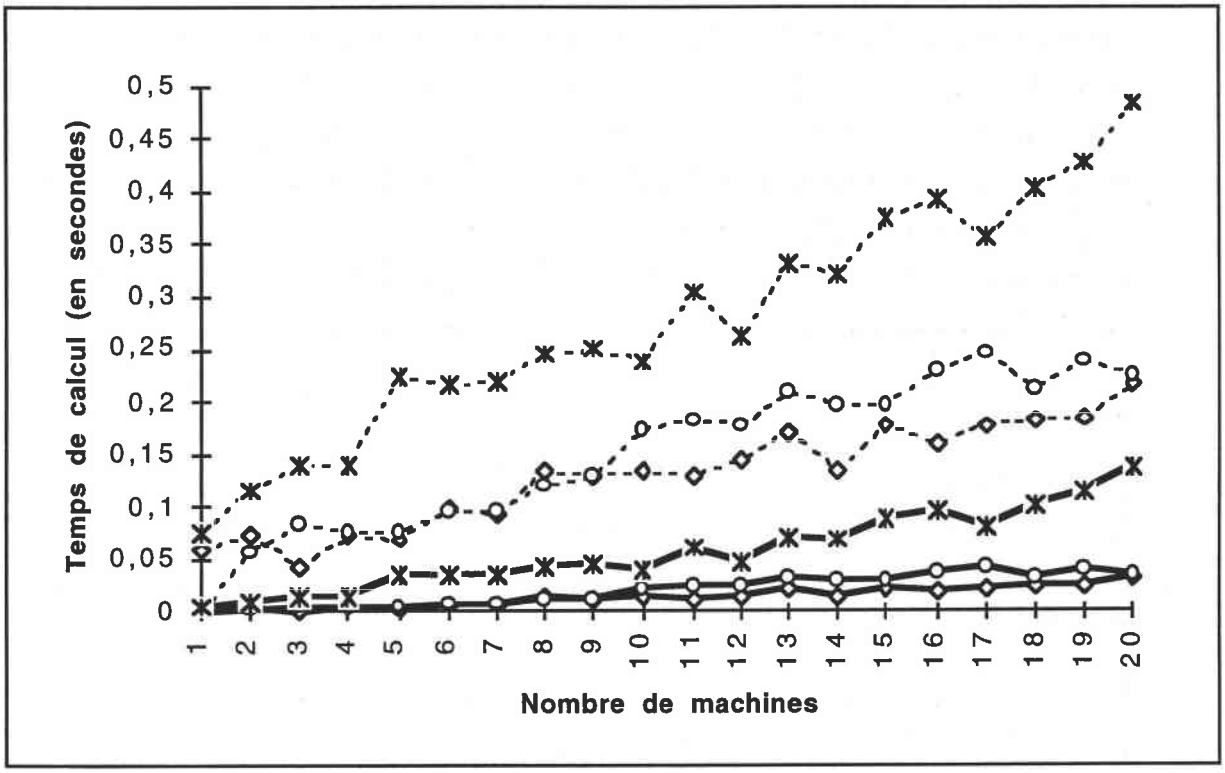

Figure II- 19 : Temps de calcul des différents modes de gestion

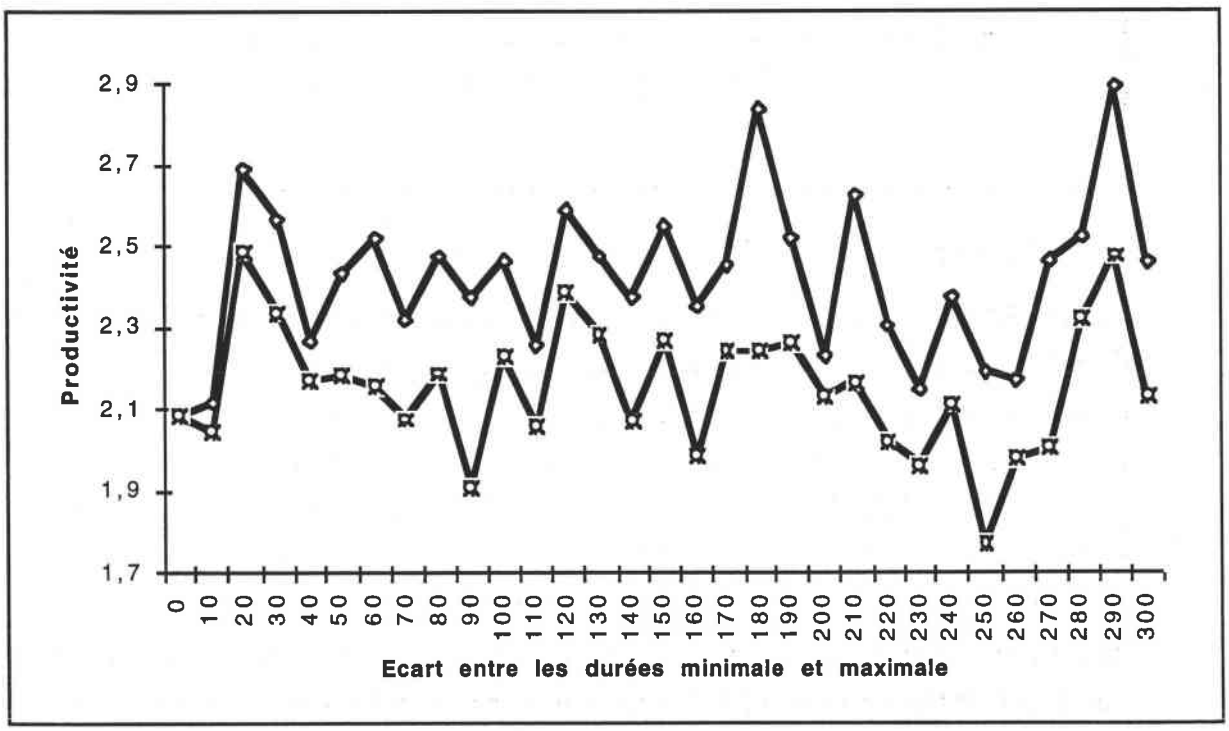

Figure II- 20 : Productivité fonction de l'écart entre durées maximales et minimale

Dans ce contexte de *flow shop* pour le cas à $m=10$ machines, la Figure II- 20 donne la productivité du système pour différents écarts entre durées minimale et maximale. Quand cet écart est nul, nous sommes dans un problème à durées opératoires fixes (P.T.O.F.) (voir Chapitre I). Quand l'écart est limité, nous sommes dans un problème à durées opératoires limitées (P.T.O.L.). Pour des écarts importants, nous pouvons considérer être dans un problème à durées opératoires illimitées (P.T.O.I.). Pour un écart nul, les durée opératoires sont imposées et les trois politiques de gestion en temps réel fournissent les mêmes résultats. Quand la durée maximale des opérations n'est plus égale à la durée minimale, le choix d'une politique n'est plus indifférent. Les résultats sont alors stables et peu dépendants de la différence entre les durées opératoires.

II.6.4.2.Du cas d'atelier de type flow shop au job shop

Nous considérons un atelier constitué de 10 machines, numérotées de 1 à 10. A l'aide de ces machines, nous réalisons 10 opérations sur 1000 produits. La durée minimale des opérations exécutée sur chaque machine est choisie aléatoirement dans l'intervalle [0 ; 50]. La durée opératoire maximale est déterminée aléatoirement et est inférieure à 150.

Chaque produit passe successivement sur les machines dans un ordre qui lui est propre. Mais cet ordre peut se déduire d'une séquence commune à tous les produits. La séquence globale minimale (ou plus courte séquence commune) [Souilah, 1990] est la plus petite séquence de machines telle que tout routage de produit puisse en être déduit par suppression d'un ou plusieurs éléments de la séquence.

Nous nous donnons une longueur de séquence Θ. Tous les produits sont alors générés suivant cette séquence. Par exemple, pour $\Theta=30$, la séquence de machines est alors 1, 2, ..., 9, 10, 1, 2, ..., 9, 10, 1, 2, ..., 9, 10. Un produit peut alors subir jusqu'à trois opérations sur chaque machine.

Pour une valeur de $\Theta=10$, les produits subissent exactement une opération sur chaque machine et dans le même ordre : nous sommes dans le cas d'un *flow shop*. Pour une valeur de $\Theta=100$, un produit peut subir des opérations sur les machines dans n'importe quel ordre : nous sommes dans le cas d'un *job shop*.

Nous constatons sur la Figure II- 21 que dans un contexte de *flow shop*, la productivité est la meilleure si nous produisons au plus tôt (courbe ◊). Mais dès que la séquence n'est plus égale à 10, les autres politiques (produire au plus tard, courbe o et produire à moindre coût *) sont plus avantageuses. Ces deux dernières politiques sont moins efficaces pour les *flow shops* mais permettent d'atteindre une meilleure productivité pour les autres situations (i.e. $\Theta>10$).

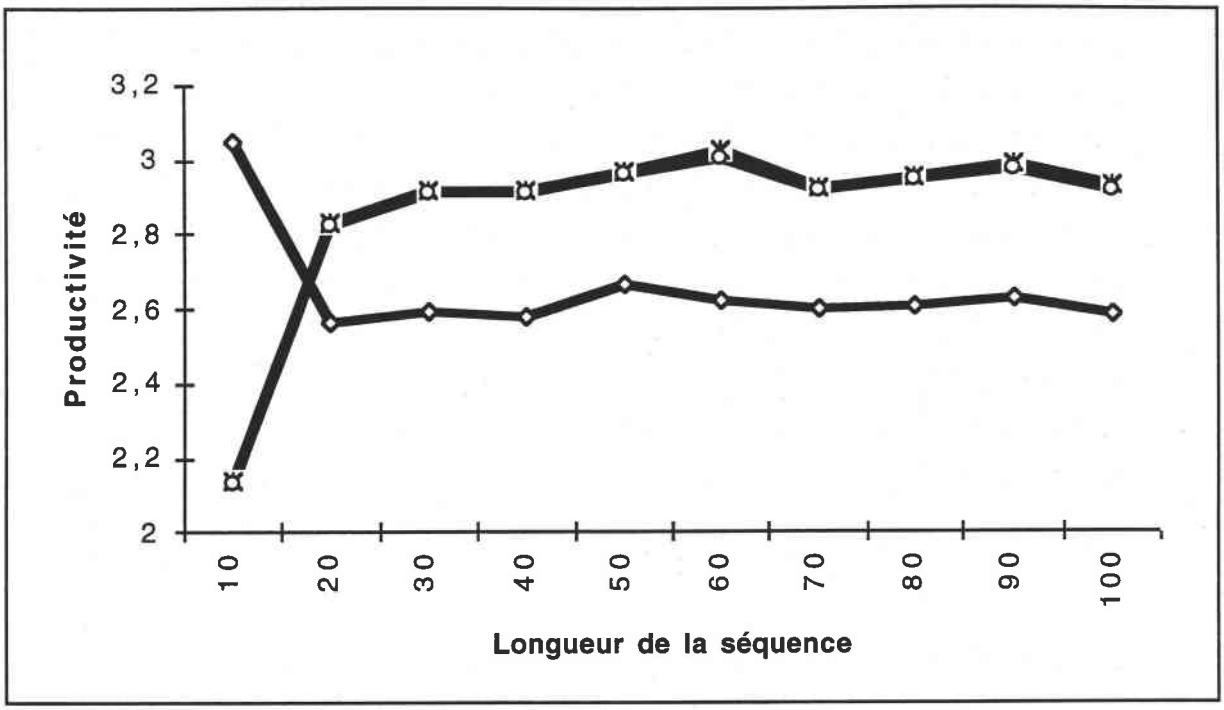

Figure II- 21 : Productivité des différents modes de gestion

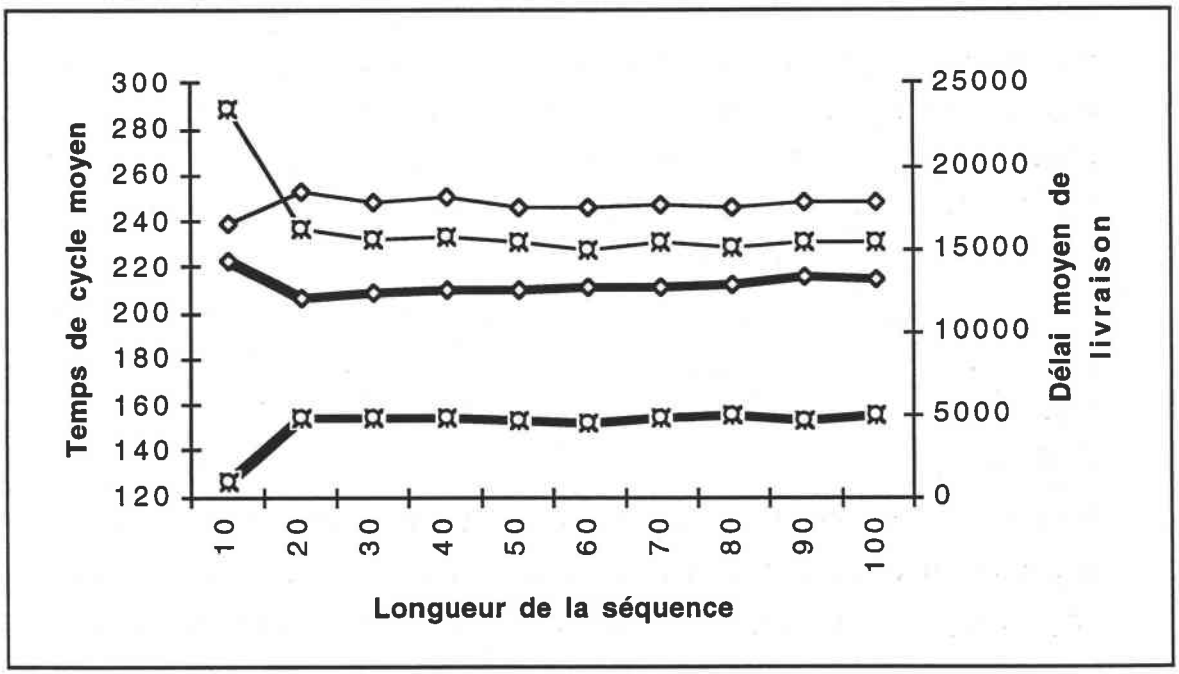

Figure II- 22 : Temps de cycle et délai de livraison des différents modes de gestion

Produire au plus tôt (courbe ◊) augmente le temps de cycle moyen (courbe épaisse sur la Figure II- 22). Les deux autres politiques permettent de réduire ce temps de cycle. La productivité accrue en produisant au plus tôt dans un contexte de *flow shop* permet de réduire les délais de livraison. Dès que la séquence n'est plus égale à 10, les autres politiques (produire au plus tard,

courbe o et produire à moindre coût *) sont là encore plus avantageuses, puisqu'elles garantissent des délais de livraison plus courts.

II.6.4.3.Influence des durées opératoires

Nous considérons un atelier de type constitué d'une ligne de 12 machines, numérotées de 1 à 12. Sur cette ligne, nous fabriquons 1000 produits. Chaque produit subit entre 8 et 12 opérations successives sur ces machines. L'ordre dans lequel le produit subit les opérations est l'ordre des machines. La durée minimale des opérations exécutée sur chaque machine est choisie aléatoirement dans l'intervalle [α ; 50]. Pour des différentes valeurs de α allant de 0 à 50, nous observons l'évolution des indicateurs. La durée opératoire maximale est déterminée aléatoirement et est inférieure à 150.

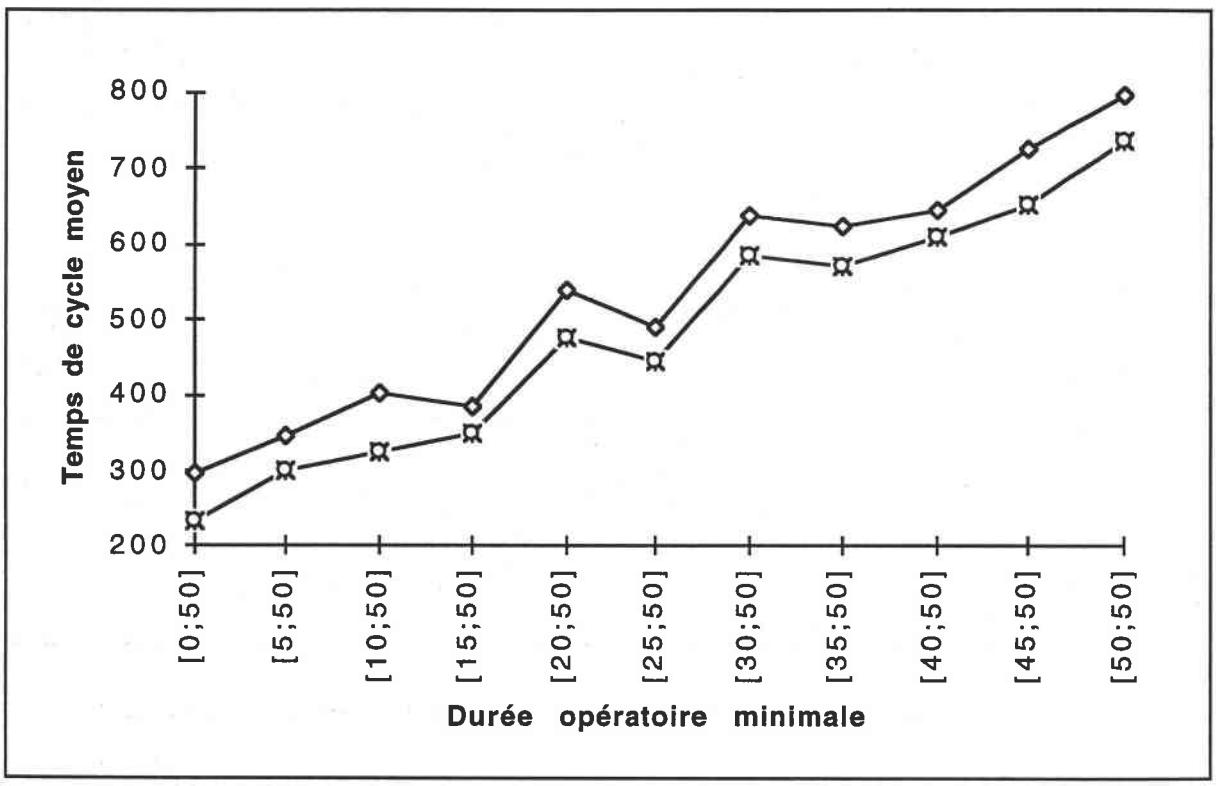

Figure II- 23 : Temps de cycle moyen suivant les modes de gestion

L'évolution de la durée minimale ne modifie pas l'allongement moyen des opérations exécutées sur les produits. Mais l'augmentation de la durée minimale entraîne un allongement du temps de cycle, quelle que soit la méthode de gestion adoptée (voir Figure II- 23). Si la fabrication est lancée au plus tôt (courbe ◊), le temps de cycle est significativement plus long que si la fabrication est lancée au plus tard (courbe o), ou à coût minimal (courbe *). Le délai de livraison suit la même évolution pour les trois modes de gestion.

Le temps de calcul est stable quel que soit le mode de gestion. Très court, si la fabrication est lancée au plus tôt (de l'ordre de 9 millième de seconde par produit). Le temps de calcul est doublé si la fabrication est lancée au plus tard, et multiplié par 5 si la fabrication est déterminée de manière à ce que le coût soit minimal.

La Figure II- 24 représente la productivité (courbes épaisses) et le rendement moyen (courbes fines) si on fabrique au plus tôt (courbe ◊), au plus tard (courbe o) et à coût minimal (courbe *). Plus la durée minimale des opérations augmente, plus il faut de temps pour réaliser un produit. Ceci induit une diminution de la productivité. Parallèlement, les produits ont des durées opératoires de plus en plus proches les unes des autres. Ceci permet de diminuer les écarts d'utilisation entre les différentes ressources et d'augmenter leur rendement moyen.

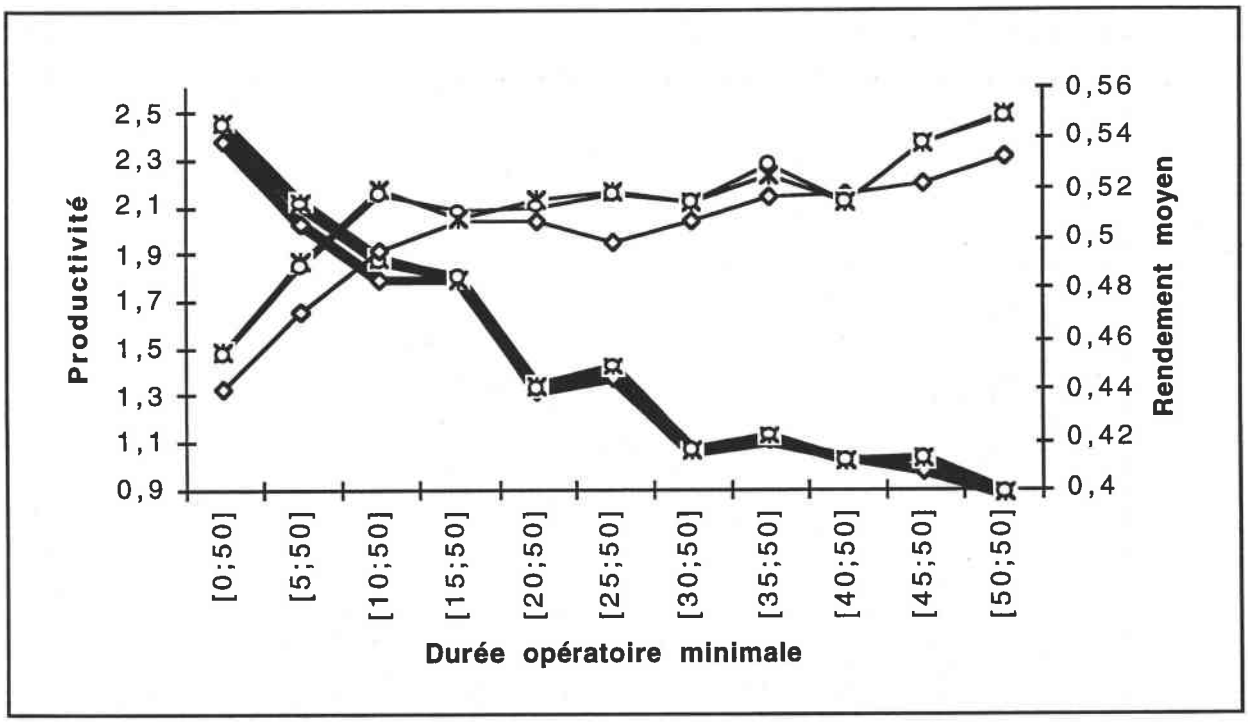

Figure II- 24 : Productivité et rendement des différents modes de gestion

Globalement, les écarts de rendement entre les machines diminuent au fur et à mesure que la durée opératoire minimale augmente, quelle que soit la méthode de gestion appliquée. La Figure II- 25 correspond au cas où on fabrique à moindre coût. Nous donnons les rendements minimaux et maximaux des machines (courbes Δ). Le rendement moyen des machines (courbe au centre) est encadré par les rendements d'écart moyen (courbes en pointillé).

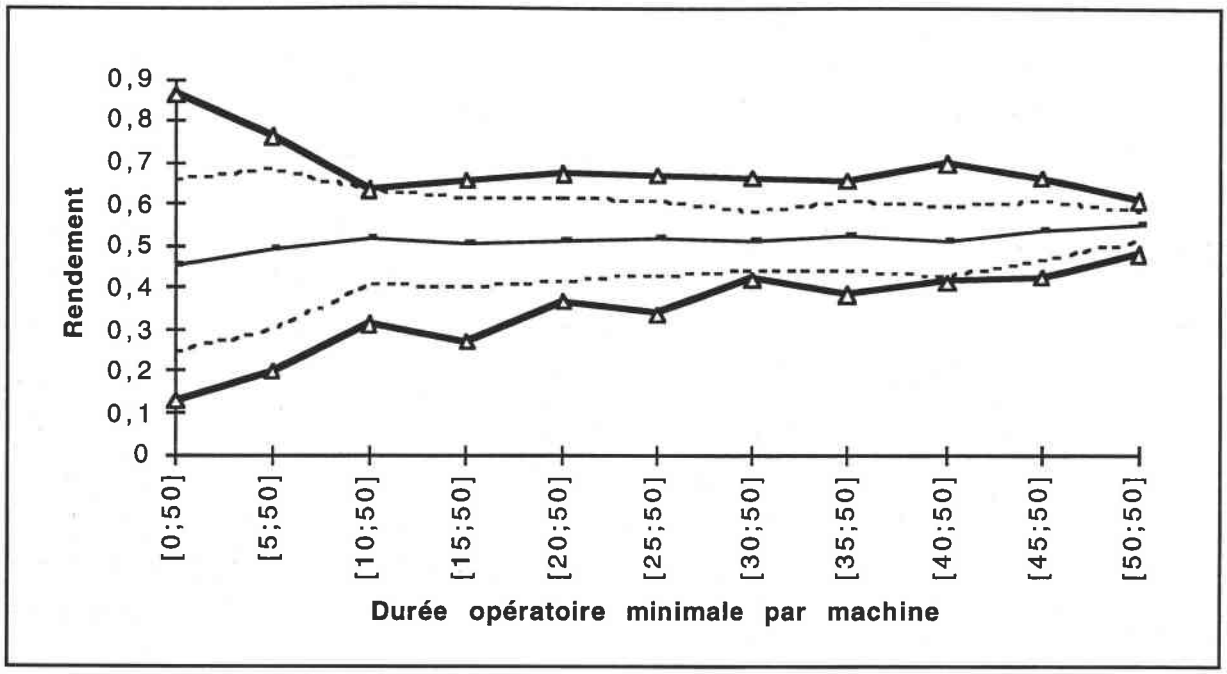

Figure II- 25 : Rendement dans le cas d'une fabrication à coût minimal

II.6.5. Conclusion

Dans cette section, nous avons comparé les différents modes de gestion en temps réel présentés dans les sections précédentes : fabriquer au plus tôt, au plus tard et à moindre coût. Notre objectif est d'obtenir une gestion qui maximise la productivité du système de production à long terme. Pour mieux comprendre les effets des modes de gestion sur la productivité, nous nous sommes limités à l'étude de situations où les produits passent successivement sur m machines.

Nous avons défini des indicateurs pour mesurer la productivité du système. L'évaluation dans le pire des cas du *makespan* ne permet pas de trancher quant à la supériorité d'un des trois modes de gestion en temps réel sur les autres. Cependant, cette analyse montre que les trois modes donnent de meilleurs résultats si les durées minimales des opérations exécutées sur une même machine sont proches d'un produit à l'autre. Cette situation est la plus courante en pratique.

L'étude en moyenne confirme les résultats théoriques. A savoir, fabriquer au plus tôt permet de maximiser la productivité à long terme quand les produits à réaliser sont identiques, ou presque identiques. Cette méthode de gestion très rapide du point de vue temps de calcul impose, cependant, un temps de cycle plus long que les autres méthodes.

Les deux autres méthodes ont un comportement très semblable et fournissent une productivité et un rendement excellents dans tous les cas. Nous n'avons pas pu mettre en évidence des situations où fabriquer à moindre coût induit une productivité significativement plus importante que produire au plus tard. Il semble que lorsque les charges des machines sont très différenciées, fabriquer à moindre coût soit plus avantageux.

II.7. CONCLUSION

II.7.1. Approche suivie

Dans ce chapitre, nous nous sommes intéressés à des systèmes de production dont les produits à ordonnancer ont une gamme linéaire. De plus, ils sont à durées opératoires mixtes, c'est-à-dire que les opérations se suivent sans attente possible et que les durées opératoires sont choisies par le gestionnaire entre deux limites. Le système de gestion est en temps réel et doit fournir, pour chaque nouvelle commande de produit, un ordonnancement. Ceci revient à déterminer les durées opératoires du produit et les périodes pendant lesquelles les ressources peuvent être utilisées pour réaliser les opérations. Le système de gestion ne peut modifier l'ordonnancement des produits en cours de fabrication. Un exemple d'application est proposé pour la gestion d'un atelier de traitement de surface que la société SODETEG T.A.I. située à Buc (78) a dû mettre en place. **L'objectif est d'obtenir une productivité maximale à long terme.**

Nous proposons, tout d'abord, **de terminer chaque produit le plus tôt possible**. Ceci signifie que nous cherchons à minimiser le *makespan* de chaque produit. Pour cela, nous définissons les dates d'exécution au plus tôt des opérations effectuées sur le produit, compatibles avec les contraintes de notre système de production, en appliquant l'Algorithme II-1 et l'Algorithme II- 2. Cette méthode de gestion est particulièrement efficace pour les ateliers de type *flow shop* et, en outre, permet de fournir en temps réel un délai de livraison précis.

Nous donnons une deuxième méthode de gestion qui consiste **à fabriquer le produit au plus tard**, tout en respectant la date de livraison fixée au plus tôt. Pour cela, nous définissons les dates d'exécution au plus tard des opérations, en appliquant l'Algorithme II- 3 et l'Algorithme II- 4. L'intérêt est de réduire le plus possible le temps pendant lequel des ressources sont utilisées pour fabriquer le produit. Ainsi, nous réduisons le coût de fabrication du produit, et libérons des périodes pendant lesquelles les opérations peuvent être exécutées sur les produits à venir. Cette méthode permet d'atteindre une meilleure productivité à long terme.

Nous proposons également de déterminer comment fabriquer chaque produit de manière à **minimiser le coût d'utilisation des ressources de chaque produit**, sans dépasser la date de livraison fixée au plus tôt. L'Algorithme II- 5 et l'Algorithme II- 6 permettent d'atteindre cet objectif. Nous utilisons également ces algorithmes pour contraindre le système à utiliser le moins possible les ressources-goulots et ainsi, espérer accroître la productivité du système.

Les algorithmes présentés dans ce chapitre sont tout à fait nouveaux et leur complexité très réduite permet leur utilisation en temps réel. Une partie des résultats obtenus a déjà été publiée sous forme d'article [Chauvet et *al.*, 1999a]. Dans le chapitre suivant, nous voulons élargir cette étude aux produits qui nécessitent des opérations d'assemblage.

II.7.2.Extension : Déplacement et collision des robots

II.7.2.1.Introduction

Un atelier de traitement de surface est constitué de cuves. Chacune contient une solution chimique ou électrolytique dans laquelle les pièces à réaliser sont immergées, comme le montre la Figure II- 1. Des robots permettent de transporter les pièces entre les cuves. Ces robots peuvent être des palans de manutention, des ponts roulants, ou encore des systèmes automatiques circulant sur un rail.

Pour simplifier la présentation, nous avons pris en compte les seuls déplacements en charge des robots, comme des opérations à réaliser sur le produit. Ainsi, nous avons supposé que les collisions entre robots de transport sont impossibles et nous avons négligé les déplacements à vide des robots. Nous montrons dans cette section comment nous pouvons étendre les méthodes de gestion proposées dans les sections précédentes à ces situations.

II.7.2.2.Chargement et déchargement

Quand un robot vient déposer un produit dans une cuve, une phase délicate de descente du produit dans la solution chimique ou électrolytique est entreprise comme sur la Figure II- 26. De manière analogue à cette phase de déchargement du robot, une phase de chargement du robot est nécessaire à la fin de la réaction chimique ou électrolytique.

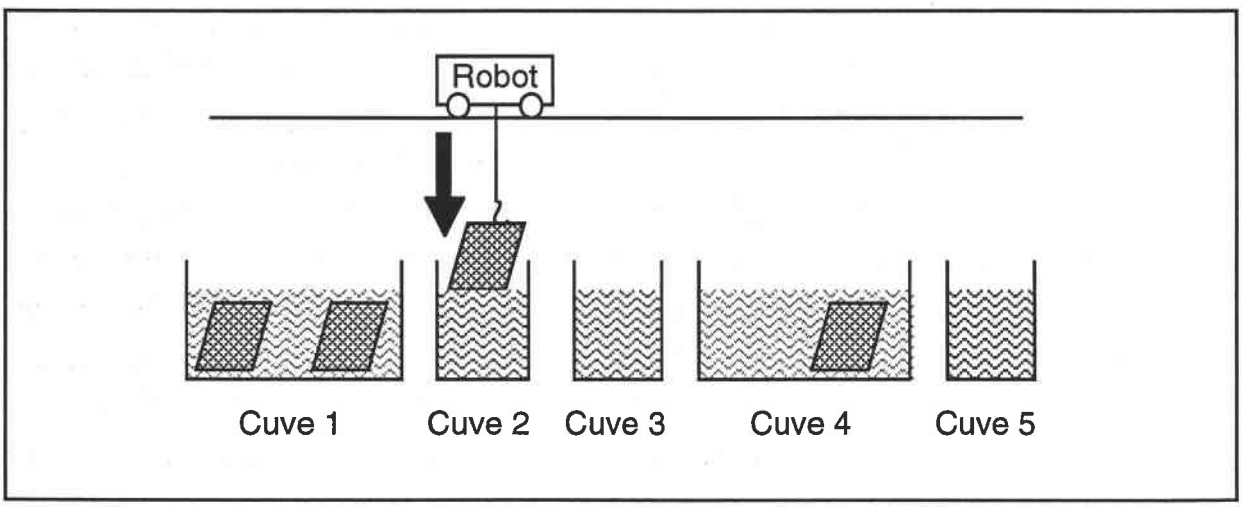

Figure II- 26 : Déchargement du robot

L'action de déchargement du robot est une phase de préparation nécessaire au début d'une opération chimique. Nous notons $e_{i,j}$ la durée de la phase de préparation nécessaire au début de

l'opération i à réaliser sur le produit j, que l'opération i soit une réaction chimique ou un transport. Nous posons $e_{i,j}$=0 si aucune préparation n'est nécessaire au début de l'opération i.

De manière analogue, l'action de chargement du robot est une phase post-opératoire qui suit une opération chimique. Nous désignons par $f_{i,j}$, la durée nécessaire à la phase post-opératoire qui suit l'opération i exécutée sur le produit j.

L'étude présentée dans les sections précédentes ne prenait pas en compte le déchargement et le chargement. Nous pouvons étendre facilement les algorithmes II- 1 à II- 6. Pour cela, il suffit de réduire chaque fenêtre $[a_{z,i,j}\ ;\ b_{z,i,j}]$ dans laquelle peut être exécutée l'opération i. Au lieu de l'ensemble de fenêtres $\{[a_{z,i,j}\ ;\ b_{z,i,j}]\ /\ 1\leq z\leq y_{i,j}\}$, nous appliquons ces algorithmes avec l'ensemble $\{[a_{z,i,j}+e_{i,j}\ ;\ b_{z,i,j}-f_{i,j}]\ /\ 1\leq z\leq y_{i,j}\}$. Une solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ fournie par un des différents algorithmes prend alors en considération les temps de chargement et de déchargement de manière transparente. Nous retenons alors la cuve sélectionnée pour réaliser l'opération i pendant la période $[T_{i,j}-e_{i,j}\ ;\ T_{i+1,j}+f_{i,j}]$.

II.7.2.3.Déplacement à vide

Les robots se déplacent soit pour transporter un produit d'une cuve à une autre (c'est ce que l'on appelle un déplacement en charge), soit pour aller à une cuve où il chargera un produit (c'est ce que l'on nomme un déplacement à vide). Dans la présentation que nous avons faite, seuls les déplacements en charge sont considérés.

Les algorithmes II- 1 à II- 6 peuvent être appliquées en prenant en compte les déplacements à vide. En effet, à chaque fenêtre $[a_{z,i,j}\ ;\ b_{z,i,j}]$ dans laquelle peut être exécutée l'opération de transport i, est associée un robot disponible. Nous connaissons la position initiale de ce robot à la date $a_{z,i,j}$. De même, nous connaissons la position qu'il doit rejoindre à la date $b_{z,i,j}$ pour déplacer un produit préalablement ordonnancé. Nous désignons par $e_{z,i,j}$ la durée nécessaire au robot pour aller de sa position précédente à une cuve du bain permettant de réaliser l'opération $(i+1)$. Nous appelons $f_{z,i,j}$ la durée nécessaire au robot pour, à partir d'une de ces cuves, rejoindre la position suivante. Ces durées ne dépendent pas de la cuve dans laquelle le produit est immergé pour réaliser l'opération $(i+1)$, car nous avons supposé que les cuves contenant les mêmes bains sont regroupées (dans le cas où cette hypothèse n'est pas vérifiée, nous utilisons des gammes alternatives pour modéliser les différents choix possibles). Pour une opération i qui ne correspond pas à un mouvement par robot, nous posons $e_{z,i,j}$=$f_{z,i,j}$=0.

Avant d'exécuter les algorithmes II- 1 à II- 6, nous retranchons les temps de déplacements à vide de chaque fenêtre $[a_{z,i,j}\ ;\ b_{z,i,j}]$ où peut être exécutée l'opération i. Au lieu de la fenêtre $[a_{z,i,j}\ ;\ b_{z,i,j}]$ pour $1\leq z\leq y_{i,j}$, [Chauvet et *al.*, 1999a] nous appliquons ces algorithmes avec $[a_{z,i,j}+e_{z,i,j}\ ;\ b_{z,i,j}-f_{z,i,j}]$ pour $1\leq z\leq y_{i,j}$. La solution $\{T_{i,j}\}_{1\leq i\leq m_j+1}$ fournie par un des algorithmes tient alors compte des déplacements à vide. Ainsi, un robot est sélectionné pour

réaliser l'opération de transport i pendant la période $[T_{i,j}\ ;\ T_{i+1,j}]$ et les déplacements à vide pendant les périodes $[T_{i,j}\text{-}e_{z,i,j}\ ;\ T_{i,j}]$ et $[T_{i+1,j}\ ;\ T_{i+1,j}\text{+}f_{z,i,j}]$.

La Figure II- 27 présente les résultats de l'application de l'Algorithme II- 6 si les chargements, déchargements et les déplacements à vide sont pris en compte. Les plages grisées correspondent aux périodes pendant lesquelles les ressources sont utilisées. Elles sont étendues par des portions noires pour permettre les déchargements et les chargements du produit au début et respectivement à la fin des opérations chimiques. Les zones en damier sont les périodes réservées pour permettre les déchargements, les chargements du produit et les déplacements à vide au début et à la fin des opérations de transport.

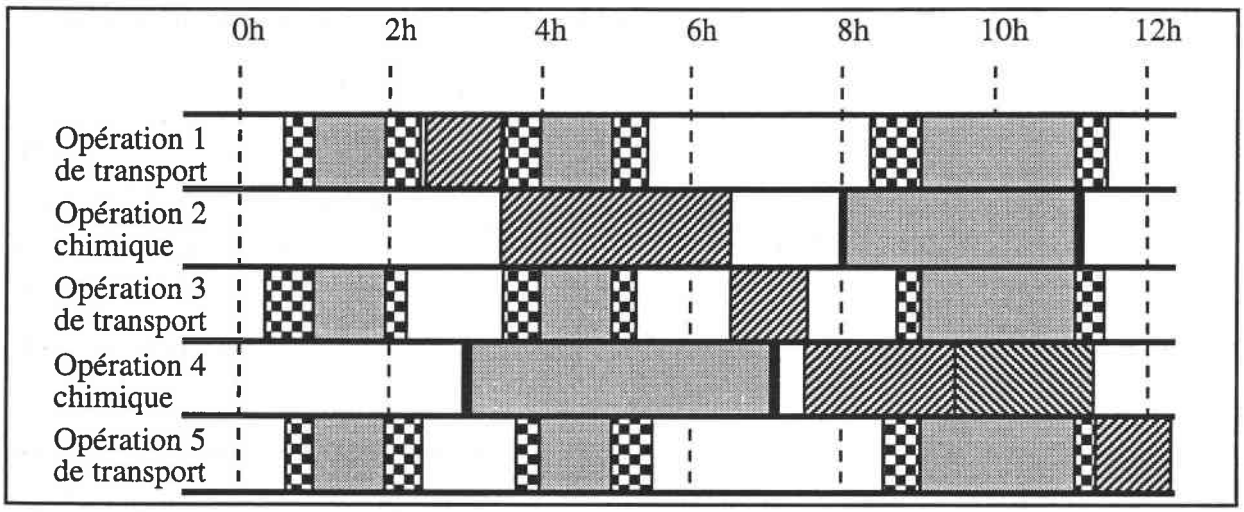

Figure II- 27 : Prise en compte des chargements et des déplacements à vide

Seule la période de transport en charge du robot $[T_{i,j}\ ;\ T_{i+1,j}]$ est réservée et devient indisponible pour tout autre utilisation. Nous retenons également en mémoire les positions attendues du robot aux dates $T_{i,j}$ et $T_{i+1,j}$. Les déplacements à vide pour atteindre ces positions seront reconsidérés si nous essayons de l'utiliser pendant l'une des périodes $[a_{z,i,j}\ ;\ T_{i,j}]$ et $[T_{i+1,j}\ ;\ b_{z,i,j}]$ pour déplacer un autre produit.

Ainsi, notre étude permet de déterminer le trajet que doivent suivre les robots. Le routage des robots est ainsi élaboré tout en tenant compte des contraintes d'ordonnancement de notre système de production.

II.7.2.4. Collision entre robot

Dans différents ateliers de traitement de surface, les robots sont des ponts roulants, ou encore des systèmes automatiques circulant sur un rail. Dans ces cas, tout croisement de deux robots sur un rail est interdit, et toute collision doit être évitée, voir sur la Figure II- 28. Nous avons considéré jusque là que ces collisions étaient impossibles.

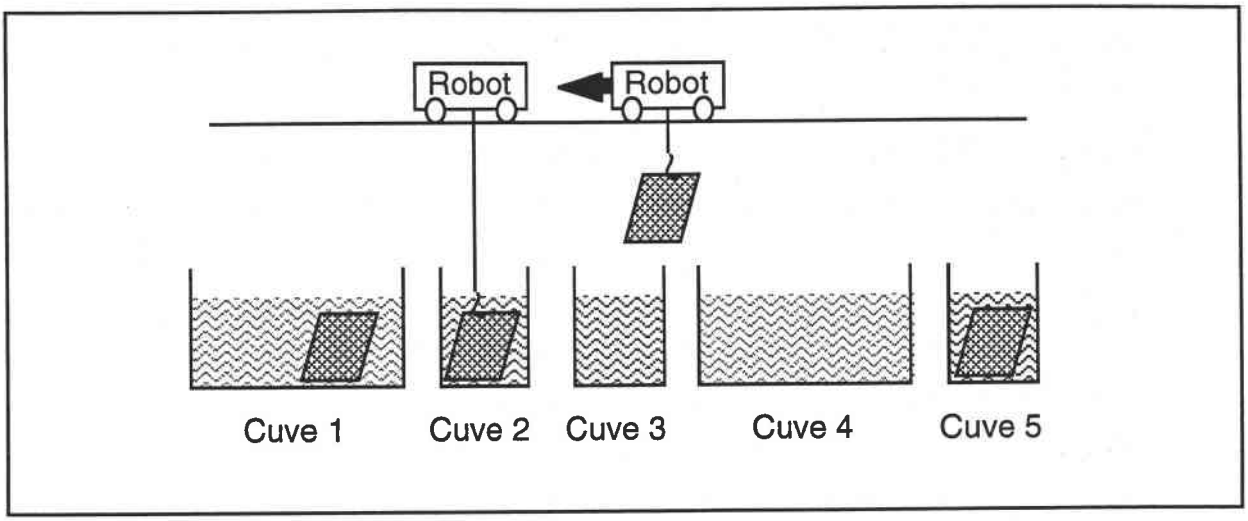

Figure II- 28 : Collision entre deux robots

Pour les prendre en compte, il suffit de considérer les rails comme une ressource à part entière. Un déplacement nécessite alors non seulement qu'un robot soit libre, mais aussi que les portions de rail qu'il doit emprunter soient disponibles. Ces portions de rail sont appelées cantons. Chaque canton ne peut porter qu'un robot à la fois. Deux cantons successifs sont définis de manière à ce qu'une petite partie leur soit commune pour interdire tout croisement de deux robots.

Pour prendre en compte les déplacements à vide, nous utilisons des gammes alternatives (introduites dans le Chapitre IV). Pour chaque routage possible du produit entre deux cuves, nous définissons une gamme différente décrivant les utilisations successives des cantons. Le système de gestion retient alors le routage permettant d'ordonnancer le produit de manière optimale.

En outre, les gammes alternatives permettent de considérer que les durées de déplacements de robot sont quelconques et que les cuves contenant un même bain ne sont pas regroupées (voir Hypothèse II- 12).

II.7.2.5. Conclusion

Dans cette section, nous avons montré comment l'étude présentée dans les sections précédentes peut s'étendre pour prendre en compte des spécificités des ateliers de traitement de surface. Ces spécificités concernent les robots permettant de déplacer les produits entre les cuves dans lesquelles les pièces à réaliser sont immergées.

CHAPITRE III

PROBLEMES AVEC GAMMES D'ASSEMBLAGE ET DE DESASSEMBLAGE

Résumé

Dans ce chapitre, nous proposons d'étendre les méthodes de gestion en temps réel présentées dans le chapitre précédent à des systèmes de production fabriquant des produits nécessitant des opérations d'assemblage ou de désassemblage. Ces méthodes de gestion restent applicables à des systèmes de production du type des *problèmes mixtes*. Ainsi, elles permettent d'appréhender une large variété de situations Nous illustrons notre étude par la gestion d'une unité de fabrication de produits alimentaires. Les algorithmes développés sont originaux et aussi performants que ceux introduits dans le premier chapitre. Ils permettent de déterminer le délai de livraison de chaque produit, et d'ordonnancer chaque produit de manière à minimiser le temps et le coût d'utilisation des ressources.

Sommaire

III.1. INTRODUCTION

Dans ce chapitre, nous nous intéressons à des systèmes de production dont les caractéristiques sont les mêmes que celles qui sont présentées dans le chapitre précédent. La seule exception est qu'il ne s'agit plus de gammes linéaires. La caractéristique concernant les gammes devient :

> « Chaque produit subit un ensemble d'opérations sur lequel est défini un ordre partiel. (Dans le cas plus restrictif des gammes linéaires, cet ordre est total). Toute opération qui suit (immédiatement) plusieurs opérations, est une opération d'assemblage, alors que toute opération qui est suivie (immédiatement) de plusieurs opérations, est une opération de désassemblage. »

L'objectif est de trouver un ordonnancement en temps réel pour chaque produit dès que la commande est passée. Nous illustrons notre étude par la gestion d'une unité de préparation de produits alimentaires.

III.1.1. L'ordonnancement des unités de produits alimentaires

Une telle unité prépare généralement les aliments par lot. Chaque lot est, pour nous, un produit. Les produits doivent subir un certain nombre d'opérations comme celles qui sont représentées dans la Figure III- 1. Elles peuvent être de plusieurs natures :

- Des opérations de transformation des produits qui permettent de modifier les caractéristiques des produits par cuisson, séchage…
- Des opérations de déplacements des produits entre les différents postes pendant les opérations de transport à l'aide de chariots, de convoyeurs…
- Des ajouts d'ingrédients à chaque opération d'assemblage.
- Des opérations de découpe, pour la volaille par exemple. Chacune des parties découpées subit alors sa propre série d'opérations qu'il est nécessaire d'ordonnancer. Les opérations de découpe sont des désassemblages. En fait, le produit obtenu est constitué de l'ensemble des parties découpées.

Dans ces unités, certaines opérations, comme la cuisson ou le séchage, doivent être exécutées successivement avec le minimum d'attente. En effet, tout arrêt en cours de traitement peut être préjudiciable à la qualité du produit et réduit d'autant le délai de péremption du produit. Mais il est autorisé d'allonger, dans une certaine mesure, la durée des opérations sans dégrader le produit.

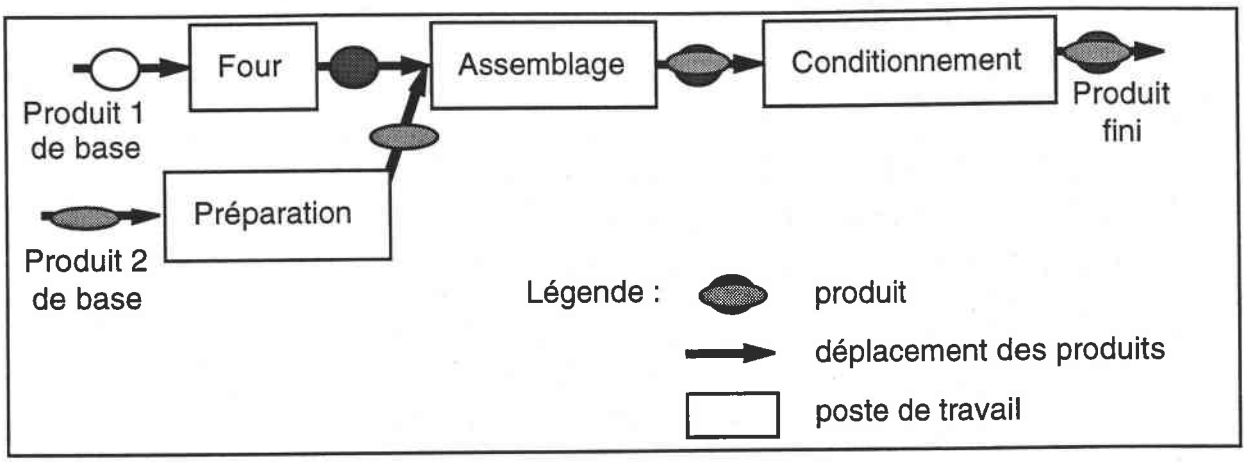

Figure III- 1 : Exemple d'une unité de préparation de produits cuisinés

A chaque fois qu'un événement aléatoire se produit (i.e. une nouvelle commande ou une panne), les opérations restant à réaliser sont ordonnancées. L'ordonnancement se fait sans connaître les commandes à venir. Les références bibliographiques qui concernent la gestion de ce type de fabrication sont inexistantes. Comme les durées opératoires ne sont pas fixées, les heuristiques classiques d'ordonnancement [Baker, 1974] [Song et *al.*, 1993] ne sont pas applicables. Dans la suite, nous proposons des heuristiques spécifiques pour ces problèmes.

III.1.2. Approche proposée

Pour faciliter la lecture, ce chapitre est organisé comme le précédent. Ainsi, dans la section 2, nous précisons les hypothèses. Le but du chapitre est de donner des modes de gestion en temps réel qui proposent un ordonnancement de chaque produit dès que sa commande est connue. **L'objectif global est d'obtenir une gestion qui maximise la productivité du système de production à long terme**. Le premier mode de gestion consiste **à fabriquer le produit au plus tôt** (c'est l'objet de la section 3). En outre, il permet de fournir au client un délai minimum de livraison. Ensuite, nous proposons un deuxième mode de gestion qui revient **à lancer le produit en fabrication au plus tard**, tout en livrant le produit au plus tôt (voir section 4). Enfin, nous donnons un troisième mode de gestion dont le but est de **minimiser le coût d'utilisation des ressources** (voir section 5), tout en livrant le produit au plus tôt.

Dans la section 6, nous appliquons le premier mode de gestion et évaluons les effets de la limitation du temps de stockage sur la productivité du système de production. Chaque section est illustrée par un exemple de gestion d'une unité de fabrication de produits alimentaires. Nous supposons que les conflits de ressources entre les opérations pouvant être effectuées en parallèle sur le même produit, sont impossibles. Dans les extensions étudiées (voir section 7), nous considérons les cas où ces conflits peuvent apparaître. Une partie des travaux de ce chapitre ont été présentés [Chauvet et Proth, 1998a] [Chauvet et Proth, 1998b] [Chauvet et *al.*, 1998b].

III.2. HYPOTHESES

Dans cette section, nous décrivons le cadre d'étude. Les hypothèses que nous faisons sont les mêmes que celles faites au chapitre précédent, mis à part l'Hypothèse III- 4 qui diffère, et les Hypothèses III- 15 et III- 16 qui sont ajoutées. Nous rappelons l'ensemble des hypothèses sans détailler celles qui sont identiques au Chapitre II. Les notations nécessaires à la lecture de ce chapitre sont reprises.

Hypothèse III- 1 : *L'ordonnancement est en temps réel.* Nous devons ordonnancer chaque produit dès que sa commande arrive. Nous numérotons les demandes de 1 à n dans l'ordre de leur arrivée. Nous désignons par J l'ensemble des demandes.

Hypothèse III- 2 : *La matière première et les composants du produit à fabriquer sont disponibles.* Nous supposons que la matière première et les composants du produit à fabriquer sont disponibles à la date r_j.

Hypothèse III- 3 : *L'ordonnancement d'un produit ne remet pas en question l'ordonnancement des précédents.* Nous considérons que l'ordonnancement d'un produit est défini une fois pour toutes et ne peut donc être remis en question quand une nouvelle commande apparaît. Nous examinons des systèmes qui autorisent le ré-ordonnancement dans les extensions de ce chapitre (voir section 7).

Hypothèse III- 4 : La gamme de chaque produit peut comporter des opérations d'assemblage ou de désassemblage.

Nous supposons que les gammes des produits à réaliser peuvent comporter des assemblages et des désassemblages, c'est-à-dire que les opérations doivent respecter un ordre partiel. La Figure III- 2 représente une telle gamme. Chaque opération est représentée par un arc. Les opérations d'assemblage sont précédées de plusieurs opérations, tandis que les opérations de désassemblage sont suivies de plusieurs opérations. Une gamme linéaire qui induit un ordre total sur les opérations est un cas particulier des gammes étudiées dans ce chapitre.

Les gammes des produits peuvent être différentes d'un produit à l'autre. Cela signifie que les opérations peuvent avoir des durées différentes et peuvent nécessiter des ressources distinctes. Pour le produit j dont la gamme comporte m_j opérations, nous notons I_j l'ensemble des opérations à effectuer sur le produit j.

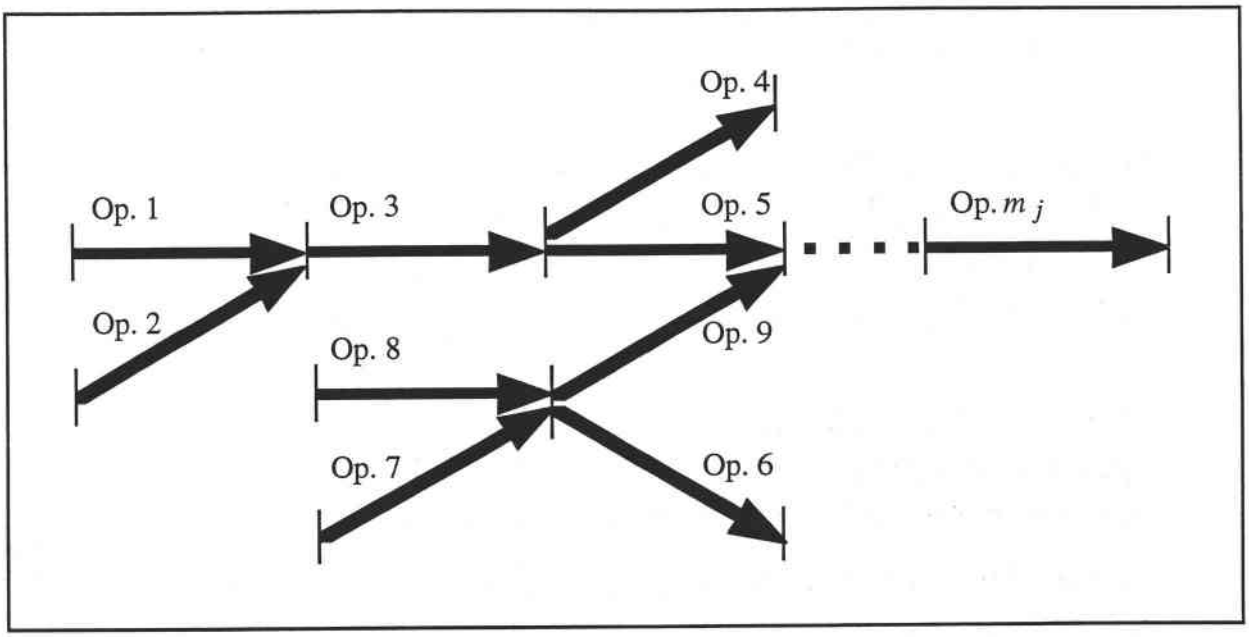

Figure III- 2 : Représentation de gammes avec assemblage et désassemblage

Hypothèse III- 5 : *Nous connaissons les ressources interchangeables en cours d'opération.*

Hypothèse III- 6 : *Le temps de changement de ressource en cours d'opération (lorsque cela est possible) est nul.*

Hypothèse III- 7 : *Chaque opération peut nécessiter l'utilisation de plusieurs types de ressources.*

Hypothèse III- 8 : *L'utilisation d'une ressource ne dépend pas d'autres utilisations de la même ressource.*

Hypothèse III- 9 : *Les exemplaires d'un même type de ressources sont identiques.*

Le coût d'utilisation des ressources nécessaires pour effectuer l'opération i sur le produit j par unité de temps est : $s_{i,j}$.

Hypothèse III- 10 : *Les disponibilités des ressources sont connues.*

Nous supposons défini l'ensemble $W_{i,j}$ des $y_{i,j}$ fenêtres de temps durant lesquelles l'opération i peut être effectuée sur le produit j : $W_{i,j}=\{[a_{z,i,j}\ ;\ b_{z,i,j}]\ /\ 1 \leq z \leq y_{i,j}\}$. Les dates de début et de fin de la z -ième période pendant laquelle l'opération i peut être exécutée sur le produit j sont notées respectivement $a_{z,i,j}$ et $b_{z,i,j}$. L'Annexe I présente les procédures nécessaires à la définition et à la gestion de ces fenêtres.

Hypothèse III- 11 : *Les opérations se suivent sans interruption.* Puisque les opérations se suivent sans attente, l'instant de fin d'une opération est également l'instant de début des opérations qui la suivent. Nous représentons les données relatives à un produit j par un graphe

orienté $G_j=(H_j,I_j)$, comme celui de la Figure III- 3. Ce graphe est composé d'un ensemble H_j de sommets marquant les instants de début et de fin des opérations. Ces sommets sont reliés par m_j arcs représentant l'ensemble I_j des opérations à exécuter.

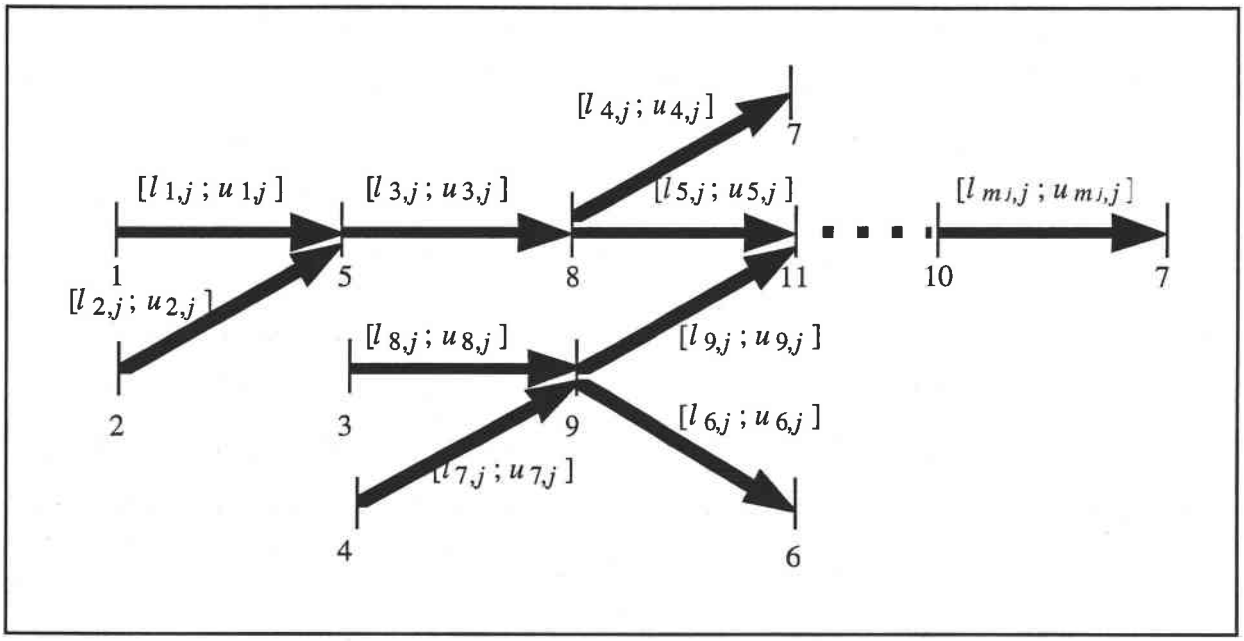

Figure III- 3 : Graphe représentant la gamme d'un produit

A partir de ce graphe $G_j=(H_j,I_j)$, nous définissons les successeurs et prédécesseurs d'un sommet. Un sommet k de l'ensemble H_j est **successeur du sommet** h si et seulement s'il existe un arc i de I_j reliant h à k. Nous pouvons noter cet arc $i=(h,k)$. Le sommet h est un **prédécesseur** de k. Le sommet h (respectivement k) est l'extrémité initiale (respectivement terminale) de l'arc i.

Hypothèse III- 12 : *Les durées des opérations sont contrôlables.* La durée $P_{i,j}$ de l'opération i peut être choisie dans un intervalle de valeurs admissibles : $P_{i,j} \in [l_{i,j} \ ; \ u_{i,j}]$.

Hypothèse III- 13 : *La préemption est interdite.* Cependant, une adaptation possible de notre étude aux cas où les opérations peuvent être interrompues pendant un laps de temps puis reprises est présentée dans les extensions de ce chapitre (voir section 7).

Hypothèse III- 14 : *L'objectif est d'ordonnancer chaque produit dès que sa commande est enregistrée de manière à maximiser la productivité du système à long terme.* Nous désignons par $T_{h,j}$ la date à laquelle correspond l'instant d'indice h. La date de début de l'opération $i=(h,k)$ effectuée sur le produit j est notée $B_{i,j}=T_{h,j}$ et sa date de fin $C_{i,j}=T_{h,k}$. Ordonnancer un produit consiste à déterminer dans quelle période $Z_{i,j}$ l'opération i doit être exécutée sur le produit j, ainsi que ses dates de début $B_{i,j}$ et de fin $C_{i,j}=B_{i,j}+P_{i,j}$. Pour atteindre une

productivité maximale du système à long terme, nous étendons les méthodes de gestion présentées au chapitre précédent :

a) **terminer chaque produit le plus tôt possible**, c'est-à-dire minimiser son *makespan* (voir section 3),

b) **lancer le produit en fabrication au plus tard**, tout en respectant la date de livraison fixée au plus tôt (voir section 4),

c) **minimiser le coût d'utilisation des ressources de chaque produit**, tout en respectant la date de livraison fixée au plus tôt (voir section 5).

Hypothèse III- 15 : Deux opérations ne peuvent avoir à la fois une même opération qui les précède, et une même opération qui les suit.

Plus précisément, nous supposons que le graphe non-orienté associé $G_j=(H_j,I_j)$ n'admet aucun cycle. La notion de cycle utilisée ici est traditionnelle dans la théorie des graphes [Gondran et Minoux, 1979]. Formellement, nous appelons **chaîne** de h_1 à h_p la suite de sommets distincts $(h_1, h_2, \ldots, h_p)$ qui vérifient soit $(h_q,h_{q+1})\in I_j$, soit $(h_{q+1},h_q)\in I_j$, pour $1\leq q<p$. Nous désignons par **cycle** une chaîne reliant h_1 à h_p qui vérifie soit $(h_p,h_1)\in I_j$, soit $(h_1,h_p)\in I_j$.

Dans les extensions à ce chapitre (voir section 7), nous proposons des adaptations des approches présentées dans la suite aux cas (peu fréquents) où le graphe non-orienté associé $G_j=(H_j,I_j)$ admet des cycles.

Sans perte de généralité, nous supposons que le graphe est **connexe**, i.e. il existe une chaîne qui relie tout couple de sommets. Dans le cas contraire, nous définirions un produit pour chaque partie connexe du graphe. Le graphe $G_j=(H_j,I_j)$ donnant les précédences entre les opérations est donc un **arbre** muni d'une orientation (*tree precedence*, en anglais) [Gondran et Minoux, 1979]. De plus, puisque la gamme du produit j comporte m_j opérations, le nombre de sommets du graphe G_j est $|H_j|=|I_j|+1=(m_j+1)$. On note $|E|$ le cardinal de l'ensemble E.

Nous numérotons l'ensemble des sommets du graphe G_j de manière à ce que, **pour tout sommet d'indice h, il n'existe pas plus d'un sommet k parmi les successeurs et prédécesseurs directs de h, ayant un indice supérieur à h**. Les sommets de la Figure III- 3 sont numérotés de cette manière. Formellement, pour tout sommet $h\in H_j$, nous avons :

$$|\{k>h \ / \ (h,k)\in I_j\}|+|\{k>h \ / \ (k,h)\in I_j\}|\leq 1.$$

Le graphe étant un arbre, il est possible d'obtenir une telle numérotation [Chauvet et *al.*, 1998b]. De plus, elle peut être déterminée par une procédure arborescente. Un tel algorithme

peut être implémenté avec une complexité égale à $O(m_j)$. **Cette numérotation est essentielle pour les algorithmes présentés dans la suite.**

La numérotation des sommets n'est pas unique. En particulier, pour le cas d'une gamme linéaire comportant m_j opérations, il existe 2^{m_j} numérotations possibles pour les sommets du graphe associé. Parmi celle-ci, nous avons choisi, au chapitre précédent, de numéroter les sommets dans l'ordre chronologique.

Hypothèse III- 16 : Il n'existe pas de conflits de ressources entre les opérations pouvant être effectuées en parallèle sur un même produit.

Autrement dit, si deux opérations i_1 et i_2 utilisent la même ressource dont un seul exemplaire est disponible, elles ne peuvent être exécutées en parallèle. Deux conditions suffisantes sont données ci-dessous. Elles sont basées sur les durées séparant deux instants de début et de fin des opérations i_1 et i_2.

Puisque le graphe $G_j=(H_j,I_j)$ est un arbre, il existe une et seule chaîne reliant tout couple de sommets. Nous désignons par **durée minimale** séparant l'instant correspondant à h_1 de celui correspondant à h_p, la valeur $L(h_1,h_p)=\sum_{\substack{r=1,\ldots,p-1\\(h_r,h_{r+1})\in I_j}} l_{(h_r,h_{r+1}),j} - \sum_{\substack{r=1,\ldots,p-1\\(h_{r+1},h_r)\in I_j}} u_{(h_r,h_{r+1}),j}$. Dans cette formule, $(h_1, h_2, \ldots, h_p)$ désigne la chaîne reliant h_1 à h_p. Quant à $l_{(h,k),j}$ et $u_{(h,k),j}$, elles sont les durées minimale et maximale séparant l'instant relatif au sommet h de l'instant relatif à k i.e. $l_{(h,k),j} \le T_{k,j}-T_{h,j} \le u_{(h,k),j}$, pour $(h, k)\in I_j$. De manière analogue, nous désignons par **durée maximale** séparant l'instant correspondant à h_1 de celui correspondant à h_p, la valeur $U(h_1,h_p)=\sum_{\substack{r=1,\ldots,p-1\\(h_r,h_{r+1})\in I_j}} u_{(h_r,h_{r+1}),j} - \sum_{\substack{r=1,\ldots,p-1\\(h_{r+1},h_r)\in I_j}} l_{(h_r,h_{r+1}),j}$.

Nous démontrons (par récurrence sur le nombre de sommets de la chaîne) que la durée séparant les instants $T_{h_1,j}$ relatif au sommet h_1 de $T_{h_p,j}$ relatif au sommet h_p est comprise entre les durées minimale et maximale séparant les instants correspondants à h_1 et h_p, i.e. nous avons

$$L(h_1,h_p) \le T_{h_p,j}-T_{h_1,j} \le U(h_1,h_p).$$

Nous pouvons également montrer que les durées minimale et maximale sont asymétriques l'une par rapport à l'autre, c'est-à-dire que $L(h_1,h_p)=-U(h_p,h_1)$.

Deux opérations $i_1=(h_1,k_1)$ et $i_2=(h_2,k_2)$ utilisant la même ressource, ne peuvent être exécutées en parallèle, si l'une des conditions suivantes est vérifiée :

a) La durée minimale séparant la fin de i_1 du début de i_2 est positive, i.e.
$T_{h_2,j}-T_{k_1,j} \ge L(k_1,h_2)=-U(h_2,k_1) \ge 0$.

b) La durée minimale séparant la fin de i_2 du début de i_1 est positive, i.e.
$T_{h_1,j}$-$T_{k_2,j} \geq L(k_2,h_1)$=-$U(k_1,h_2) \geq 0$.

La condition a) assure que l'opération $i_1=(h_1,k_1)$ est exécutée avant l'autre opération $i_2=(h_2,k_2)$, tandis que la condition b) garantit que les opérations sont exécutées dans l'ordre inverse. Pour vérifier l'une des deux conditions, il suffit de calculer les durées reliant les extrémités de chaque opération i aux extrémités des autres opérations. Pour chaque opération i, ces calculs peuvent être obtenus à partir d'algorithmes classiques de calcul de longueur de chemins dans l'arbre G_j en à $O(m_j)$. Pour une gamme linéaire, l'Hypothèse III- 16 est toujours vérifiée.

Dans le cas de gammes où deux opérations ne vérifient aucune des deux conditions, les méthodes proposées dans les trois sections suivantes ne garantissent pas que les opérations ne soient pas exécutées simultanément. Si les ressources ne sont pas en nombre suffisant, nous disons qu'il y a conflit de ressources. Il faut alors, au moment d'ordonnancer, déterminer laquelle des deux opérations doit être exécutée avant l'autre. Nous examinons cette situation dans les extensions de ce chapitre (voir section 7).

Les opérations $i_1=(h_1,k_1)$ et $i_2=(h_2,k_2)$ sont toujours exécutées simultanément, si les deux conditions suivantes sont vérifiées :

a) La durée maximale séparant la fin de i_1 du début de i_2 est négative, i.e.
$T_{h_2,j}$-$T_{k_1,j} \leq U(k_1,h_2)$=-$L(h_2,k_1) \leq 0$.

b) La durée maximale séparant la fin de i_2 du début de i_1 est négative, i.e.
$T_{h_1,j}$-$T_{k_2,j} \leq L(k_2,h_1)$=-$U(k_1,h_2) \leq 0$.

Par conséquent, si les deux opérations i_1 et i_2 utilisent la même ressource dont un seul exemplaire est disponible, il n'est pas possible de lever le conflit. Il n'existe pas d'ordonnancement admissible.

III.3. MINIMISATION DU *MAKESPAN*

III.3.1. Introduction

Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit au plus tôt, i.e. qui minimise le *makespan* de chaque produit. Comme dans le Chapitre II, nous formalisons le problème avant de fournir un algorithme original de très bonne complexité. A notre connaissance, aucune autre étude prenant en compte toutes les hypothèses proposées dans la section précédente n'a été menée à ce jour.

III.3.2. Formalisation

Nous voulons déterminer les dates de début des opérations effectuées sur le produit j sachant que la date de disponibilité est r_j (voir contraintes (III-1)). Les dates $T_{h,j}$ et $T_{k,j}$ définissent les instants de début et de fin de chaque opération $(h,k)\in I_j$. La durée d'exécution $P_{(h,k),j}=T_{k,j}-T_{h,j}$ de chaque opération $i=(h,k)$ est à déterminer et doit vérifier : $l_{i,j}\leq P_{i,j}\leq u_{i,j}$ (voir contraintes (III-2)). Chaque opération i peut être exécutée dans l'une des $y_{i,j}$ fenêtres de temps de l'ensemble $W_{i,j}=\{[a_{z,i,j}\ ;\ b_{z,i,j}]\ /\ 1\leq z\leq y_{i,j}\}$ (voir contraintes (III-3)-(III-4)). La fenêtre retenue est la fenêtre d'indice $Z_{i,j}$ (voir contraintes (III-5)). Notre objectif est de minimiser la date d'achèvement du produit, i.e. la date de fin de la dernière opération effectuée sur le produit. Les contraintes (III-6) et (III-7) donnent les intervalles de définition de toutes les variables de décisions $T_{h,j}$ et $Z_{i,j}$. Le problème à résoudre s'écrit :

Problème III- 1 : Minimisation du *makespan* de chaque produit

$$\min \quad \max_{h\in H_j}\left\{T_{h,j}\right\}$$

$$\text{tel que}\begin{cases} r_j \leq T_{h,j} & \forall h\in H_j & \text{(III-1)}\\ l_{(h,k),j} \leq T_{k,j}-T_{h,j} \leq u_{(h,k),j} & \forall (h,k)\in I_j & \text{(III-2)}\\ a_{Z_{(h,k),j},(h,k),j} \leq T_{h,j} & \forall (h,k)\in I_j & \text{(III-3)}\\ T_{k,j} \leq b_{Z_{(h,k),j},(h,k),j} & \forall (h,k)\in I_j & \text{(III-4)}\\ 1\leq Z_{i,j} \leq y_{i,j} & \forall i\in I_j & \text{(III-5)}\\ T_{h,j}\in \Re^+ & \forall h\in H_j & \text{(III-6)}\\ Z_{i,j}\in \mathrm{N} & \forall i\in I_j & \text{(III-7)} \end{cases}$$

Comme le Problème II- 1, ce problème peut également se formaliser comme un problème de programmation linéaire mixte. L'approche que nous proposons dans les deux sections suivantes est similaire à celle décrite pour résoudre le Problème II- 1. Elle consiste à résoudre une suite de sous-problèmes du Problème III- 1 pour aboutir à la solution optimale en un temps polynomial.

III.3.3. Sélection des durées opératoires

L'objectif de cette section est de déterminer les dates $T_{h,j}$ fixées au plus tôt pour chaque instant h, si chaque opération (h,k) est exécutée sur le produit j dans la fenêtre donnée $Z_{(h,k),j}$, i.e. :

Problème III- 2 : Définition des dates au plus tôt des opérations quand leur fenêtre d'exécution est donnée

$$\min \quad \sum_{h \in H_j} \mu_h T_{h,j}$$

$$\text{tel que} \begin{cases} r_j \le T_{h,j} & \forall h \in H_j \quad \text{(III-1)} \\ l_{(h,k),j} \le T_{k,j} - T_{h,j} \le u_{(h,k),j} & \forall (h,k) \in I_j \quad \text{(III-2)} \\ a_{Z_{(h,k),j},(h,k),j} \le T_{h,j} & \forall (h,k) \in I_j \quad \text{(III-3)} \\ T_{h,j} \in \Re^+ & \forall h \in H_j \quad \text{(III-6)} \end{cases}$$

Nous recherchons une solution optimale pour ce problème quel que soit l'ensemble $\{\mu_h \ / \ h \in H_j\}$. Cette solution correspond au cas où chaque date est calée au plus tôt. Si une telle solution vérifie les contraintes (III- 4) (i.e. $T_{k,j} \le b_{Z_{(h,k),j},(h,k),j}$, pour $(h,k) \in I_j$), alors cette solution est réalisable pour le Problème III- 1 et est calée au plus tôt pour cet ensemble de fenêtres $\{Z_{(h,k),j}\}_{(h,k)\in I_j}$. Sinon, aucune solution n'est réalisable pour $\{Z_{(h,k),j}\}_{(h,k)\in I_j}$.

Algorithme III- 1 : Calage au plus tôt des opérations du produit [Chauvet et al., 1998b]

Procédure Définition des dates d'exécution au plus tôt des opérations effectuées sur le produit j.

Entrée : j est l'indice du produit considéré.

r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.
m_j est le nombre d'opérations à effectuer sur le produit j.
H_j est l'ensemble des instants marquant le début et la fin des opérations effectuées sur le produit j.
I_j est l'ensemble des opérations à effectuer sur le produit j.
$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.
$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.
$Z_{i,j}$ est la fenêtre dans laquelle on veut réaliser l'opération i sur le produit j, pour $i \in I_j$.
$a_{Z_{i,j},i,j}$ est la date de début de la période durant laquelle on réalise l'opération i, pour $i \in I_j$.

Sortie : $T_{h,j}$ est la date fixée au plus tôt qui correspond à h, où $h \in H_j$.

1. **Etape 1.**

1.1. **Pour** h allant de 1 à m_j+1 : *On note h l'indice de l'instant sélectionné.*

1.1.1. **Poser** $\tau_h := \max\left\{ r_j ; \max_{\substack{(f,h)\in I_j \\ f<h}} \left\{ \tau_f + l_{(f,h),j} \right\} ; \max_{\substack{(h,k)\in I_j \\ k<h}} \left\{ \tau_k - u_{(h,k),j} \right\} ; \max_{(h,k)\in I_j} \left\{ a_{Z_{(h,k),j},(h,k),j} \right\} \right\}$.

On note τ_h la borne inférieure de la date correspondant à h.

2. **Etape 2.**

2.1. **Poser** $T_{m_j+1,j} := \tau_{m_j+1}$.

2.2. **Pour** h allant de m_j à 1 : *On note h l'indice de l'instant sélectionné.*

2.2.1. **Poser** $T_{h,j} := \max\left\{ \tau_h ; \max_{\substack{(f,h)\in I_j \\ f>h}} \left\{ T_{f,j} + l_{(f,h),j} \right\} ; \max_{\substack{(h,k)\in I_j \\ k>h}} \left\{ T_{k,j} - u_{(h,k),j} \right\} \right\}$

En fait, dans le Problème III- 2, nous recherchons à déterminer les durées des opérations du produit comme nous définirions les longueurs des marches et des paliers d'un escalier, chaque marche et chaque palier correspondant à une opération. La longueur des marches est aussi ajustable dans une certaine limite de la même manière que les durées opératoires sont contrôlables.

L'Algorithme III- 1 permet de résoudre le Problème III- 2 par une méthode en deux étapes similaires à celles de l'Algorithme II- 1. Dans la première, on définit τ_h, borne inférieure des dates recherchées. Dans la seconde étape, on raffine cette borne pour obtenir la date optimale. La difficulté rencontrée ici provient du fait qu'à un même palier, plusieurs escaliers se rejoignent et doivent donc coïncider. La numérotation des sommets introduite à la suite de l'Hypothèse III-15 permet de construire simplement l'escalier. En effet, elle garantit que tous les escaliers qui doivent coïncider au niveau d'un palier sont construits avant que le palier lui-même soit bâti.

Exemple

Tableau III- 1 : Données relatives à l'exécution de l'Algorithme III- 1

ENTREES					RESULTATS INTERME-DIAIRES	SORTIES
Indice de l'instant h	Successeurs k	Durée minimale $l_{(h,k),j}$	Durée maximale $u_{(h,k),j}$	Début des périodes $a_{Z_{(h,k),j}(h,k),j}$	Borne inférieure de la date τ_h	Date au plus tôt $T_{h,j}$
1	3	1	$+\infty$	2	2	2
2					0	7
3	2 5	3 1	4 1	4 1	4	4
4	5	2	4	0	0	1
m_j=5	6	1	2	2	5	5
m_j+1=6					6	6

Nous proposons un exemple d'application de l'Algorithme III- 1. Les données relatives au produit j à réaliser sont fournies dans le Tableau III- 1 et la Figure III- 4. Le produit peut être réalisé à partir de la date r_j=0.

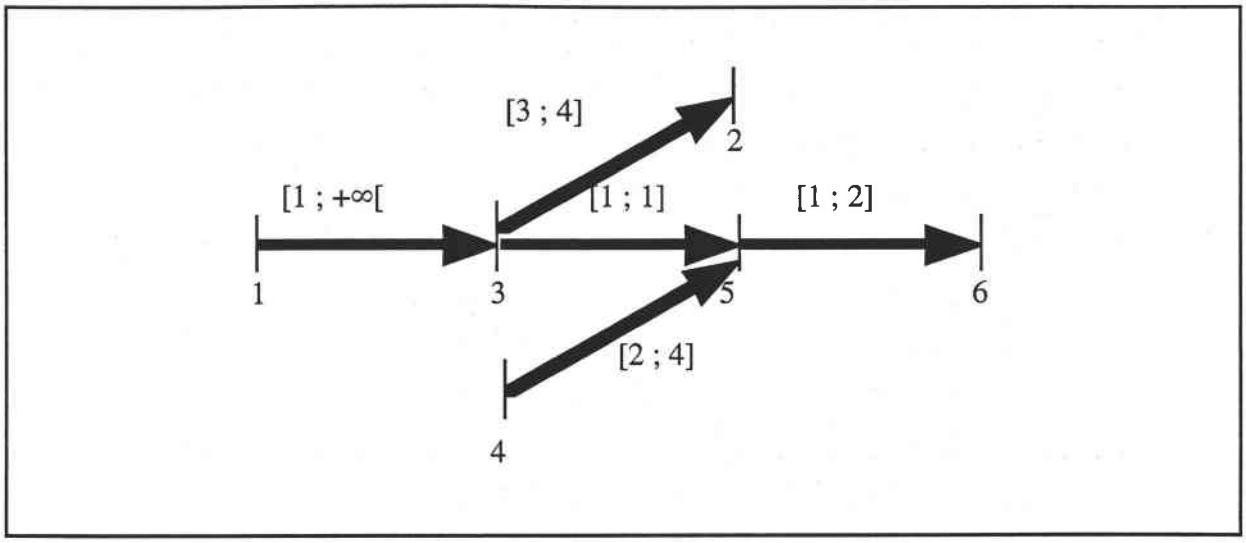

Figure III- 4 : Représentation graphique du produit à fabriquer

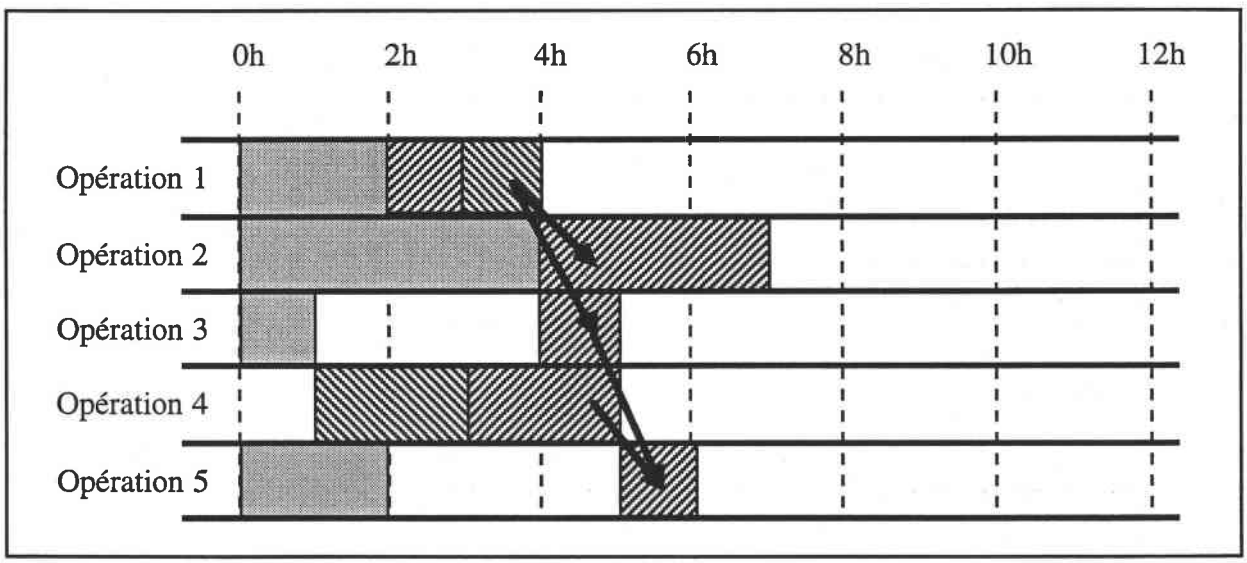

Figure III- 5 :Résultats fournis par l'Algorithme III- 1

Les résultats de l'exécution sont explicités sur la Figure III- 5. Les opérations effectuées sur le produit j à ordonnancer sont représentées par des zones hachurées. Dans le cas où la durée d'une opération dépasse la durée minimale, on utilise des hachures différentes pour la partie représentant le dépassement. Les arcs indiquent comment s'enchaînent les opérations.

Théorème III- 1 [Chauvet et *al.*, 1999a]

L'Algorithme III- 1 détermine les dates au plus tôt de toutes les opérations, i.e. il fournit une solution unique optimale au Problème III- 2.

Démonstration

a) Nous montrons tout d'abord que la solution $\{T_{h,j}\}_{1\le h\le m_j+1}$ fournie par l'Algorithme III- 1 est réalisable.

Pour tout $(h,k)\in I_j$., nous avons soit $h<k$, soit $h>k$.

Premier cas. Nous supposons que $h<k$. D'après la numérotation des sommets introduite à la suite de l'Hypothèse III- 15, pour tout sommet d'indice h, il ne peut exister qu'un seul sommet parmi les successeurs et prédécesseurs directs de h, ayant un indice supérieur à h. Ainsi, k est ce sommet et tous les autres successeurs et prédécesseurs directs de h ont un indice inférieur à h. Selon le pas 2.2.1 de l'algorithme, nous en déduisons que $T_{h,j}=\max\{\tau_h \ ; \ T_{k,j}-u_{(h,k),j}\}$. Par conséquent, nous avons $T_{k,j}-T_{h,j}\le u_{(h,k),j}$.

Nous pouvons aussi en déduire que $T_{h,j}+l_{(h,k),j}=\max\{\tau_h+l_{(h,k),j} \ ; \ T_{k,j}-u_{(h,k),j}+l_{(h,k),j}\}$. Cela implique, d'après le pas 1.1.1 appliqué à l'indice k, que $T_{h,j}+l_{(h,k),j}\le\max\{\tau_k \ ; \ T_{k,j}\}$. Comme, suivant les pas 2.1 et 2.2.1, $\tau_k\le T_{k,j}$, nous avons $T_{h,j}+l_{(h,k),j}\le T_{k,j}$, i.e. $T_{k,j}-T_{h,j}\ge l_{(h,k),j}$.

Deuxième cas. Nous supposons que $k<h$. D'après la numérotation des sommets, pour tout sommet d'indice k, il ne peut exister qu'un seul sommet parmi les successeurs et prédécesseurs directs de k, ayant un indice supérieur à k. Ainsi, h est ce sommet et tous les autres successeurs et prédécesseurs directs de k ont un indice inférieur à k. Par conséquent, selon le pas 2.2.1 de l'algorithme appliqué à l'indice k, nous avons $T_{k,j}=\max\{\tau_k \ ; \ T_{h,j}+l_{(h,k),j}\}$. Nous en déduisons que $T_{k,j}-T_{h,j}\ge l_{(h,k),j}$.

Nous pouvons aussi en déduire que $T_{k,j}-u_{(h,k),j}=\max\{\tau_k-u_{(h,k),j} \ ; \ T_{h,j}+l_{(h,k),j}-u_{(h,k),j}\}$. Cela implique, d'après le pas 1.1.1, que $T_{k,j}-u_{(h,k),j}\le\max\{\tau_h \ ; \ T_{h,j}\}$. Comme, suivant les pas 2.1 et 2.2.1, $\tau_h\le T_{h,j}$, nous avons $T_{k,j}-u_{(h,k),j}\le T_{h,j}$, i.e. $T_{k,j}-T_{h,j}\le u_{(h,k),j}$.

Dans les 2 cas, nous avons $l_{(h,k),j}\le T_{k,j}-T_{h,j}\le u_{(h,k),j}$. Par conséquent, la solution vérifie l'inéquation (III-2).

Suivant les pas 1.1.1, 2.1 et 2.2.1 de l'algorithme, on a $r_j\le\tau_h\le T_{h,j}$. Par conséquent, la solution vérifie l'inéquation (III-1).

Suivant les pas 1.1.1, 2.1 et 2.2.1 de l'algorithme, on a $a_{Z_{(h,k),j}(h,k),j}\le\tau_h\le T_{h,j}$, pour $(h,k)\in I_j$. Par conséquent, la solution vérifie les inéquations (III-3).

La solution fournie par l'algorithme satisfait les contraintes (III-1), (III-2), (III-3) et (III-6). Elle est réalisable.

b) Nous montrons ensuite que la solution $\{T_{h,j}\}_{1 \le h \le m_j+1}$ fournie par l'Algorithme III- 1 est optimale pour tout ensemble $\{\mu_h \ / \ 1 \le h \le m_j+1\}$.

Selon les contraintes (III-1) et (III-3), la valeur $\max\left\{r_j ; \max_{\substack{(h,k)\in I_j \\ h=1}}\left\{a_{Z_{(h,k),j},(h,k),j}\right\}\right\}$ est une borne inférieure de la date correspondant à $h=1$. D'après le pas 1.1.1 de l'algorithme, τ_1 est donc une borne inférieure de cette date.

Supposons que τ_k soit une borne inférieure de la date correspondant à k pour $k<h$ et $1<h \le m_j+1$. Selon les contraintes (III-1), (III-2) et (III-3), la valeur $\max\left\{r_j ; \max_{\substack{(f,h)\in I_j \\ f<h}}\left\{\tau_f + l_{(f,h),j}\right\}; \max_{\substack{(h,k)\in I_j \\ k<h}}\left\{\tau_k - u_{(h,k),j}\right\}; \max_{(h,k)\in I_j}\left\{a_{Z_{(h,k),j},(h,k),j}\right\}\right\}$ est une borne inférieure de la date correspondant à h. D'après le pas 1.2.1 de l'algorithme, τ_h est donc une borne inférieure de cette date, pour $1 \le h \le m_j+1$.

Selon le pas 2.1 de l'algorithme, $T_{m_j+1,j}$ est donc une borne inférieure de la date correspondant à m_j+1.

Supposons que T_k soit une borne inférieure de la date correspondant à k pour $k>h$ et $1<h \le m_j+1$. Selon les contraintes (III-2), puisque τ_h est une borne inférieure de la date correspondant à h, la valeur $\max\left\{\tau_h ; \max_{\substack{(f,h)\in I_j \\ f>h}}\left\{T_{f,j} + l_{(f,h),j}\right\}; \max_{\substack{(h,k)\in I_j \\ k>h}}\left\{T_{k,j} - u_{(h,k),j}\right\}\right\}$ est aussi une borne inférieure de cette date. D'après le pas 2.2.1 de l'algorithme, T_h est donc une borne inférieure de la date correspondant à h, pour $1 \le h \le m_j+1$.

On en conclut que l'algorithme détermine les dates d'exécution au plus tôt. Par conséquent, il fournit la solution optimale au Problème III- 2 quel que soit l'ensemble $\{\mu_h \ / \ 1 \le h \le m_j+1\}$ de valeurs strictement positives.

CQFD

Complexité

Dans tous les cas, l'Algorithme III- 1 affecte une valeur τ_h (au pas 1.1.1), puis $T_{h,j}$ (aux pas 2.1 et 2.2.1) relatives au sommet h. Par conséquent, cet algorithme réalise $2m_j+2$ affectations, où m_j est le nombre d'opérations et m_j+1 est le nombre de sommets sur le graphe associé. De plus, puisque le nombre d'arcs est égal à $|I_j|=m_j$, l'Algorithme III- 1 nécessite $(3|I_j|+2m_j+1)=6m_j+1$ comparaisons (effectuées aux pas 1.1.1 et 2.2.1). La complexité de **l'Algorithme III- 1 est en $O(m_j)$**, comme l'Algorithme II- 1 qui résout le même problème dans le cas particulier de gammes linéaires.

Tout algorithme doit au moins examiner chacune des opérations pour résoudre le problème. Par conséquent, aucun algorithme ne peut avoir une complexité inférieure à $O(m_j)$, c'est-à-dire inférieure à celle de l'Algorithme III- 1.

III.3.4. Sélection des fenêtres

L'objectif de cette section est de résoudre le Problème III- 1. Autrement dit, nous allons déterminer dans quelle fenêtre $Z_{i,j}$, l'opération i doit être effectuée de manière à ce que le produit j soit terminé le plus tôt possible. Nous utilisons les résultats de la section précédente qui permettent, pour un ensemble de fenêtres $\{Z_{i,j}\}_{1 \le i \le m_j}$, de déterminer les dates au plus tôt $\{T_{h,j}\}_{1 \le h \le m_j+1}$ d'exécution des opérations effectuées sur le produit j, où $i=(h,k) \in I_j$. Dans le cas où cet ensemble de dates ne vérifient pas les contraintes (III- 4) (i.e. $T_{k,j} \le b_{Z_{(h,k),j}(h,k),j}$ pour $(h,k) \in I_j$), alors aucune solution réalisable n'existe pour ces fenêtres.

Nous utilisons l'Algorithme III- 2 pour déterminer la solution optimale au Problème III- 1. Il examine les fenêtres relatives à chaque opération dans leur ordre chronologique sans qu'aucun retour arrière ne soit nécessaire.

Algorithme III- 2 : Choix des fenêtres pour une fin au plus tôt

Il est identique à l'Algorithme II- 2, mis à part les deux modifications suivantes :

a) Au pas 2.1, nous appliquons l'Algorithme III- 1 (au lieu d'exécuter l'Algorithme II- 1).

b) Aux pas 2.2 et 2.3.1.1, la date de fin de l'opération $i=(h,k)$ est $T_{k,j}$ (au lieu de $T_{i+1,j}$), pour tout $i \in I_j$.

Nous proposons un exemple d'application de l'Algorithme III- 2. Les données relatives au produit j à réaliser sont fournies dans le Tableau III- 2. Les durées opératoires considérées sont celles de la Figure III- 1. Le produit peut être réalisé à partir de la date $r_j=0$.

Les résultats de l'exécution sont explicités sur la Figure III- 6 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Exemple

Tableau III- 2 : Données illustratives de l'Algorithme III- 2

ENTREES			RESULTATS INTERMEDIAIRES				SORTIES		
			Itération 1		Itération 2		Itération 3		
h	k	$\{[a_{z,(h,k),j};b_{z,(h,k),j}] / 1 \le z \le y_{(h,k),j}\}$	$[a_{z,(h,k),j};$ $b_{z,(h,k),j}]$	$T_{h,j}$	$[a_{z,(h,k),j};$ $b_{z,(h,k),j}]$	$T_{h,j}$	$[a_{z,(h,k),j};$ $b_{z,(h,k),j}]$	$T_{h,j}$	$Z_{(h,k),j}$
1	3	[0;1], [2; 9], [11;+∞[	[0;1]	0	[0;1]	0	[2; 9]	2	2
2				4		7		7	
3	2 5	[0;2], [4;8], [11;+∞[[1;6], [7;+∞[	[0;2] [1;6]	1	[4;8] [1;6]	4	[4;8] [1;6]	4	2 1
4	5	[0;8], [10;+∞[	[0;8]	0	[0;8]	5	[0;8]	5	1
5	6	[0;1], [2; 9], [11;+∞[	[0;1]	2	[2;9]	1	[2;9]	1	2
6				3		6		6	

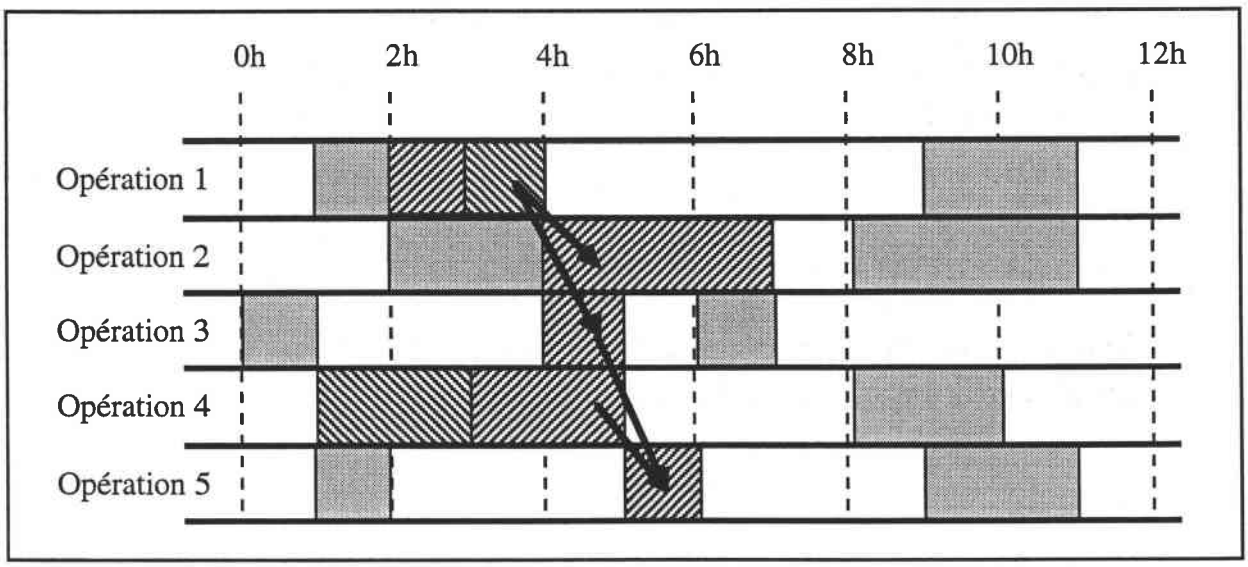

Figure III- 6 : Résultats fournis par l'Algorithme III- 2

Théorème III- 2 [Chauvet et *al.*, 1999a]

Si le Problème III- 1 a une solution réalisable, l'Algorithme III- 2 détermine les fenêtres et les dates d'exécution qui permettent de finir au plus tôt toutes les opérations effectuées sur le produit. Par conséquent, la solution fournie est optimale pour le Problème III- 1.

Démonstration

La démonstration est identique à celle du Théorème II- 2, si ce n'est qu'elle s'appuie sur le Théorème III- 1 (au lieu du Théorème II- 1).

CQFD

Complexité

La complexité de l'Algorithme III- 2 est identique à celle de l'Algorithme II- 2 et **est en** $O(y_j\ m_j + y_j \log_2 y_j)$.

III.3.5. Conclusion

Dans cette section, nous avons étendu la première méthode de gestion proposée dans le chapitre précédent aux produits dont la gamme peut comporter des opérations d'assemblage et de désassemblage. Cette méthode (i.e. l'Algorithme III- 2) consiste à définir les dates d'exécution au plus tôt des opérations compatibles avec leurs durées opératoires admissibles et les fenêtres de temps dans lesquelles les opérations doivent être exécutées. Nous minimisons ainsi la date d'achèvement du produit, i.e. le *makespan* de chaque produit. De plus, nous garantissons que tous les éléments produits, dans le cas d'un désassemblage, sont terminés le plus tôt possible. Il est possible d'utiliser la même approche pour prendre en compte des dates de disponibilité différentes pour les composants assemblés. Pour cela, nous remplaçons dans l'Algorithme III- 1 la date de disponibilité r_j par $r_{h,j}$ qui représente la date de disponibilité de l'élément du produit commençant à l'instant h.

III.4. MINIMISATION DU TEMPS TOTAL D'UTILISATION DES RESSOURCES

III.4.1. Introduction

Comme nous l'avons vu dans la section précédente, dès qu'une commande arrive, nous sommes capables de fournir un délai minimum de fabrication, et ainsi de négocier avec le client la date de livraison des éléments produits. Dans le cas d'un désassemblage, nous pouvons définir une date de livraison pour chaque élément produit. Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit au plus tard, tout en respectant la date de livraison fixée au plus tôt par l'Algorithme III- 2 (voir section précédente).

Comme dans la section 4 du chapitre précédent, nous voulons retarder le plus possible le début de fabrication du produit pour limiter le temps d'occupation des ressources. Nous espérons ainsi laisser plus de liberté pour ordonnancer les produits futurs. En outre, en minimisant le temps d'utilisation des ressources, nous réduisons le coût de fabrication du produit.

Nous formalisons ce problème avant de le résoudre par une méthode similaire à celle qui est développée dans le chapitre précédent et adaptée à la présence d'opérations d'assemblage et de désassemblage.

III.4.2. Formalisation

Nous voulons déterminer les dates de début des opérations effectuées sur le produit j sachant que la date de livraison de l'élément s'achevant à l'instant h est $d_{h,j}$ (voir contraintes (III-8)), $d_{h,j} \in \Re^+$. Cette date $d_{h,j}$ est supérieure ou égale à la date de fin au plus tôt du produit (établie dans la section précédente), sans quoi il serait impossible de fabriquer le produit à temps. Notre objectif est de maximiser la date de lancement du produit en fabrication. De plus, la solution doit vérifier les contraintes (III-2) à (III-7) introduites dans le Problème III- 1. Le problème à résoudre s'écrit donc :

Problème III- 3 : Minimisation du temps total d'utilisation des ressources

$$\max \quad \min_{h \in H_j} \{T_{h,j}\}$$

$$\text{tel que} \begin{cases} T_{h,j} \le d_{h,j} & \forall h \in H_j \quad \text{(III-8)} \\ l_{(h,k),j} \le T_{k,j} - T_{h,j} \le u_{(h,k),j} & \forall (h,k) \in I_j \quad \text{(III-2)} \\ a_{Z_{(h,k),j},(h,k),j} \le T_{h,j} & \forall (h,k) \in I_j \quad \text{(III-3)} \\ T_{k,j} \le b_{Z_{(h,k),j},(h,k),j} & \forall (h,k) \in I_j \quad \text{(III-4)} \\ 1 \le Z_{i,j} \le y_{i,j} & \forall i \in I_j \quad \text{(III-5)} \\ T_{h,j} \in \Re^+ & \forall h \in H_j \quad \text{(III-6)} \\ Z_{i,j} \in \mathrm{N} & \forall i \in I_j \quad \text{(III-7)} \end{cases}$$

Abstraction faite des contraintes (III-1) du Problème III- 1 et des contraintes (III-8) du Problème III- 3, les Problèmes III- 1 et III- 3 sont équivalents. Aussi, nous utilisons une approche analogue pour résoudre les deux problèmes. Elle est présentée dans les deux sections suivantes et consiste à résoudre une suite de sous-problèmes décrits dans le paragraphe III.4.3. Cette méthode est également une extension de celle utilisée pour donner la solution au Problème II- 3.

III.4.3. Sélection des durées opératoires

L'objectif de cette section est de déterminer les dates $T_{h,j}$ fixées au plus tôt correspondant à h, si chaque opération (h,k) est exécutée sur le produit j dans la fenêtre donnée $Z_{(h,k),j}$, i.e. :

Problème III- 4 : Définition des dates de début au plus tard des opérations quand leur fenêtre d'exécution est donnée

$$\max \quad \sum_{h \in H_j} \mu_h T_{h,j}$$

$$\text{tel que} \begin{cases} T_{h,j} \le d_{h,j} & \forall h \in H_j \quad \text{(III-8)} \\ l_{(h,k),j} \le T_{k,j} - T_{h,j} \le u_{(h,k),j} & \forall (h,k) \in I_j \quad \text{(III-2)} \\ T_{k,j} \le b_{Z_{(h,k),j},(h,k),j} & \forall (h,k) \in I_j \quad \text{(III-4)} \\ T_{h,j} \in \Re^+ & \forall h \in H_j \quad \text{(III-6)} \end{cases}$$

Nous recherchons une solution qui soit optimale pour ce problème et qui soit unique quel que soit l'ensemble $\{\mu_h \ / \ h \in H_j\}$. Cette solution correspond au cas où chaque date est calée au plus tard. Si une telle solution vérifie les contraintes (III- 3) (i.e. $b_{Z_{(h,k),j},(h,k),j} \le T_{h,j}$, pour $(h,k) \in I_j$), alors cette solution est réalisable pour le Problème III- 3 et est calée au plus tard pour cet ensemble de fenêtres $\{Z_{(h,k),j}\}_{(h,k) \in I_j}$. Sinon, aucune solution n'est réalisable en pratique pour cet ensemble de fenêtres.

L'Algorithme III- 3 permet de résoudre le Problème III- 4 par une méthode qui se déroule en deux étapes. Dans la première, on définit τ_h, borne supérieure des dates $T_{h,j}$. Dans la seconde étape, on raffine cette borne et la date obtenue est optimale.

Algorithme III- 3 : Calage au plus tard des opérations effectuées sur le produit

Procédure Définition des dates au plus tard d'exécution des opérations effectuées sur le produit j.

Entrée : j est l'indice du produit considéré.

$d_{h,j}$ est la date avant laquelle on doit livrer l'élément du produit j s'achevant à l'instant h, pour $h \in H_j$.

m_j est le nombre d'opérations à réaliser sur le produit j.

H_j est l'ensemble des instants marquant le début et la fin des opérations effectuées sur le produit j.

I_j est l'ensemble des opérations à réaliser sur le produit j.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$Z_{i,j}$ est la fenêtre dans laquelle on veut réaliser l'opération i sur le produit j, pour $i \in I_j$.

$b_{z_{i,j},i,j}$ est la date de fin de la période durant laquelle on réalise l'opération i sur le produit j, pour $i \in I_j$.

Sortie : $T_{h,j}$ est la date fixée au plus tard qui correspond à h, où $h \in H_j$.

1. **Etape 1.**

1.1. **Pour** h allant de 1 à m_j+1 : *On note h l'indice de l'instant sélectionné.*

1.1.1. **Poser** $\tau_h := \min\left\{ d_{j,h} ; \min_{\substack{(f,h)\in I_j \\ f<h}} \left\{ \tau_f + u_{(f,h),j} \right\} ; \min_{\substack{(h,k)\in I_j \\ k<h}} \left\{ \tau_k - l_{(h,k),j} \right\} ; \min_{(f,h)\in I_j} \left\{ b_{z_{(f,h),j},(f,h),j} \right\} \right\}$.

On note τ_h la borne supérieure de la date correspondant à h.

2. **Etape 2.**

2.1. **Poser** $T_{m_j+1,j} := \tau_{m_j+1}$.

2.2. **Pour** h allant de m_j à 1 : *On note h l'indice de l'instant sélectionné.*

2.2.1. **Poser** $T_{h,j} := \min\left\{ \tau_h ; \min_{\substack{(f,h)\in I_j \\ f>h}} \left\{ T_{f,j} - u_{(f,h),j} \right\} ; \min_{\substack{(h,k)\in I_j \\ k>h}} \left\{ T_{k,j} + l_{(h,k),j} \right\} \right\}$

Exemple

Nous proposons un exemple d'application de l'Algorithme III- 3. Les données relatives au produit j à réaliser sont fournies dans le Tableau III- 3 et la Figure III- 1. Dans cet exemple, nous fixons la date de livraison à la date de fin du produit au plus tôt, i.e. $d_{h,j}=d_j=7$.

Les résultats de l'exécution de l'Algorithme III- 3 sont explicités sur la Figure III- 7. Les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Tableau III- 3 : Données relatives à l'exécution de l'Algorithme III- 3

ENTREES					RESULTATS INTERME-DIAIRES	SORTIES
Indice de l'instant h	Successeurs k	Durée minimale $l_{(h,k),j}$	Durée maximale $u_{(h,k),j}$	Fin des périodes $b_{z_{(h,k),j},(h,k),j}$	Borne supérieure de la date τ_h	Date au plus tard $T_{h,j}$
1	3	1	$+\infty$	9	7	3
2					7	7
3	2 5	3 1	4 1	8 6	4	4
4	5	2	4	8	7	3
m_j=5	6	1	2	9	5	5
m_j+1=6					7	7

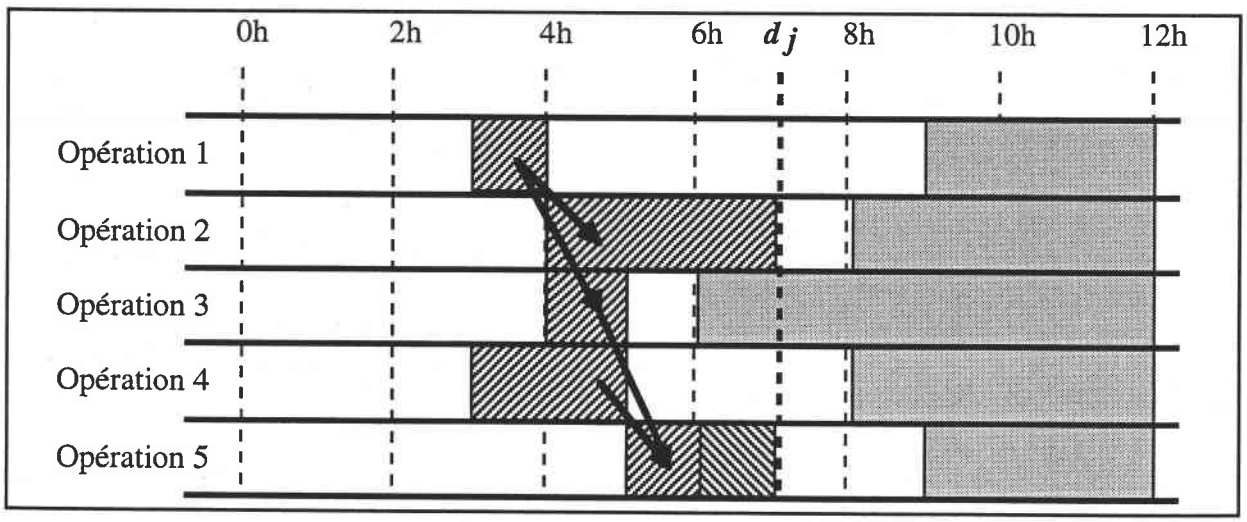

Figure III- 7 : Résultats fournis par l'Algorithme III- 3

Théorème III- 3

L'Algorithme III- 3 détermine les dates de début et de fin d'exécution au plus tard de toutes les opérations, i.e. il fournit une solution unique optimale au Problème III- 4.

Démonstration

a) Nous montrons tout d'abord que la solution $\{T_{h,j}\}_{1\le h\le m_j+1}$ fournie par l'Algorithme III- 3 est réalisable.

Pour tout $(h,k)\in I_j$., nous avons soit $h<k$, soit $h>k$.

Premier cas. Nous supposons que $k<h$. D'après la numérotation des sommets, pour tout sommet d'indice k, il ne peut exister qu'un seul sommet parmi les successeurs et prédécesseurs directs de k, ayant un indice supérieur à k. h est ce sommet et tous les autres successeurs et prédécesseurs directs de k ont un indice inférieur à k. Par conséquent, selon le pas 2.2.1 de l'algorithme appliqué à l'indice k, nous avons $T_{k,j}=\min\{\tau_k\ ;\ T_{h,j}+u_{(h,k),j}\}$. Nous en déduisons que $T_{k,j}-T_{h,j}\le u_{(h,k),j}$.

Nous pouvons aussi en déduire que $T_{k,j}-l_{(h,k),j}=\min\{\tau_k-l_{(h,k),j}\ ;\ T_{h,j}+u_{(h,k),j}-l_{(h,k),j}\}$. Cela implique, d'après le pas 1.1.1, que $T_{k,j}-l_{(h,k),j}\ge\min\{\tau_h\ ;\ T_{h,j}\}$. Comme, suivant les pas 2.1 et 2.2.1, $T_{h,j}\le\tau_h$, nous avons $T_{k,j}-l_{(h,k),j}\ge T_{h,j}$, i.e. $T_{k,j}-T_{h,j}\ge l_{(h,k),j}$.

Deuxième cas. Nous supposons que $h<k$. D'après la numérotation des sommets, k est le seul sommet parmi les successeurs et prédécesseurs directs de h ayant un indice supérieur à h. Par conséquent, selon le pas 2.2.1 de l'algorithme, nous avons $T_{h,j}=\min\{\tau_h\ ;\ T_{k,j}-l_{(h,k),j}\}$. Nous en déduisons que $T_{k,j}-T_{h,j}\ge l_{(h,k),j}$.

Nous pouvons aussi en déduire que $T_{h,j}+u_{(h,k),j}=\min\{\tau_h+u_{(h,k),j}\ ;\ T_{k,j}-l_{(h,k),j}+u_{(h,k),j}\}$. Cela implique, d'après le pas 1.1.1 appliqué à l'indice k, que $T_{h,j}+u_{(h,k),j}\ge\min\{\tau_k\ ;\ T_{k,j}\}$. Comme, suivant les pas 2.1 et 2.2.1, $\tau_k\le T_{k,j}$, nous avons $T_{h,j}+u_{(h,k),j}\ge T_{k,j}$, i.e. $T_{k,j}-T_{h,j}\le u_{(h,k),j}$.

Dans les deux cas, nous avons $l_{(h,k),j}\le T_{k,j}-T_{h,j}\le u_{(h,k),j}$. Par conséquent, la solution vérifie l'inéquation (III-2).

Suivant les pas 1.1.1, 2.1 et 2.2.1 de l'algorithme, on a $d_{h,j}\ge\tau_h\ge T_{h,j}$. Par conséquent, la solution vérifie l'inéquation (III-8).

Suivant les pas 1.1.1, 2.1 et 2.2.1 de l'algorithme, on a $b_{Z_{(h,k),j}(h,k),j}\ge\tau_h\ge T_{h,j}$, pour $(h,k)\in I_j$. Par conséquent, la solution vérifie les inéquations (III-4).

La solution fournie par l'algorithme satisfait les contraintes (III-8), (III-2), (III-4) et (III-6). Elle est réalisable.

b) Nous montrons ensuite que la solution $\{T_{h,j}\}_{1 \le h \le m_j+1}$ fournie par l'Algorithme III- 1 est optimale pour tout ensemble $\{\mu_h \ / \ 1 \le h \le m_j+1\}$.

Selon les contraintes (III-8) et (III-4), la valeur $\min\left\{d_{h,j}; \min_{\substack{(h,k)\in I_j \\ h=1}}\left\{b_{Z_{(h,k),j},(h,k),j}\right\}\right\}$ est une borne supérieure de la date correspondant à $h=1$. D'après le pas 1.1.1 de l'algorithme, τ_1 est donc une borne inférieure de cette date.

Supposons que τ_k soit une borne supérieure de la date correspondant à k pour $k<h$ et $1<h \le m_j+1$. Selon les contraintes (III-8), (III-2) et (III-4), la valeur $\min\left\{d_{j,h}; \min_{\substack{(f,h)\in I_j \\ f<h}}\left\{\tau_f - l_{(f,h),j}\right\}; \min_{\substack{(h,k)\in I_j \\ k<h}}\left\{\tau_k + u_{(h,k),j}\right\}; \min_{(f,h)\in I_j}\left\{b_{Z_{(f,h),j},(f,h),j}\right\}\right\}$ est une borne supérieure de la date correspondant à h. D'après le pas 1.2.1 de l'algorithme, τ_h est donc une borne supérieure de cette date, pour $1 \le h \le m_j+1$.

Selon le pas 2.1 de l'algorithme, $T_{m_j+1,j}$ est donc une borne supérieure de la date correspondant à m_j+1.

Supposons que T_k soit une borne supérieure de la date correspondant à k pour $k>h$ et $1<h \le m_j+1$. Selon les contraintes (III-2), puisque τ_h est donc une borne supérieure de la date correspondant à h, la valeur $\min\left\{\tau_h; \min_{\substack{(f,h)\in I_j \\ f>h}}\left\{T_{f,j} - l_{(f,h),j}\right\}; \min_{\substack{(h,k)\in I_j \\ k>h}}\left\{T_{k,j} + u_{(h,k),j}\right\}\right\}$ est aussi une borne supérieure de cette date. D'après le pas 2.2.1 de l'algorithme, T_h est donc une borne supérieure de la date correspondant à h, pour $1 \le h \le m_j+1$.

On en conclut que l'algorithme détermine les dates d'exécution au plus tard. Par conséquent, il fournit une solution unique et optimale au Problème III- 4 quel que soit l'ensemble $\{\mu_h \ / \ 1 \le h \le m_j+1\}$ de valeurs strictement positives.

CQFD

Complexité

Une analyse similaire à celle faite pour l'Algorithme III- 1, nous permet de conclure que la complexité de **l'Algorithme III- 3 est en $O(m_j)$**, comme l'Algorithme II- 3 qui résout le même problème dans le cas particulier de gammes linéaires.

III.4.4. Sélection des fenêtres

L'objectif de cette section est de résoudre le Problème III- 3. Autrement dit, nous allons déterminer dans quelle fenêtre $Z_{i,j}$, l'opération doit être effectuée de manière à ce que le produit j soit lancé le plus tard possible, tout en respectant la date de livraison fixée au plus tôt. Nous utilisons les résultats de la section précédente qui permettent, pour un ensemble de fenêtres $\{Z_{i,j}\}_{1 \le i \le m_j}$, de déterminer les dates au plus tôt $\{T_{h,j}\}_{1 \le h \le m_j+1}$ d'exécution des opérations effectuées sur le produit j, où $i=(h,k) \in I_j$. Dans le cas où ces dates ne vérifient pas les contraintes (III-3) (i.e. $b_{Z_{(h,k),j}(h,k),j} \le T_{h,j}$ pour $(h,k) \in I_j$), alors aucune solution réalisable n'existe pour ces fenêtres.

Nous utilisons l'Algorithme III- 5 pour déterminer la solution optimale au Problème III- 3. Il examine les fenêtres relatives à chaque opération dans leur ordre chronologique sans qu'aucun retour arrière ne soit nécessaire.

Algorithme III- 4 : Choix des fenêtres pour un lancement au plus tard

Il est identique à l'Algorithme II- 4, mises à part les deux modifications suivantes :

a) Au pas 2.1, nous appliquons l'Algorithme III- 3 (au lieu d'exécuter l'Algorithme II- 3).

b) Aux pas 2.2 et 2.3.1.1, la date de début de l'opération $i=(h,k)$ est $T_{h,j}$ (au lieu de $T_{i,j}$), pour tout $i \in I_j$.

Exemple

Nous proposons un exemple d'application de l'Algorithme III- 4. Les données relatives au produit j à réaliser sont fournies dans le Tableau III- 4 et dans la Figure III- 1 pour les durées opératoires. Tous les composants du produit doivent être livrés avant la date $d_{h,j}=d_j=7$.

Les résultats de l'exécution sont explicités sur la Figure III- 8 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Tableau III- 4 : Données relatives à l'exécution de l'Algorithme III- 4

ENTREES			RESULTATS INTERMEDIAIRES		SORTIES		
			Itération 1		Itération 2		
h	k	$\{[a_{z,(h,k),j};b_{z,(h,k),j}] / 1 \le z \le y_{(h,k),j}\}$	$[a_{z,(h,k),j};b_{z,(h,k),j}]$	$T_{h,j}$	$[a_{z,(h,k),j};$ $b_{z,(h,k),j}]$	$T_{h,j}$	$Z_{(h,k),j}$
1	3	[0;1], [2; 9], [11;+∞[	[11;+∞[	3	[2; 9]	2	2
2				7		7	
3	2 5	[0;2], [4;8], [11;+∞[[1;6], [7;+∞[	[11;+∞[[7;+∞[	4	[4;8] [1;6]	4	2 1
4	5	[0;8], [10;+∞[	[10;+∞[	3	[0;8]	5	1
5	6	[0;1], [2; 9], [11;+∞[	[11;+∞[	5	[2;9]	1	2
6				7		6	

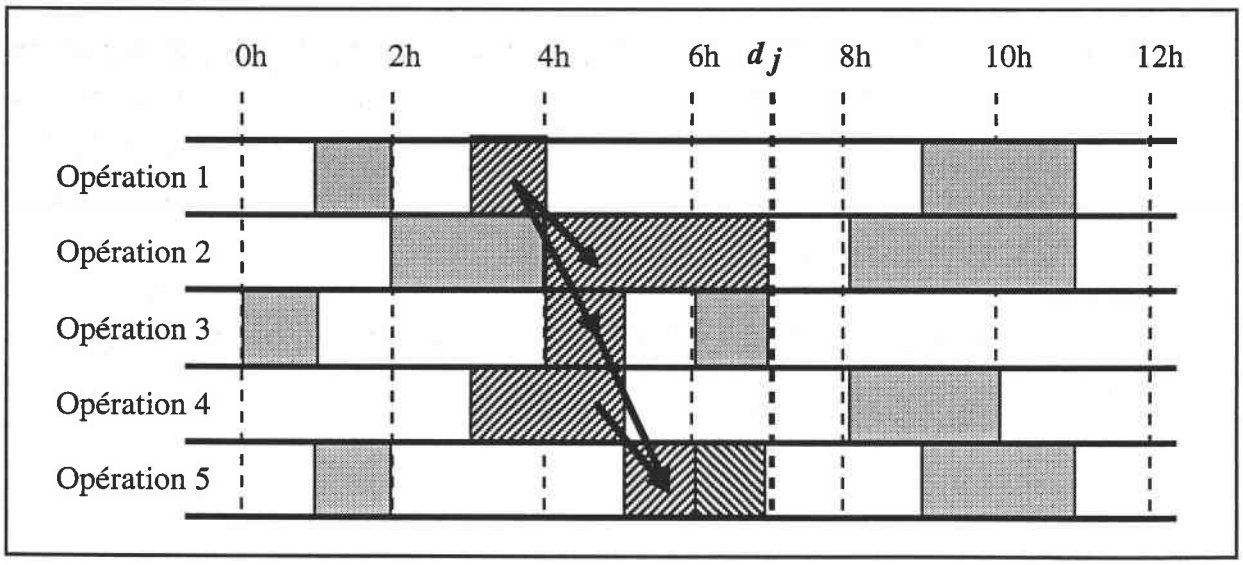

Figure III- 8 : Résultats fournis par l'Algorithme III- 4

Théorème III- 4

Si le Problème III- 3 a une solution réalisable, l'Algorithme III- 4 détermine les fenêtres et les dates d'exécution qui permettent de terminer au plus tard toutes les opérations effectuées sur le produit, tout en respectant les dates de fin $d_{h,j}$. Par conséquent, la solution fournie est optimale pour le Problème III- 3.

Démonstration

La démonstration est identique à celle du Théorème II- 4, si ce n'est qu'elle s'appuie sur le Théorème III- 3 (au lieu du Théorème II- 3).

CQFD

Complexité

La complexité de l'Algorithme III- 4 est identique à celle de l'Algorithme II- 4 et **est en** $O(y_j\ m_j + y_j \log_2 y_j)$.

III.4.5. Conclusion

Nous avons étendu le deuxième mode de gestion aux produits dont la gamme peut comporter des opérations d'assemblage et de désassemblage. Il s'agit de définir les dates d'exécution au plus tard des opérations compatibles avec leurs durées opératoires admissibles et les fenêtres de temps dans lesquelles les opérations doivent être exécutées, sans dépasser les dates de livraison fixées au plus tôt (calculées dans la section précédente). Nous maximisons ainsi la date de commencement du produit et limitons le plus possible le temps d'occupation des ressources. Nous espérons ainsi laisser plus de liberté pour ordonnancer les produits à venir.

III.5. MINIMISATION DU COUT D'UTILISATION DES RESSOURCES

III.5.1. Introduction

Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit de manière à minimiser le coût d'utilisation des ressources. Cet ordonnancement doit également respecter le délai de livraison fixé au plus tôt (voir section 3). Nous affectons aux ressources goulots un coût important afin d'inciter le système à minimiser les temps opératoires sur ces machines. Nous espérons ainsi augmenter le nombre d'opérations qui peuvent être effectuées sur ces ressources. A notre connaissance, aucune étude qui s'intéresse à ce type de problème n'existe. Nous proposons de formaliser le problème, puis de le résoudre.

III.5.2. Formalisation

Nous voulons déterminer les dates $T_{h,j}$ et $T_{k,j}$ qui définissent les instants de début et de fin de chaque opération $(h,k) \in I_j$ exécutée sur le produit j. Notre objectif est de minimiser le coût d'utilisation des ressources sachant que le coût induit en fabrication par l'opération i est $s_{i,j}$. Les contraintes similaires au Problème III- 1 et au Problème III- 3 peuvent se formaliser ainsi :

Problème III- 5 : Minimisation du coût d'utilisation des ressources

$$\min \sum_{(h,k)\in I_j} s_{(h,k),j}\left(T_{k,j} - T_{h,j}\right)$$

tel que

$$\begin{cases} r_j \le T_{h,j} & \forall h \in H_j \quad \text{(III - 1)} \\ T_{h,j} \le d_{h,j} & \forall h \in H_j \quad \text{(III - 8)} \\ l_{(h,k),j} \le T_{k,j} - T_{h,j} \le u_{(h,k),j} & \forall (h,k) \in I_j \quad \text{(III - 2)} \\ a_{Z_{(h,k),j},(h,k),j} \le T_{h,j} & \forall (h,k) \in I_j \quad \text{(III - 3)} \\ T_{k,j} \le b_{Z_{(h,k),j},(h,k),j} & \forall (h,k) \in I_j \quad \text{(III - 4)} \\ 1 \le Z_{i,j} \le y_{i,j} & \forall i \in I_j \quad \text{(III - 5)} \\ T_{h,j} \in \Re^+ & \forall h \in H_j \quad \text{(III - 6)} \\ Z_{i,j} \in \mathrm{N} & \forall i \in I_j \quad \text{(III - 7)} \end{cases}$$

Dans la section suivante, nous proposons une approche directe de résolution du problème formulé ici en un temps pseudo polynomial. Une question reste ouverte : est-ce que déterminer la solution optimale au Problème III- 5 est un problème NP-difficile ?

III.5.3. Sélection des fenêtres

L'objectif de cette section est de résoudre le Problème III- 5. Autrement dit, nous allons déterminer dans quelle fenêtre $Z_{i,j}$ et entre quelles dates $T_{h,j}$ et $T_{k,j}$ l'opération $i=(h,k)$ doit être effectuée de manière à ce que le coût d'utilisation des ressources pour fabriquer le produit j soit minimal.

L'approche proposée est une extension de l'Algorithme II- 6 du chapitre précédent et consiste en une discrétisation du temps. Nous supposons aussi que les durées des opérations ainsi que les dates de début et de fin des fenêtres sont des valeurs rationnelles. En multipliant toutes ces valeurs par le plus petit multiple commun de leur dénominateur Q, nous pouvons transformer notre problème de manière à n'avoir que des dates et des durées entières. Sans perte de généralité et pour simplifier la présentation, nous admettons donc que ces valeurs sont entières.

L'approche est basée sur la programmation dynamique. Pour cela, nous définissons (m_j+1) étapes correspondant aux (m_j+1) instants de début et de fin des opérations à réaliser sur le produit j. Les états du système à chaque étape correspondent à toutes les valeurs possibles pour la date considérée. La date $T_{h,j}$ correspond à l'instant de fin d'exécution des opérations $(f,h)\in I_j$ et de début d'exécution des opérations $(h,k)\in I_j$ sur le produit j. Puisque le produit ne peut commencer avant la date r_j et doit être terminé avant la date d_j (voir définition des contraintes (III-1) et (III-8)), nous avons :

$r_j \le T_{h,j} \le d_j$, pour $1 \le h \le m_j+1$. (III-9)

Comme les données sont entières, le nombre de dates possibles (i.e. d'états à chaque étape) est égal à $d_j - r_j + 1$.

La durée de chaque opération i est contrôlable et peut être choisie dans l'intervalle $[l_{i,j};u_{i,j}]$ (voir contrainte (III-2)). Ainsi, les dates de début de l'opération (f,h) qui autorisent une fin à la date $T_{h,j}$ sont comprises entre $T_{h,j}-u_{(f,h),j}$ et $T_{h,j}-l_{(f,h),j}$. De même, les dates de fin de l'opération (h,k) qui autorisent un début d'exécution à la date $T_{h,j}$ sont comprises entre $T_{h,j}+l_{(h,k),j}$ et $T_{h,j}+u_{(h,k),j}$.

Si nous décidons de commencer l'opération (f,h) à l'instant $T_{f,j}$ et de la terminer à l'instant $T_{h,j}$, deux cas peuvent se présenter :

a) l'intervalle $[T_{f,j}\ ;\ T_{h,j}]$ est inclus dans une période de disponibilité des ressources, $[a_{z,(f,h),j};b_{z,(f,h),j}]$ où $1 \le z \le y_{(f,h),j}$. Dans ce cas, nous associons à l'opération (f,h) qui se termine à l'instant $T_{h,j}$ un coût correspondant à l'utilisation des ressources durant cette période.

b) l'intervalle $[T_{f,j}\ ;\ T_{h,j}]$ n'est pas inclus dans une période de disponibilité des ressources ; dans ce cas, nous associons à l'opération (f,h) qui se termine à l'instant $T_{h,j}$ un coût infini.

Parmi toutes les dates possibles $T_{f,j}$, nous retenons la meilleure date (au sens du coût qu'elle induit) permettant de terminer l'opération (f,h) à la date $T_{h,j}$. Nous notons $S_{(f,h)}(T_{h,j})$ l'ensemble des dates permettant de terminer l'opération (f,h) à la date $T_{h,j}$. Formellement, nous avons :

$S_{(f,h)}(T_{h,j})=\{t \ / \ r_j \le t \le d_j,\ T_{h,j}-l_{(f,h),j} \le t \le T_{h,j}-u_{(f,h),j}$, et $\exists 1 \le z \le y_{(f,h),j}$ tq. $[t;T_{h,j}] \subseteq [a_{z,(f,h),j};b_{z,(f,h),j}]\}$
pour $r_j \le T_{h,j} \le d_j,\ 1 \le h \le m_j+1$. (III-10)

Nous désignons par $C_{(f,h)}(T_{h,j})$ le plus petit coût induit par l'opération (f,h) se terminant à la date $T_{h,j}$:

$$C_{(f,h)}(T_{h,j})=\begin{cases}+\infty & \text{si } S_{(f,h)}(T_{h,j})=\varnothing \\ \min\limits_{T_{f,j}\in S_{(f,h)}(T_{h,j})}\left\{C(T_{f,j})+s_{(f,h),j}(T_{h,j}-T_{f,j})\right\} & \text{sinon}\end{cases}$$

pour $r_j \le T_{h,j} \le d_j,\ 1 \le h \le m_j+1$. (III-11)

Dans cette formulation, $C(T_{f,j})$ désigne le coût d'utilisation des ressources utilisées par les opérations différentes de (f,h) qui se terminent ou commencent à l'instant $T_{f,j}$.

De même, si nous décidons de commencer l'opération (h,k) à l'instant $T_{h,j}$ et de la terminer à l'instant $T_{k,j}$, deux cas peuvent se présenter :

a) l'intervalle $[T_{h,j}\ ;\ T_{k,j}]$ est inclus dans une période de disponibilité des ressources, $[a_{z,(h,k),j};b_{z,(h,k),j}]$ où $1 \le z \le y_{(h,k),j}$; dans ce cas, nous associons à l'opération (h,k) qui commence à l'instant $T_{h,j}$ un coût correspondant à l'utilisation des ressources durant cette période.

b) dans l'autre cas, nous associons à (h,k) qui commence à l'instant $T_{h,j}$ un coût infini.

Nous notons $S_{(h,k)}(T_{h,j})$ l'ensemble des dates permettant de commencer l'opération (h,k) à la date $T_{h,j}$. Formellement, nous avons :

$S_{(h,k)}(T_{h,j})=\{t \ / r_j \le t \le d_j,\ T_{h,j}+l_{(h,k),j} \le t \le T_{h,j}+u_{(h,k),j}$, et $\exists 1 \le z \le y_{(h,k),j}$ tq. $[T_{h,j};t] \subseteq [a_{z,(h,k),j};b_{z,(h,k),j}]\}$
pour $r_j \le T_{h,j} \le d_j,\ 1 \le h \le m_j+1$. (III-12)

Nous désignons par $S_{(h,k)}(T_{h,j})$ le plus petit coût induit par l'opération (h,k) commençant à la date $T_{h,j}$:

$$C_{(h,k)}(T_{h,j})=\begin{cases}+\infty & \text{si } S_{(h,k)}(T_{h,j})=\varnothing \\ \min\limits_{T_{k,j}\in S_{(h,k)}(T_{h,j})}\left\{C(T_{k,j})+s_{(h,k),j}(T_{k,j}-T_{h,j})\right\} & \text{sinon}\end{cases}$$

pour $r_j \le T_{h,j} \le d_j,\ 1 \le h \le m_j+1$. (III-13)

Dans la Figure III- 9, chaque état est représenté par un disque noir. Les états d'une même étape sont alignés horizontalement. Nous donnons sur cette figure les états de trois étapes : f, h et k. Les états des étapes f et k ont été évalués avant les états de l'étape h. Ces états correspondent aux instants de début et de fin de deux opérations (f,h) et (h,k). L'ensemble des dates de l'étape f qui permettent de terminer l'opération (f,h) à la date $T_{h,j}$=7 est entouré et est noté $S_{(f,h)}(T_{h,j}=7)$. De même, l'ensemble des dates de l'étape k qui permettent de terminer l'opération (h,k) à la date $T_{h,j}$=7 est entouré et est noté $S_{(h,k)}(T_{h,j}=7)$.

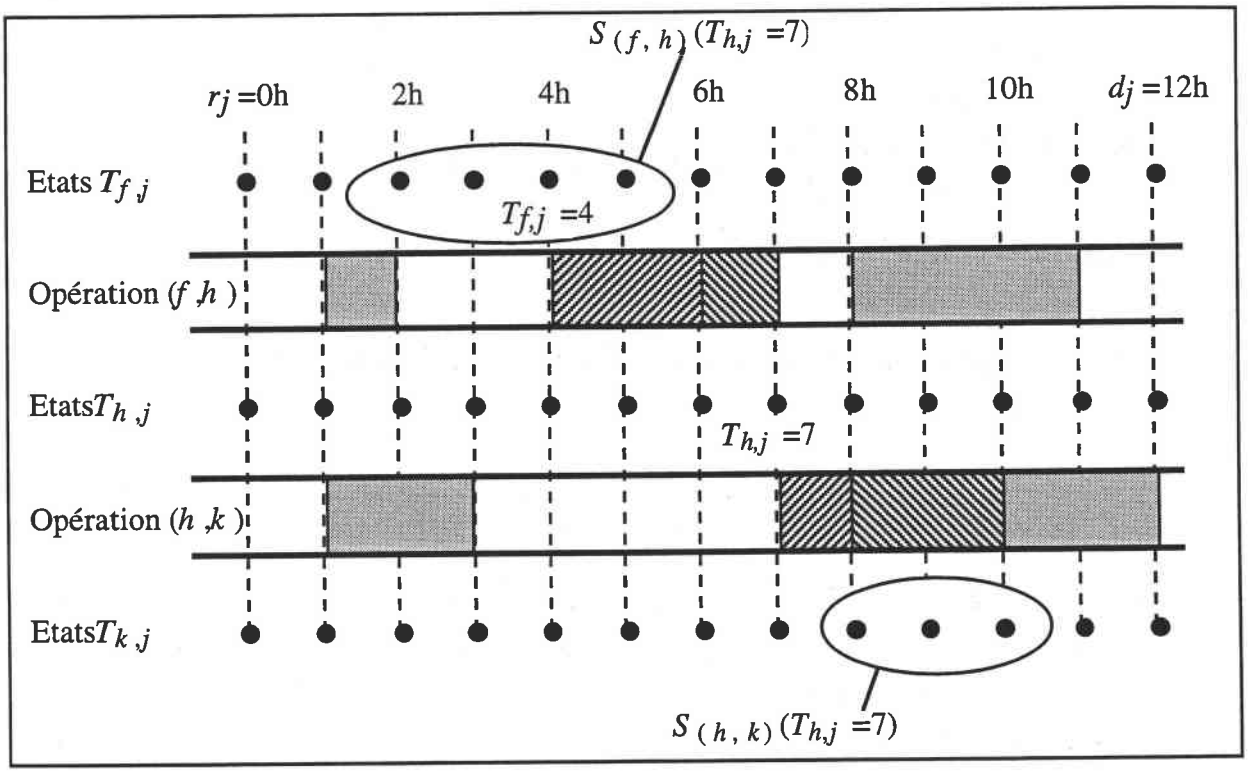

Figure III- 9 : Deux étapes f et k précédents l'étape h

Finalement, nous associons à chaque état $T_{h,j}$ un coût égal à la somme des coûts relatifs aux opérations (h,k) commençant ou (f,h) se terminant à l'instant $T_{h,j}$.

$$C(T_{h,j}) = \sum_{\substack{(h,k)\in I_j \\ k<h}} C_{(h,k)}(T_{h,j}) + \sum_{\substack{(f,h)\in I_j \\ f<h}} C_{(f,h)}(T_{h,j}) \qquad \text{(III-14)}$$

Dans cette formule, nous ne considérons que les états des étapes f et k pour lesquels le coût relatif a déjà été calculé, c'est-à-dire tels que $f<h$ et $k<h$. D'après la numérotation introduite dans le paragraphe relatif à l'Hypothèse III- 15, il n'existe qu'un seul sommet h' parmi les successeurs et prédécesseurs directs de h, ayant un indice supérieur à h. Cela signifie qu'une seule opération (h,h') commençant ou (h',h) se terminant à l'instant $T_{h,j}$ n'est pas prise en compte dans $C(T_{h,j})$. Le coût relatif à cette opération sera introduit dans le calcul de $C(T_{h,j})$ et prendra en compte $C(T_{h,j})$.

Un état $T_{h,j}$ qui ne correspond pas à une solution réalisable (i.e. $S(T_{h,j})\neq\varnothing$) a un coût infini. L'évaluation $C(T_{m_j+1,j})$ donne le coût minimal d'utilisation des ressources permettant de réaliser les opérations.

Si $\{T_{h,j}\}_{1\le h\le m_j+1}$ est la solution optimale au Problème III- 5, on a $\sum_{(h,k)\in I_j} s_{(h,k),j}\left(T_{k,j}-T_{h,j}\right)=\min_{r_j\le T_{m_j+1,j}\le d_j}\left\{C(T_{m_j+1,j})\right\}$. L'Algorithme III- 5 découle des relations (III-9) à (III-14) et permet de résoudre le Problème III- 5.

Algorithme III- 5 : Choix des fenêtres pour un coût d'utilisation minimal

Procédure Définition des fenêtres induisant un coût d'utilisation des ressources minimal pour le produit j.

Entrée : j est l'indice du produit considéré.

r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.

$d_{h,j}$ est la date avant laquelle on doit livrer l'élément du produit j s'achevant à l'instant h, pour $h\in H_j$.

m_j est le nombre d'opérations à réaliser sur le produit j.

H_j est l'ensemble des instants marquant le début et la fin des opérations effectuées sur le produit j.

I_j est l'ensemble des opérations à réaliser sur le produit j.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i\in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i\in I_j$.

$y_{i,j}$ est le nombre de fenêtres dans laquelle on peut réaliser l'opération i sur le produit j, pour $i\in I_j$.

$a_{z,i,j}$ est la date de début de la z -ième fenêtre de l'opération i effectuée sur le produit j, pour $1\le z\le y_{i,j}$, $i\in I_j$.

$b_{z,i,j}$ est la date de fin de la z -ième fenêtre de l'opération i effectuée sur le produit j, pour $1\le z\le y_{i,j}$, $i\in I_j$.

Sortie : $T_{h,j}$ est la date correspondant à h, où $h\in H_j$.

$Z_{i,j}$ est la fenêtre dans laquelle on doit réaliser l'opération i sur le produit j, où $i\in I_j$.

1. **Initialisation.**

1.1. **Pour** $(f,h)\in I_j$: *On examine chaque opération (f,h) une par une.*

1.1.1. **Si** ($f<h$) : *Les états de l'étape f sont examinées avant ceux de l'étape h.*

1.1.1.1. **Trier** les fenêtres $[a_{z,(f,h),j}\ ;\ b_{z,(f,h),j}]$ pour $1\le z\le y_{(f,h),j}$ dans l'ordre croissant des $a_{z,(f,h),j}$ et, en cas d'égalité, dans l'ordre croissant des $b_{z,(f,h),j}$. *Dans la suite, la z -ième fenêtre est le z -ième élément dans cet ordre.* $1\le z\le y_{z,(f,h),j}$

1.1.2. **Sinon** :

1.1.2.1. **Trier** les fenêtres $[a_{z,(f,h),j}\ ;\ b_{z,(f,h),j}]$ pour $1\le z\le y_{(f,h),j}$ dans l'ordre croissant des $b_{z,(f,h),j}$ et, en cas d'égalité, dans l'ordre croissant des $a_{z,(f,h),j}$. *Dans la suite, la z -ième fenêtre est le z -ième élément dans cet ordre.* $1\le z\le y_{z,(f,h),j}$

2. **Pour** h allant de 1 à m_j+1 : *On note h l'indice de l'étape examinée.*

2.1. **Pour** $T_{h,j}$ allant de r_j à $d_{h,j}$: *On énumère chaque état $T_{h,j}$ de l'étape h.*

2.1.1. **Pour** $C(T_{h,j}):=0$. *On initialise le coût de l'état $T_{h,j}$ à 0.*

2.1.2. **Pour** $f\in\{f<h\ /\ (f,h)\in I_j\}$: *On note f l'indice de l'étape précédente.*

2.1.2.1. **Poser** $C_{(f,h)}(T_{h,j}):=+\infty$. *On initialise le coût relatif à l'opération (f,h) à $+\infty$.*

2.1.2.2. **Poser** $z:=\min\{z\ /\ 1\le z\le y_{(f,h),j},\ b_{z,(f,h),j}\ge T_{h,j}\}$. *On note z la $1^{ère}$ fenêtre permettant de réaliser l'opération (f,h) qui se termine à la date $T_{h,j}$.*

2.1.2.3. **Pour** $T_{f,j}$ allant de $\max\{r_j\ ;\ a_{z,(f,h),j}\ ;\ T_{h,j}-u_{(f,h),j}\}$ à $\min\{d_{h,j}\ ;\ T_{h,j}-l_{(f,h),j}\}$: *On examine chaque état $T_{f,j}$ permettant d'atteindre l'état $T_{h,j}$.*

2.1.2.4. **Si** ($C_{(f,h)}(T_{h,j})>C(T_{f,j})+(T_{h,j}-T_{f,j})\ s_{(f,h),j}$) **alors** : *L'état $T_{f,j}$ permet de diminuer le coût.*

2.1.2.4.1. **Poser** $C_{(f,h)}(T_{h,j}):=C(T_{f,j})+(T_{h,j}-T_{f,j})\,s_{(f,h),j}$

2.1.2.4.2. **Poser** $Sopt_{(f,h)}(T_{h,j}):=T_{f,j}$. *On désigne par* $Sopt_{(f,h)}(T_{h,j})$ *la date permettant de terminer l'opération* (f,h) *à la date* $T_{h,j}$ *au coût le plus petit.*

2.1.2.5. **Poser** $C(T_{h,j}):=C(T_{h,j})+C_{(f,h)}(T_{h,j})$. *On additionne au coût de l'état le coût relatif à l'état* $T_{h,j}$.

2.1.3. **Pour** $k\in\{k<h \ / \ (h,k)\in I_j\}$: *On note* k *l'indice de l'étape suivante.*

2.1.3.1. **Poser** $C_{(h,k)}(T_{h,j}):=+\infty$. *On initialise le coût relatif à l'opération* (h,k) *à* $+\infty$.

2.1.3.2. **Poser** $z:=\max\{z \ / \ 1\le z\le y_{(h,k),j},\ a_{z,(h,k),j}\le T_{h,j}\}$. *On note* z *la* $1^{ère}$ *fenêtre permettant de réaliser l'opération* (h,k) *qui commence à la date* $T_{h,j}$.

2.1.3.3. **Pour** $T_{k,j}$ allant de $\max\{r_j\ ;\ T_{h,j}+l_{(f,h),j}\}$ à $\min\{d_{k,j}\ ;\ b_{z,(f,h),j}\ ;T_{h,j}+u_{(f,h),j}\}$: *On examine chaque état* $T_{k,j}$ *permettant d'atteindre l'état* $T_{h,j}$.

2.1.3.4. **Si** ($C_{(h,k)}(T_{h,j})>C(T_{k,j})+(T_{k,j}-T_{h,j})\,s_{(h,k),j}$) **alors** : *L'état* $T_{k,j}$ *permet de diminuer le coût.*

2.1.3.4.1. **Poser** $C_{(h,k)}(T_{h,j}):=C(T_{k,j})+(T_{k,j}-T_{h,j})\,s_{(h,k),j}$.

2.1.3.4.2. **Poser** $Sopt_{(h,k)}(T_{h,j}):=T_{k,j}$. *On désigne par* $Sopt_{(h,k)}(T_{h,j})$ *la date permettant de commencer l'opération* (h,k) *à la date* $T_{h,j}$ *au coût le plus bas.*

2.1.3.5. **Poser** $C(T_{h,j}):=C(T_{h,j})+C_{(h,k)}(T_{h,j})$. *On additionne au coût de l'état le coût relatif à l'état* $T_{h,j}$.

3. **Poser** $Copt:=+\infty$. *On note* $Copt$ *le meilleur coût trouvé pour un état* $T_{m_j+1,j}$ *de la dernière étape.*

4. **Pour** $T_{m_j+1,j}$ allant de r_j à $d_{m_j+1,j}$: *On énumère chaque état* $T_{m_j+1,j}$.

4.1. **Si** ($Copt>C(T_{m_j+1,j})$) **alors** : *L'état* $T_{m_j+1,j}$ *à un meilleur coût.*

4.1.1. **Poser** $Copt:=C(T_{m_j+1,j})$.

4.1.2. **Poser** $Topt:=T_{m_j+1,j}$. *On note par* $Topt$ *la date induisant le coût le plus bas.*

5. **Reconstitution de la solution.**

5.1. **Si** ($Copt=+\infty$) **alors** : *Aucun état n'est atteignable.*

5.1.1. **Quitter.** *Il n'existe aucune solution réalisable.*

5.2. **Sinon** :

5.2.1. **Poser** $T_{m_j+1,j}:=Topt$.

5.2.2. **Pour** h allant de m_j+1 à 2 : *On note* h *l'indice de l'étape considérée.*

5.2.2.1. **Pour** $f\in\{f<h \ / \ (f,h)\in I_j\}$: *On note* f *l'indice de l'étape considérée.*

5.2.2.1.1. **Poser** $T_{f,j}:=Sopt_{(f,h)}(T_{h,j})$.

5.2.2.1.2. **Poser** $Z_{(f,h),j}:=\min\{z \ / \ 1\le z\le y_{(f,h),j},\ b_{z,(f,h),j}\ge T_{h,j}\}$.

5.2.2.2. **Pour** $k\in\{k<h \ / \ (h,k)\in I_j\}$: *On note* k *l'indice de l'étape considérée.*

5.2.2.2.1. **Poser** $T_{k,j}:=Sopt_{(h,k)}(T_{h,j})$.

5.2.2.2.2. **Poser** $Z_{(h,k),j}:=\max\{z \ / \ 1\le z\le y_{(h,k),j},\ a_{z,(h,k),j}\le T_{h,j}\}$.

Si l'Algorithme III- 5 s'achève au pas 5.1.1, il n'existe aucune solution réalisable. Cela signifie qu'il est impossible de fabriquer le produit avec les contraintes imposées au système de production.

Exemple

Nous proposons un exemple d'application de l'Algorithme III- 5. Les données relatives au produit j à réaliser sont fournies dans le Tableau III- 5. Le produit peut être réalisé à partir de la date $r_j=0$ et avant la date $d_{h,j}=7$.

Tableau III- 5 : Données fournies à l'Algorithme III- 5

ENTREES					
Indice de l'opération i	Durée minimale $l_{i,j}$	Durée maximale $u_{i,j}$	Coût opératoire $s_{i,j}$	Instant de début et de fin (h,k)	Périodes durant lesquelles on peut réaliser l'opération i $\{[a_{z,i,j};b_{z,i,j}] / 1 \le z \le y_{i,j}\}$
1	1	+∞	3	(1,3)	[0;1], [2;9], [11;+∞[
2	3	4	2	(3,2)	[0;2], [4;8], [11;+∞[
3	1	1	3	(3,5)	[1;6], [7;+∞[
4	2	4	1	(4,5)	[0;8], [10;+∞[
m_j=5	1	+∞	3	(5,6)	[0;1], [2;9], [11;+∞[

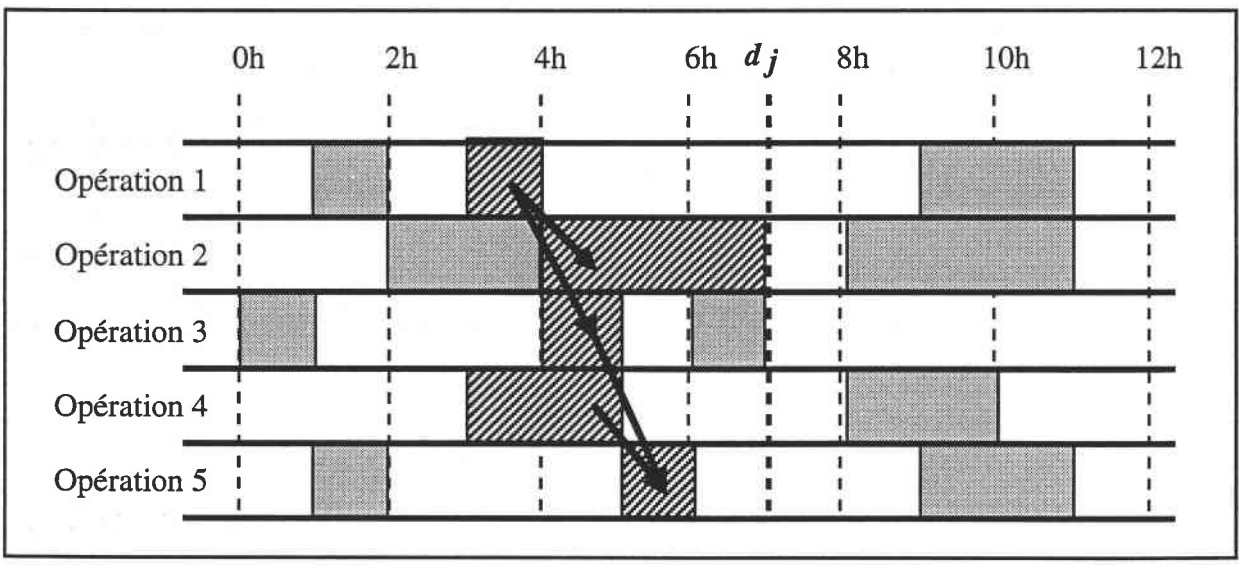

Figure III- 10 : Résultats fournis par l'Algorithme III- 5

Les résultats de l'exécution sont donnés dans la Figure III- 10 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Tableau III- 6 : Résultats de l'exécution de l'Algorithme III- 5

ENTREE	RESULTATS INTERMEDIAIRES												SORTIE
	$T_{1,j}$	$T_{2,j}$	Etat $T_{3,j}$			$T_{4,j}$	Etat $T_{5,j}$			Etat $T_{6,j}$			
T	C	C	k / $(h,k)\in I_j$ ou $(k,h)\in I_j$	$Sopt$	C	C	k / $(h,k)\in I_j$ ou $(k,h)\in I_j$	$Sopt$	C	k / $(h,k)\in I_j$ ou $(k,h)\in I_j$	$Sopt$	C	$T_{h,j}$
r_j=0	0	0			+∞	0			+∞			+∞	
1	0	0			+∞	0			+∞			+∞	
2	0	0			+∞	0			+∞			+∞	
3	**0**	0			+∞	**0**			+∞			+∞	$T_{1,j}$ $T_{4,j}$
4	0	0	1 2	**3** **7**	**9**	0			+∞			+∞	$T_{3,j}$
5	0	0	1 2	4 8	9	0	3 4	**4** **3**	**14**			+∞	$T_{5,j}$
6	0	0			+∞	0	3 4	5 4	14	5	**5**	**17**	$T_{6,j}$
$d_{h,j}$=7	0	**0**			+∞	0			+∞	5	6	20	$T_{2,j}$

Théorème III- 5

Si le Problème III- 5 a une solution réalisable, l'Algorithme III- 5 détermine les fenêtres et les dates d'exécution des opérations qui permettent de réaliser le produit avec un coût minimal, i.e. la solution fournie par l'Algorithme III- 5 est optimale pour le Problème III- 5.

Démonstration

La formulation du Problème III- 5 est équivalente aux équations de programmation dynamique (III-9) à (III-14). Il en découle que l'Algorithme III- 5 fournit la solution optimale au Problème III- 5.

CQFD

Complexité

Nous appelons $d_j = \max\{d_{h,j} \ / \ h \in I_j\}$ la date à laquelle tous les composants produits de j doivent être livrés. Le principe de l'Algorithme III- 5 étant similaire à l'Algorithme II- 6, **leur complexité est identique en** $O(y_j \log_2 y_j + y_j \ m_j(d_j - r_j + 1) + m_j(d_j - r_j + 1)^2)$. Les remarques proposées pour réduire le temps de calcul de l'Algorithme II- 6 sont également applicables à l'Algorithme III- 5.

Place mémoire

Contrairement à l'Algorithme II- 6, nous devons conserver les évaluations des états (i.e. des dates) de toutes les étapes précédentes. **Par conséquent, l'Algorithme III- 5 requiert** $(2m_j)(d_j - r_j + 1)$ **mots.**

III.5.4. Conclusion

Dans cette section, la méthode de gestion en temps réel décrite ordonnance chaque produit de manière à minimiser le coût d'utilisation des ressources, ceci sans dépasser la date de livraison des produits fixée au plus tôt. Cette méthode de gestion basée sur la programmation dynamique permet de fabriquer chaque produit au coût le plus bas en tenant compte des produits déjà ordonnancés.

En outre, nous pouvons utiliser cette méthode pour augmenter la productivité du système. Pour cela, nous affectons aux machines pour lesquelles la charge minimale de travail (voir définition Section II.6.2.2) est la plus grande, un coût d'utilisation élevé. Ceci permet de limiter le plus possible l'utilisation de ces machines et ainsi espérer accroître la productivité du système.

III.6. APPLICATIONS

III.6.1. Introduction

Dans cette section, nous proposons d'évaluer les conséquences de la limitation du temps de stockage sur la productivité d'un système de production. Pour cela, nous mesurons la productivité d'un des modes de gestion en temps réel suivant le temps de stockage autorisé.

Nous ne cherchons pas dans cette section à comparer les trois modes de gestion, mais à présenter une utilisation spécifique de ces méthodes. Aussi, puisque les trois méthodes donnent des résultats similaires, nous avons choisi arbitrairement d'appliquer la première méthode (i.e. l'Algorithme III- 2). Cette méthode de gestion donne l'ordonnancement de chaque produit au moment où sa commande est enregistrée. Elle consiste à fabriquer le produit au plus tôt (voir section 3).

Dans la section suivante, nous décrivons le système de production considéré. Puis, nous donnons les résultats fournis par cette étude.

III.6.2. Description du système considéré

Le système que nous considérons dans cette section est un système d'assemblage composé de K niveaux. Un produit qui a subi une opération au niveau k est immédiatement transféré au niveau $k+1$, pour $k=1, \ldots, K-1$. Le niveau K est le dernier niveau : un produit qui a subi une opération au niveau K est terminé et peut-être livré.

Chaque niveau k du système de production est composé de E_k pôles, pour $k=1, \ldots, K$. Chaque pôle e du niveau k regroupe $S_{e,k}$ places de stockage des en-cours de fabrication et $R_{e,k}$ machines, pour $e=1, \ldots, E_k$ et $k=1, \ldots, K$. Les machines d'un même pôle sont identiques et peuvent être utilisées indifféremment pour réaliser une même opération. Chacune des $S_{e,k}$ places permet de stocker une unité de produit avant que soit exécutée l'opération sur une des machines $R_{e,k}$. Si l'opération exécutée sur l'une des machines $R_{e,k}$ est un assemblage, chacune des $S_{e,k}$ places permet de stocker tous les composants entrant dans la fabrication d'une unité de produit résultant de cet assemblage. La Figure III- 11 donne un exemple d'un tel système d'assemblage.

Dans les exemples numériques présentés ci-après (qui comportent plus de niveaux et de pôles que sur la Figure III- 11), le nombre de niveaux est $K=6$. Le nombre de pôles par niveau sont donnés par le Tableau III- 7. Chaque pôle est composé de $R_{e,k}=3$ machines identiques et de $S_{e,k}=4$ places de stockage.

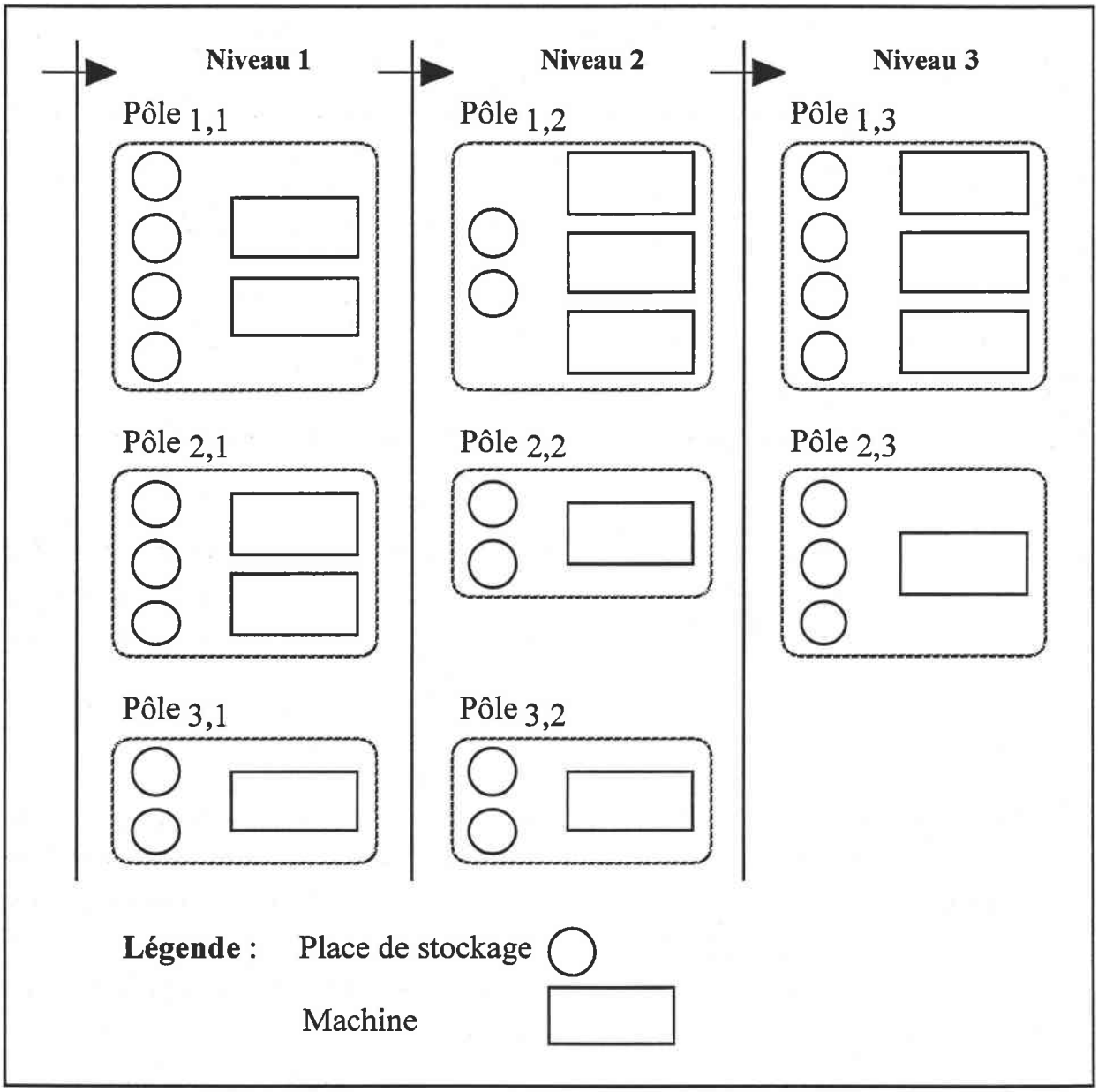

Figure III- 11 : Représentation d'un système d'assemblage

Tableau III- 7 : Nombre de pôles par niveau

Niveau	1	2	3	4	5	K =6
Nombre de pôles, E_k	6	2	5	9	5	2

Les produits fabriqués par ce système de production nécessitent des opérations d'assemblage. Les composants assemblés au niveau k proviennent chacun d'un pôle du niveau k -1, comme représenté sur la Figure III- 12. Les composants du produit sont stockés sur le pôle dans la limite des places disponibles, avant d'être assemblés sur une des machines du pôle. Dès que

l'opération d'assemblage est réalisée, le produit est immédiatement transporté sur un pôle du niveau supérieur.

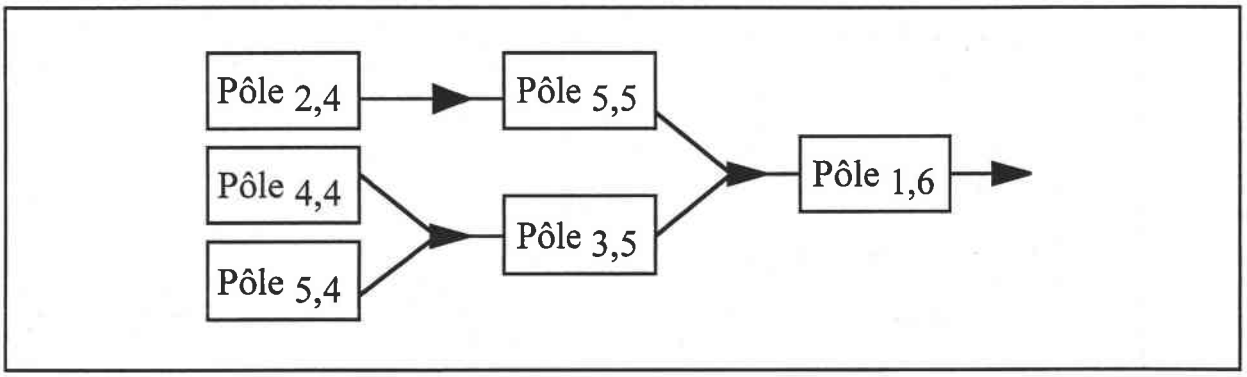

Figure III- 12 : Représentation d'un produit assemblé

Nous voulons étudier le comportement du système de production géré en temps réel par un mode de gestion qui ordonnance tous les produits au plus tôt (i.e. voir Algorithme III- 2). Pour cela, nous générons 1000 produits qui doivent être fabriqués sur le système que nous avons décrit. Par conséquent, chaque produit ne comporte pas plus de 6 niveaux d'opérations successifs. Le nombre d'opérations réalisées et l'enchaînement de ces opérations sont déterminés aléatoirement.

Les durées des opérations réalisées sur les machines sont tirées au hasard entre 0 et 10. Nous interdisons qu'un produit reste sur une machine après la réalisation de son opération. Par conséquent, les durées opératoires ne sont pas contrôlables. Le stockage entre deux opérations est facultatif (i.e. de durée minimale nulle) et est limité en durée à U_S qui est un paramètre de contrôle de notre système. Nous voulons évaluer le comportement du système de production suivant les valeurs de ce paramètre de contrôle.

III.6.3. Résultats numériques

Dans notre système de production, les périodes de stockage sont considérées comme des opérations, et les places de stockage comme des ressources disponibles en nombre limité. Les algorithmes présentés dans les sections précédentes permettent de contrôler la durée de ces périodes. Nous évaluons pour des différentes durées maximales de stockage $U_S \in [0\ ;\ 10]$, les effets sur le système de production :

a) Le taux d'utilisation moyen des places de stockage, c'est-à-dire la moyenne du rapport entre le temps d'occupation de chaque place de stockage et l'horizon de travail,

b) Le taux d'utilisation de la machine goulot, c'est-à-dire le taux d'utilisation de la machine la plus utilisée,

c) La moyenne du temps de cycle de chaque produit.

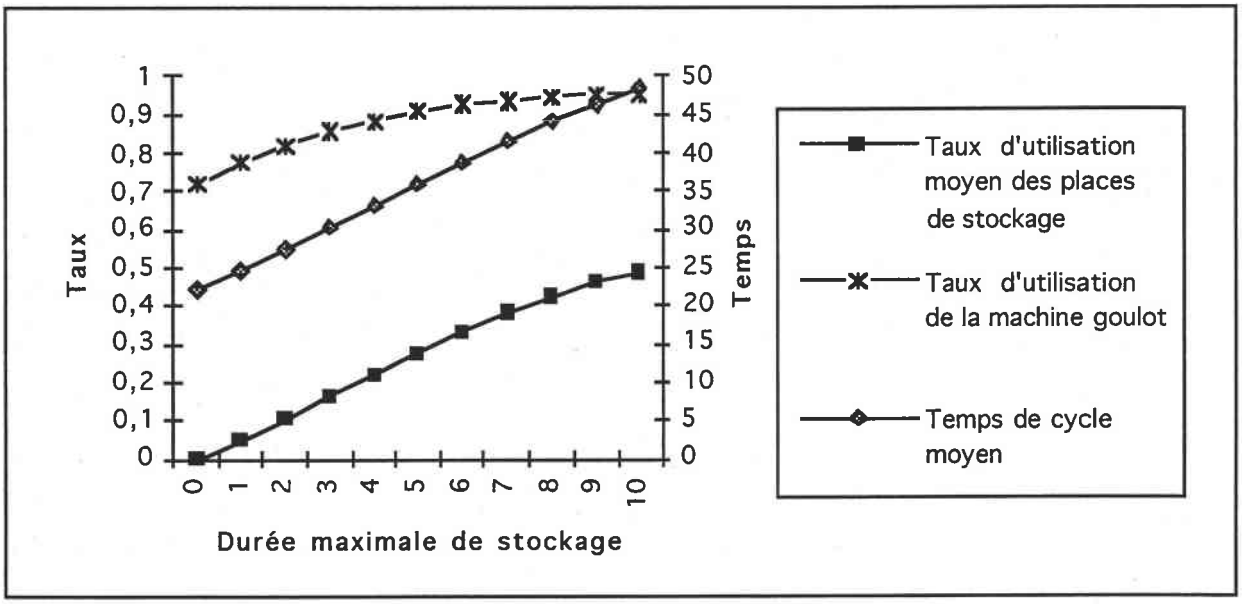

Figure III- 13 : Résultats numériques

La Figure III- 13 présente les effets de la limitation de la durée de stockage sur le système de production. En toute logique, l'utilisation des places de stockage augmente si nous permettons aux produits d'être stockés entre les opérations. Augmentant la durée de stockage en cours de production, le temps de cycle de chaque produit est allongé.

Dans notre cas, le taux d'utilisation de la machine goulot mesure la productivité du système. Nous constatons que moins la durée de stockage est contrainte, plus la productivité augmente. Mais, alors que la productivité peut être augmentée de manière significative (de l'ordre de 5 % par unité de temps) pour des durées de stockage faibles (i.e. U_S proche de 0), elle devient peu sensible (de l'ordre de 0,5 % par unité de temps) pour des valeurs importantes (i.e. U_S proche de 10).

III.6.4. Conclusion

Dans cette section, nous avons évalué les conséquences de la limitation du temps de stockage sur la productivité d'un système de production. Plus nous limitons le temps de stockage, plus le système est contraint et moins la productivité est importante. Ceci illustre l'importance des effets des contraintes de type "*no-wait*" sur les systèmes de production.

De plus, cet exemple montre comment le stockage limité (en nombre de places et en durée) est pris en compte par les méthodes de gestion proposées dans les sections précédentes.

III.7. CONCLUSION

III.7.1. Approche suivie

Dans ce chapitre, nous avons étendu les méthodes de gestion de production en temps réel proposées dans le chapitre précédent. Ces méthodes ont été adaptées aux situations où des opérations d'assemblage et de désassemblage doivent être réalisées sur des produits. Elles restent applicables pour les problèmes à durées opératoires mixtes. **L'objectif est d'obtenir une productivité maximale à long terme.** Trois méthodes sont proposées :

- La première méthode de gestion consiste **à terminer chaque produit le plus tôt possible** (voir Algorithme III- 2).
- La deuxième méthode proposée revient **à fabriquer le produit au plus tard,** tout en respectant la date de livraison fixée au plus tôt (voir Algorithme III- 4). L'intérêt est de réduire le plus possible le temps pendant lequel des ressources sont utilisées pour fabriquer le produit.
- Nous proposons une troisième méthode pour laquelle chaque produit est fabriqué de manière à **minimiser le coût d'utilisation des ressources de chaque produit,** sans dépasser la date de livraison fixée au plus tôt (voir Algorithme III- 5). Nous utilisons également cet algorithme pour inciter le système à utiliser le moins possible les ressources-goulots et, ainsi, espérer accroître la productivité du système.

Les algorithmes présentés ici sont nouveaux et leur complexité très réduite permet leur utilisation en temps réel. Une partie des travaux de ce chapitre ont été présentés [Chauvet et Proth, 1998a] [Chauvet et Proth, 1998b] [Chauvet et *al.*, 1998b].

Dans le chapitre suivant, nous élargissons notre étude aux situations où plusieurs gammes alternatives peuvent être choisies pour réaliser un produit.

III.7.2. Extension : Autres gammes et conflits de ressources

III.7.2.1. Introduction

Dans cette section, nous montrons comment étendre l'étude précédente, quand cela est possible, aux situations où il existe des cycles dans le graphe représentant la gamme du produit (voir l'Hypothèse III- 15), où le ré-ordonnancement est possible (voir l'Hypothèse III- 3), où il existe des conflits de ressources (voir l'Hypothèse III- 16) et où la préemption est permise (voir l'Hypothèse III- 13). Nous montrons également comment tirer le meilleur parti possible de notre étude dans ces différents cas.

III.7.2.2. Présence de cycle

Dans ce qui précède, nous avons supposé que deux opérations ne peuvent avoir à la fois une même opération qui les précède, et une même opération qui les suit (voir l'Hypothèse III- 15). En fait, nous avons admis que le graphe non-orienté associé $G_j=(H_j,I_j)$ n'admet aucun cycle.

Mais, dans certaines applications, il arrive que des opérations doivent commencer et se terminer en même temps. Ces contraintes introduisent des cycles dans notre graphe non-orienté associé à G_j et interdisent l'utilisation des Algorithmes III- 1, III- 3 et III- 5. Dans ces cas, nous proposons une approche certes moins rapide, mais qui permet néanmoins de fournir une solution optimale aux Problèmes III- 2 et III- 4.

Pour cela, nous représentons les contraintes de ces problèmes sous forme d'un graphe "potentiels étapes" [Roy, 1960], ou d'un graphe PERT [Malcom et *al.*, 1959], comme nous l'avons fait au chapitre précédent pour le Problème II- 3 (voir Figure II- 8). Dans un graphe PERT, la valuation $\theta_{(h,h')}$ de chaque arc (h,h') donne la durée minimale qui séparant les dates T_h et $T_{h'}$ correspondant aux instants h et h' (i.e. $T_{h'}-T_h \geq \theta_{(h,h')}$). Dans le cas où $\theta_{(h,h')} \leq 0$, $-\theta_{(h,h')}$ représente la durée maximale séparant les dates $T_{h'}$ et T_h (i.e. $T_h-T_{h'} \leq -\theta_{(h,h')}$). Une telle représentation permet de modéliser toutes les contraintes des Problèmes III- 2 et III- 4, comme nous le présentons sur la Figure III- 14.

Les algorithmes de plus long chemin appliqués au graphe PERT [Gondran et Minoux, 1979] permettent alors de déterminer une solution optimale des Problèmes III- 2 et III- 4. Cependant, la présence de circuits dans le graphe PERT impose l'utilisation d'algorithmes spécifiques [Ford, 1956] [Bellman, 1958] dont la complexité en $O(m_j^2)$ est moins bonne que celle des Algorithmes III- 1 et III- 3 en $O(m_j)$.

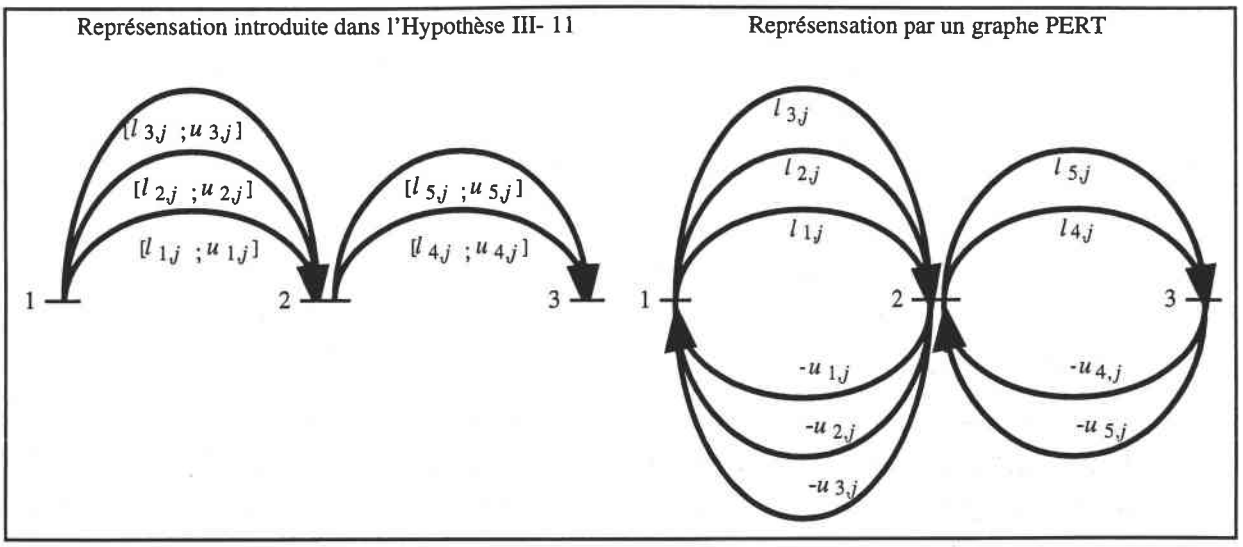

Figure III- 14 : Représentations équivalentes en présence de cycle

Nous pouvons aussi donner la solution optimale aux Problèmes III- 1 et III- 3 en présence de cycles. Pour cela, nous utilisons respectivement les Algorithmes III- 2 et III- 4 pour lesquels nous appliquons, au pas 2.1 de ces algorithmes, une méthode de recherche de plus long chemin dans le graphe PERT associé. La complexité de ces deux algorithmes est en $O(y_j\ m_j^2 + y_j \log_2 y_j)$.

Des études sont menées actuellement pour résoudre de manière optimale le Problème III- 5 en présence de cycle.

III.7.2.3. Cas où la préemption est permise

Dans cette étude, nous avons supposé que les opérations ne pouvaient être interrompues pendant un laps de temps puis reprises (voir Hypothèse III- 13). Nous proposons dans cette section une adaptation qui permet de limiter le nombre d'interruption d'une opération.

Supposons qu'une opération i exécutée sur le produit j puisse être morcelée en $f_{i,j}$ exécutions successives. Sa durée opératoire doit être comprise dans $[l_{i,j}\ ;\ u_{i,j}]$. De plus, cette opération peut être exécutée dans l'une des périodes de $W_{i,j}$.

Nous créons $f_{i,j}$ opérations correspondant chacune à un des morcellements dont la durée est comprise dans $[0\ ;\ u_{i,j}]$. Ces $f_{i,j}$ opérations doivent être exécutées successivement. La première de ces $f_{i,j}$ opérations doit débuter simultanément à l'opération i et la dernière doit s'arrêter au même moment que l'opération i. La Figure III- 15 représente l'opération i et l'enchaînement des $f_{i,j}$ morcellements. La durée totale de ces $f_{i,j}$ morcellements est égale à celle de l'opération i qui est comprise entre $[l_{i,j}\ ;\ u_{i,j}]$.

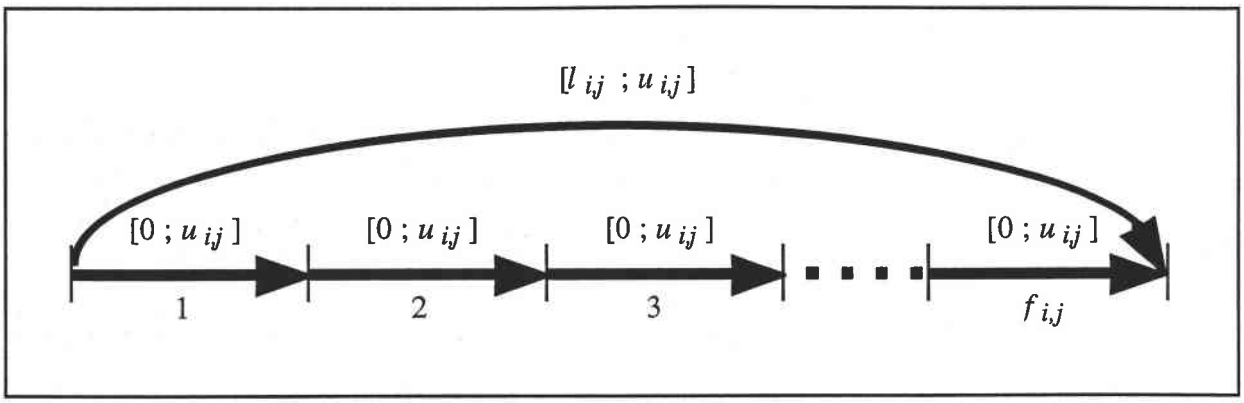

Figure III- 15 : Morcellement d'une opération

Les ressources requises initialement pour exécuter l'opération i sont affectées à chacun des $f_{i,j}$ morcellements. Ainsi, chacune de ces $f_{i,j}$ opérations doit être exécutée dans l'une des périodes de $W_{i,j}$. Quant à l'opération i, elle peut être exécutée à tout moment dans l'intervale [0 ; +∞[.

Une telle modélisation introduit un cycle sur notre graphe. Par conséquent, nous résolvons les Problèmes III- 1 à III- 4 en suivant l'approches décrite dans la section précédente.

III.7.2.4. Cas avec conflits de ressources

Jusqu'ici nous avons supposé qu'il n'existait pas de conflits de ressources entre les opérations pouvant être effectuées en parallèle sur un même produit. Deux conditions pour lesquelles cette hypothèse est vérifiée sont proposées (voir l'Hypothèse III- 16).

Ces conditions garantissent que les méthodes décrites dans les sections précédentes fournissent des ordonnancements pour lesquels les opérations utilisant les mêmes ressources ne sont pas exécutées simultanément. Si aucune de ces conditions n'est vérifiée, les Algorithmes III- 1 à III- 5 peuvent malgré tout être appliqués. Si l'ordonnancement qu'ils fournissent ne génère pas de conflit de ressources, alors il est optimal. Sinon, il donne une borne inférieure de la solution optimale, l'ordonnancement est qualifié de sous-optimal. Nous devons alors lever le conflit et, pour cela, déterminer quelles opérations en conflit doivent être exécutées avant les autres.

En présence de conflit, déterminer la solution optimale d'un des **Problèmes III- 1 à III- 5 devient combinatoire.**

Théorème III- 6

Les Problèmes III- 1 à III- 5 sont NP-difficiles au sens fort (en présence de conflits de ressources).

Démonstration

a) Montrons que minimiser le *makespan* d'un produit dont les opérations se suivent sans attente et ont des durées contrôlables est NP-difficile au sens fort (même si les fenêtres dans lesquelles sont exécutées les opérations sont données)

Pour tout ordonnancement d'un tel produit, nous pouvons vérifier en un temps polynomial qu'il satisfait toutes les contraintes, et déterminer son *makespan.*

Considérons un *flow shop* à 3 machines du type $F_3 |nwt|C_{max}$. Minimiser le *makespan* d'un tel problème sous la contrainte qu'il n'y ait pas d'attente est NP-difficile au sens fort [Röck, 1984a]. Dans ce problème, n produits doivent passer dans le même ordre sur 3 machines.

Nous imposons une opération initiale à exécuter sur chacun des n produits. Cette opération débute à l'instant 0 et se termine quand la première opération du produit commence. Elle a une durée minimale nulle et maximale illimitée. Chacune de ces opérations nécessite une unique ressource qui lui est propre.

Nous définissons à partir de ces n produits, un nouveau produit qui requiert l'assemblage de ces n produits. L'opération d'assemblage final des produits a une durée minimale nulle et une durée maximale non limitée. Minimiser le *makespan* de ce nouveau produit né de leur assemblage est équivalent à minimiser le *makespan* des n produits (puisque l'assemblage se fait instantanément). Nous en déduisons que déterminer le *makespan* d'un produit nécessitant un assemblage et pour lequel il existe des conflits de ressource est NP-difficile au sens fort.

b) Nous en déduisons que les Problèmes III- 1 à III- 5 sont NP-difficiles au sens fort

Pour toute solution proposée pour un de ces problèmes, nous pouvons vérifier en un temps polynomial qu'elle vérifie toutes ses contraintes, et calculer la valeur du critère.

De plus, nous déduisons de a) que les Problèmes III- 1 à III- 2 sont NP-difficiles au sens fort.

Nous posons pour toutes les opérations des n composants du produit décrit en a) un coût d'utilisation des ressources égales à $s_{i,j}=1$. Le coût de production de ces n produits et de leur assemblage est égal à n fois la longueur de son *makespan.* Par conséquent, résoudre le Problème III- 5 (i.e. déterminer revient l'ordonnancement de coût minimal) revient à minimiser le *makespan* des n produits. Nous en déduisons que le Problème III- 5 est également NP-difficile au sens fort.

Il est possible de montrer le Problème III- 1 est un cas particulier du Problème III- 3, ainsi que le Problème III- 2 est un cas particulier du Problème III- 4. Pour le montrer, nous remplaçons

la variable $T_{h,j}$ par $M\text{-}T_{h,j}$ dans les Problèmes III- 1 et III- 2, où M est choisi grand. De plus, nous remplaçons chaque arc (h,k) par l'arc (k,h), pour $(h,k)\in I_j$. Nous en déduisons que les Problèmes III- 3 à III- 4 sont NP-difficiles au sens fort.

CQFD

On ne connaît aucun algorithme polynomial (ou même pseudo polynomial) pour résoudre les problèmes NP-difficiles au sens fort. Les seules méthodes exactes existantes consistent à énumérer explicitement ou implicitement (avec une procédure par séparation et évaluation progressive, par exemple) l'ensemble des solutions possibles. Dans notre cas, une solution possible est un ordre d'exécution des opérations en conflit. Si peu d'opérations sont en conflit, de telles méthodes sont utilisables.

Le contexte en temps réel interdit d'utiliser ces méthodes gourmandes en temps de calcul pour des problèmes de taille importante (i.e. dans des situations où le nombre d'opérations en conflit est important). Dans ce cas, nous sommes réduits à appliquer des méthodes approchées.

Nous renvoyons le lecteur à la littérature abondante existant sur ces méthodes exactes et approchées : Brucker (1995), Chrétienne et *al.* (1995), Pinedo (1995), Chu et Proth (1996), Koole (1996), Lai et *al.* (1997), et Shmelev (1997), Esquirol et Lopez (1999). Plus spécifiquement, Blazewicz et *al.* (1996), et Jain et Meeran (1999) proposent des états de l'art sur les approches de résolution des problèmes de type *job shop*.

Finalement, en présence d'opérations en conflit, les méthodes décrites dans ce chapitre peuvent être utilisées pour :

a) soit déterminer rapidement une solution réalisable et optimale pour un ordre donné d'exécution des opérations en conflit,

b) soit donner une solution sous-optimale (pour un ordre partiel donné d'exécution des opérations en conflit) qui permet d'éviter l'exploration de toutes les situations.

III.7.2.5. Cas où le ré-ordonnancement est possible

Nous avons considéré que l'ordonnancement d'un produit est défini une fois pour toutes et ne peut donc être remis en question quand une commande apparaît (voir Hypothèse III- 3). Dans des applications, il est possible d'ordonnancer toutes les opérations qui ne sont pas terminées ou celles qui n'ont pas été encore réalisées. En ce qui concerne les opérations en cours d'exécution, il est possible de modifier leur date de fin. Pour les autres opérations, nous pouvons modifier

les dates de début et de fin. Il est également possible de changer l'ordre d'exécution des opérations sur les machines.

Evidemment, étant donné que le problème de *flow shop* à 3 machines du type $F_3|nwt|C_{max}$ est NP-difficile au sens fort, le problème de ré-ordonnancement est NP-difficile au sens fort même dans le cas de produits fabriqués suivant des gammes linéaires comportant seulement 3 opérations. Nous ne pouvons alors résoudre ce type de problèmes en temps réel et nous utilisons des heuristiques.

Outre les algorithmes présentés dans cette thèse qui peuvent être utilisés comme heuristiques pour ce problème, d'autres méthodes approchées ont été proposées s'appuyant sur des règles de priorité [Golenko-Ginzburg et Kats], la programmation par contraintes [Cheng et Smith, 1995], l'exploration arborescente [Ge et Yih, 1995] [Lamothe, 1996], ou encore les méta heuristiques [Bloch, 1999]. Puisque le ré-ordonnancement de plusieurs produits peut être vu comme un ordonnancement en présence de conflits de ressources (voir démonstration du Théorème III- 6), ces approches sont similaires à celles qui sont développées en présence de conflits de ressources (voir section précédente).

Pour terminer, notons une approche particulière. Yih (1994) propose de ne pas remettre en cause l'ordre dans lequel les opérations déjà ordonnancées doivent être exécutées sur les machines. Seul le décalage de ces opérations est autorisé. Ceci permet de réaliser certaines opérations du dernier produit commandé entre des opérations déjà ordonnancées. Ce principe permet d'accroître la productivité du système tout en limitant le temps de calcul. Pour cela, nous modélisons l'ensemble des opérations déjà ordonnancées sous forme d'un graphe PERT (voir Section III.7.2.2.). Les opérations à exécuter sur le produit commandé peuvent alors être insérées entre les opérations déjà ordonnancées que nous choisissons.

III.7.2.6. Conclusion

Dans cette section, nous avons montré comment étendre l'étude présentée dans ce chapitre aux situations où les Hypothèses III- 3, III- 13, III- 15 et III- 16 (voir section 2) ne sont pas vérifiées. De plus, nous avons montré comment utiliser les résultats de ce chapitre dans d'autres approches possibles.

CHAPITRE IV

PROBLEMES A GAMMES ALTERNATIVES

Résumé

Dans ce chapitre, nous proposons d'introduire des méthodes de gestion en temps réel adaptées aux systèmes de production pour lesquels plusieurs gammes permettent de fabriquer un même produit. Nous devons choisir parmi les gammes alternatives de chaque produit celle qui garantit la meilleure productivité du système de production. Nous proposons des méthodes de gestion applicables à des systèmes de production du type des *problèmes mixtes*. Tous les algorithmes proposés dans ce chapitre sont originaux. Ils permettent de déterminer le délai de livraison de chaque produit, et d'ordonnancer chaque produit de manière à minimiser le temps et le coût d'utilisation des ressources.

Sommaire

IV.1. INTRODUCTION

Dans ce chapitre, nous nous intéressons à des systèmes de production dont les caractéristiques sont les mêmes que celles qui sont présentées dans les deux chapitres précédents. De plus, nous considérons que, dans certains cas, le passage d'un état du produit à un autre peut se faire de plusieurs manières. Les différentes gammes possibles permettant de réaliser un produit sont appelées des gammes alternatives. L'objectif est de déterminer la "meilleure" gamme et d'ordonnancer en temps réel chaque produit dès que la commande est passée. Notons que le choix d'une gamme alternative pour un produit donné n'implique pas l'utilisation de cette même gamme pour un autre produit de même type.

Nous illustrons notre étude par la gestion d'une unité de production où plusieurs machines de performances différentes sont capables de réaliser les mêmes opérations.

IV.1.1. L'ordonnancement d'unités automatisées avec choix de machines de performances différentes

L'atelier que nous considérons est un système automatisé. Les stockages intermédiaires y sont très limités (généralement, un seul produit peut être stocké en amont ou en aval de chaque machine). Il comporte plusieurs machines capables de réaliser les mêmes opérations. Ces machines, de technologies ou d'âges différents, ont des performances et des coûts d'utilisation qui ne sont pas identiques. Pour chaque produit, nous pouvons choisir une série de machines différentes. L'objectif est de déterminer la meilleure série de machines au moment d'ordonnancer un produit.

La Figure IV- 1 présente l'ensemble des gammes alternatives d'un produit et le terme sous-gamme désigne une partie d'une gamme.

Dans l'exemple de cette figure, le produit à réaliser subit trois opérations de transformation. Nous devons tout d'abord apporter la matière première à un poste de découpe. Deux postes (l'un utilisant le Laser, l'autre le Plasma) permettent de réaliser la découpe. Du poste où est réalisée la découpe, la matière est déplacée vers un deuxième poste où elle est poinçonnée. Ensuite, elle est de nouveau déplacée vers l'un des deux postes de pliages avant de sortir de l'atelier.

Dans l'exemple de la Figure IV- 1, le passage de l'état de matière première à l'état de matière découpée peut se faire de deux manières : soit en utilisant la première sous-gamme alternative correspondant à une découpe au Laser, soit en utilisant la seconde sous-gamme par un procédé Plasma. Sur la Figure IV- 1, les hémicycles noirs marquent l'état du produit au début des sous-gammes alternatives, et à leur fin.

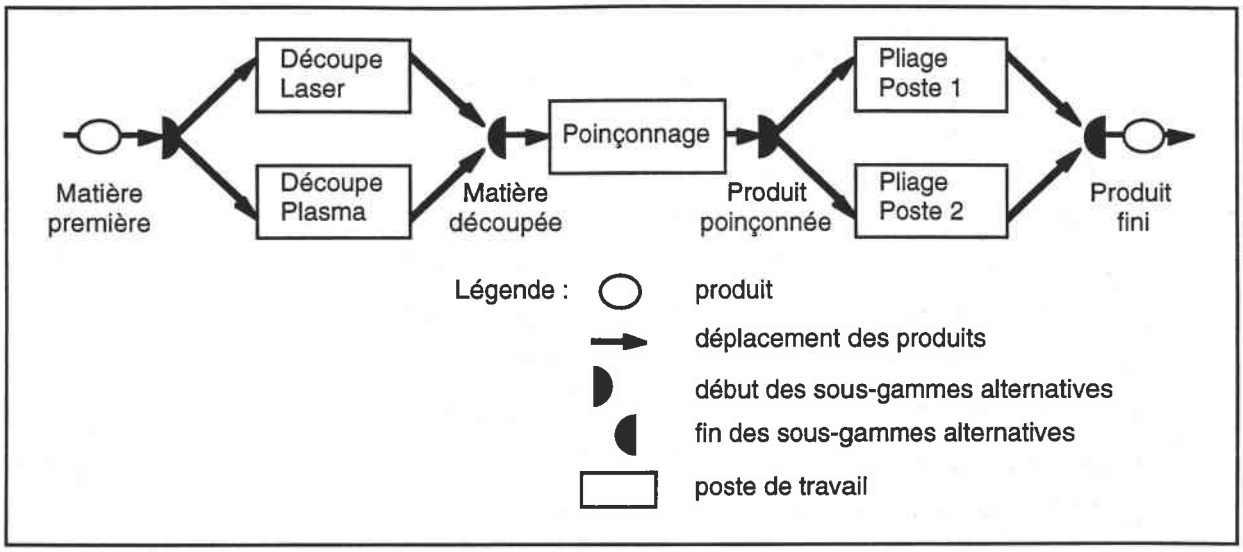

Figure IV- 1 : Exemple d'une unité automatisée comportant des choix de machines

IV.1.2. Approche proposée

Pour faciliter la lecture, ce chapitre est organisé comme les deux précédents. Ainsi, dans la section 2, nous précisons les hypothèses.

Le but du chapitre est de donner des modes de gestion en temps réel qui proposent un ordonnancement de chaque produit dès que sa commande est connue. **L'objectif global est d'obtenir une gestion qui maximise la productivité du système de production à long terme**. Le premier mode de gestion consiste **à fabriquer le produit au plus tôt** (c'est l'objet de la section 3). En outre, il permet de fournir au client un délai minimum de livraison. Ensuite, nous proposons un deuxième mode de gestion qui revient **à commencer la fabrication du produit au plus tard**, tout en livrant le produit au plus tôt (voir section 4). Enfin, nous donnons un troisième mode de gestion dont le but est de **minimiser le coût d'utilisation des ressources** (voir section 5), tout en livrant le produit au plus tôt.

Chaque section est illustrée par un exemple de gestion d'une unité de tôlerie fine. Des travaux de recherche sur les gammes alternatives ont déjà été publiés dans un contexte où l'attente des produits est permise [Chauvet et *al.*, 1998a] [Chauvet et Proth, 1999].

IV.2. HYPOTHESES

Dans cette section, le cadre d'étude est décrit. Les hypothèses faites sont les mêmes que celles faites dans les deux chapitres précédents, mis à part l'Hypothèse IV- 4 qui est modifiée pour tenir compte de l'existence de gammes alternatives. Ainsi, cette hypothèse permet d'élargir le champ d'application de l'étude. Nous rappelons l'ensemble des hypothèses et les notations nécessaires à la lecture de ce chapitre.

Hypothèse IV- 1 : *L'ordonnancement est en temps réel.* Nous devons ordonnancer chaque produit dès que sa commande arrive. Nous numérotons les demandes de 1 à n dans l'ordre de leur arrivée. Nous désignons par J l'ensemble des demandes.

Hypothèse IV- 2 : *La matière première et les composants du produit à fabriquer sont disponibles.* Nous supposons que la matière première et les composants du produit à fabriquer sont disponibles à la date r_j.

Hypothèse IV- 3 : *L'ordonnancement d'un produit ne remet pas en question l'ordonnancement des précédents.*

Hypothèse IV- 4 : La gamme de chaque produit peut comporter des opérations d'assemblage ou de désassemblage, et des sous-gammes alternatives.

Nous supposons que les gammes des produits à réaliser peuvent comporter des assemblages et des désassemblages, comme dans le chapitre précédent. De plus, nous considérons que la gamme de chaque produit peut être choisie parmi plusieurs alternatives. Nous appelons sous-gamme l'ensemble des opérations permettant le passage d'un état h du produit à un état h'. Cette sous-gamme est dite alternative s'il existe d'autres sous-gammes permettant le passage de l'état h à l'état h'. Les états h et h' sont appelés **les extrémités communes de l'ensemble des sous-gammes alternatives** (permettant le passage de l'état h à l'état h').

Pour éviter toute confusion, nous considérons qu'un état ne peut être l'extrémité commune de plusieurs ensembles de sous-gammes alternatives. Nous désignons par $e_{h,j}$ l'indice de l'ensemble des sous-gammes alternatives du produit j dont h est l'extrémité commune. Nous désignons par E_j l'ensemble des indices $e_{h,j}$. Nous avons $E_j = \bigcup_{h \in H'_j} e_{h,j}$ où H'_j est l'ensemble des états du produit j qui sont l'extrémité commune d'un ensemble de sous-gammes alternatives. Pour tout ensemble des sous-gammes alternatives $e \in E_j$, nous désignons par $v_{e,j}$, le nombre de sous-gammes alternatives.

Les exemples de produits présentés dans ce chapitre sont particuliers :

a) leurs sous-gammes alternatives comportent le même nombre d'opérations,

b) ils ne comprennent que deux extrémités pour un ensemble de sous-gammes alternatives (ceci n'est pas toujours vrai quand le produit nécessite des assemblages),

c) aucune sous-gamme alternative n'est incluse dans une autre.

Nous pouvons imaginer des situations où ces conditions ne seraient pas vraies. Néanmoins, toute l'étude et les algorithmes qui suivent resteraient valides dans ces cas plus complexes.

La Figure IV- 2 représente une telle gamme. La gamme de cet exemple comporte $|E_j|=2$ ensembles de sous-gammes alternatives. Le premier ensemble de sous-gammes alternatives permet de passer de l'état 2 du produit à l'état 12 et comporte $v_{e_{2,j},j}=v_{e_{12,j},j}=3$ sous-gammes alternatives. Le second ensemble admet $v_{e_{13,j},j}=v_{e_{20,j},j}=2$ sous-gammes alternatives. Chacune des 12 opérations est représentée par un arc. Les états du produit au début et à la fin des sous-gammes alternatives sont marqués par des hémicycles noirs.

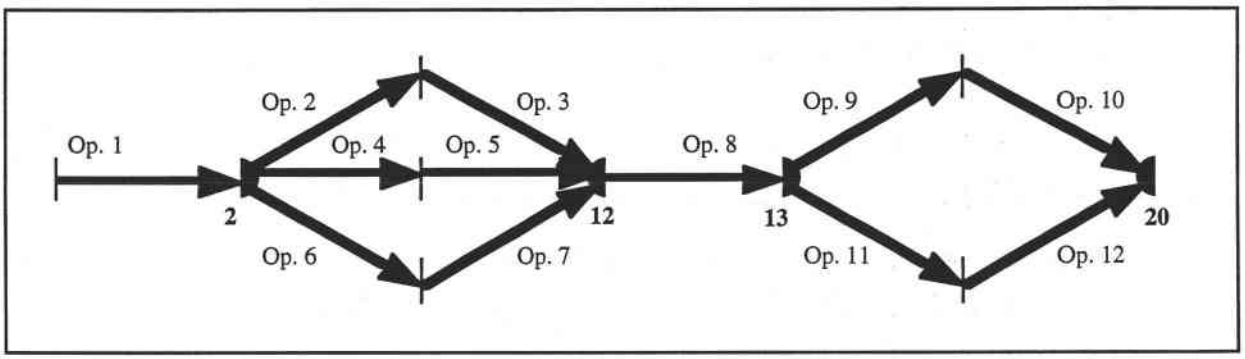

Figure IV- 2 : Représentation de gammes avec deux sous-gammes alternatives

Ainsi, dans cet exemple, après l'opération 1, nous pouvons exécuter soit les opérations 2 et 3, soit les opérations 4 et 5, soit encore les opérations 6 et 7. De même, après l'opération 6, nous pouvons exécuter soit les opérations 9 et 10, soit les opérations 10 et 12. Par conséquent, le produit à réaliser est défini par l'une des six gammes suivantes : {1, 2, 3, 8, 9, 10}, {1, 4, 5, 8, 9, 10}, {1, 6, 7, 8, 9, 10}, {1, 2, 3, 8, 11, 12}, {1, 4, 5, 8, 11, 12} et {1, 6, 7, 8, 11, 12}. Nous devons choisir la meilleure gamme parmi les six gammes possibles.

Pour le produit j, nous notons m_j le nombre d'opérations de sa gamme (y compris les opérations des sous-gammes alternatives) et I_j l'ensemble des opérations.

Hypothèse IV- 5 : *Nous savons quelles sont les ressources interchangeables en cours d'opération.*

Hypothèse IV- 6 : *Le temps de changement de ressource en cours d'opération (lorsque cela est possible) est nul.*

Hypothèse IV- 7 : *Chaque opération peut nécessiter l'utilisation de plusieurs types de ressources.*

Hypothèse IV- 8 : *L'utilisation d'une ressource ne dépend pas d'autres utilisations de la même ressource.*

Hypothèse IV- 9 : *Les exemplaires d'un même type de ressources sont identiques.* Nous pouvons toujours nous ramener à une situation où cette hypothèse est vérifiée. Pour cela, nous définissons une sous-gamme alternative pour chaque opération utilisant une ressource dont les exemplaires ne sont pas identiques. Chaque alternative correspond alors à un exemplaire différent de la ressource. Les exemplaires de chaque ressource utilisés par les opérations de chaque gamme alternative sont alors identiques.

Hypothèse IV- 10 : *Les disponibilités des ressources sont connues.*

Hypothèse IV- 11 : *Les opérations se suivent sans interruption.* Puisque les opérations se suivent sans attente, l'instant de fin d'une opération est également l'instant de début des opérations qui la suivent. Nous représentons les données relatives à un produit j par un graphe orienté $G_j=(H_j,I_j)$, comme celui de la Figure IV- 3.

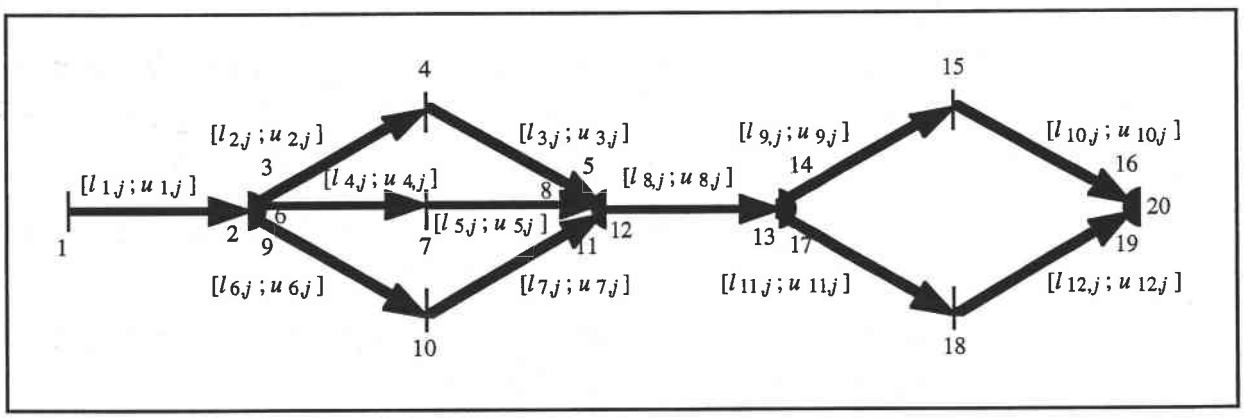

Figure IV- 3 : Graphe représentant la gamme d'un produit

Le graphe $G_j=(H_j,I_j)$ est composé d'un ensemble H_j de sommets marquant les instants de début et de fin des opérations. Ces sommets sont reliés par m_j arcs représentant l'ensemble I_j des opérations. L'ensemble des états du produit j qui sont l'extrémité commune d'un ensemble de sous-gammes alternatives, est $H'_j \subseteq H_j$. Le nombre de sommets du graphe est alors égal à $|H_j|=m_j+1+\sum_{e\in E_j}(v_{e,j}+1)$, où m_j est le nombre d'opérations de la gamme et $v_{e,j}$ le nombre de gammes alternatives correspondant à l'ensemble de sous-gammes alternatives $e\in E_j$.

$X_{e,j}$ est l'indice de la sous-gamme alternative retenue pour l'ensemble d'indice e. On a : $1\le X_{e,j}\le v_{e,j}$, pour $e\in E_j$. Nous devons déterminer les sous-gammes alternatives $X_{e,j}$ qui permettent d'obtenir le meilleur ordonnancement possible. Nous considérons qu'un état k ne peut appartenir à deux sous-gammes alternatives d'un même ensemble. Nous désignons par

$d_{k,e,j}$ l'indice de la sous-gamme alternative à laquelle appartient l'état k, pour l'ensemble de sous-gammes e.

Au début et à la fin de chaque sous-gamme alternative, nous définissons autant de sommets qu'il y a d'alternatives possibles. La Figure IV- 4 reprend la numérotation des sommets au début des premières sous-gammes alternatives de la Figure IV- 3. Le sommet 2 correspond à la fin de l'opération 1. Les sommets 3, 6 et 9 correspondent au début des opérations 2, 4 et 6.

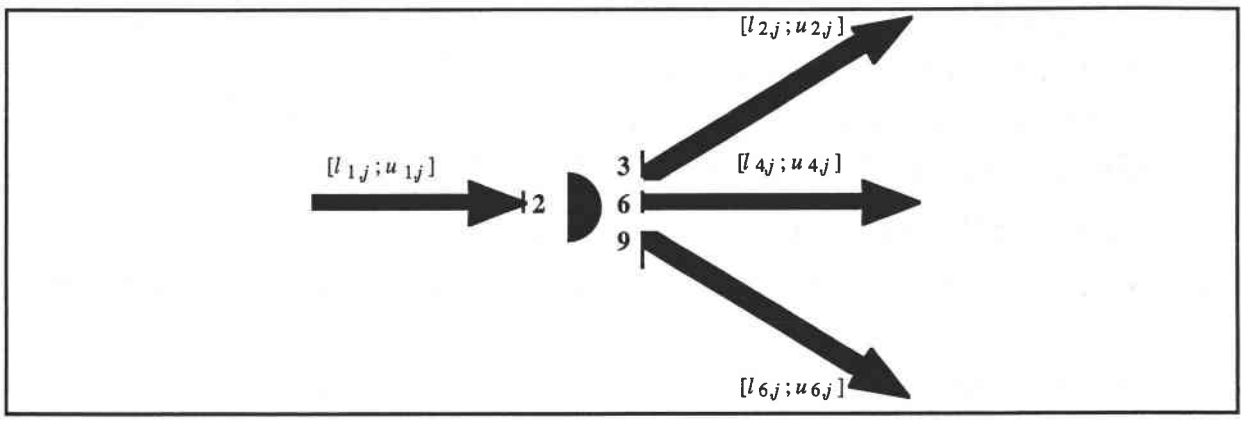

Figure IV- 4 : Numérotation des sommets au début d'une sous-gamme à alternatives

Supposons que sur l'exemple de la Figure IV- 3, la première sous-gamme alternative soit retenue pour passer de l'état 2 du produit à l'état 12, i.e. $X_{e_{2,j},j}=1$. Pour que l'ordonnancement soit valide, deux conditions doivent être vérifiées :

a) Il faut que la date de fin de l'opération 1 (i.e. correspondant à l'état 2) soit égale à la date de début de l'opération 2 (i.e. correspondant à l'état 3).

b) Il faut que la date de fin de l'opération 3 (i.e. correspondant à l'état 5) soit égale à la date de début de l'opération 8 (i.e. correspondant à l'état 12).

Pour toute sous-gamme alternative d'indice X, nous désignons par $\aleph_{e,j}(X)$ l'ensemble des couples de sommets qui doivent coïncider si l'alternative X est choisie parmi l'ensemble e des sous-gammes possibles pour réaliser le produit j. Le premier sommet du couple est une des extrémités communes de l'ensemble des sous-gammes alternatives et le second correspond au sommet de la sous-gamme retenue. Dans notre exemple, nous avons $\aleph_{e_{2,j},j}(1)=\{(2\,;3)\,;(12\,;5)\}$. De même, nous avons $\aleph_{e_{2,j},j}(2)=\{(2\,;6)\,;(12\,;8)\}$.

Pour toute sous-gamme alternative d'indice X de l'ensemble e, les couples de l'ensemble $\aleph_{e,j}(X)$ sont considérés, dans la suite, comme de nouveaux arcs de notre graphe. Nous désignons par $\Im_j$ l'ensemble de ces arcs, $\Im_j = \bigcup_{\substack{1\le X\le v_{e,j} \\ e\in E_j}} \aleph_{e,j}(X)$. Pour ces arcs, nous étendons la définition de successeur et prédécesseur d'un sommet. Un sommet k de l'ensemble H_j est un **successeur possible du sommet** h si et seulement s'il existe un couple (h,k) dans $\Im_j$

reliant h à k. Nous pouvons noter cet arc $i=(h,k)$. De même, un sommet h de l'ensemble H_j est **prédécesseur du sommet** k s'il existe un couple (h,k) dans $\Im_j$. Notons que pour tout couple (h,k) de $\Im_j$, h est toujours prédécesseur du sommet k tandis que k n'est un successeur de h qu'à la condition que la sous-gamme alternative correspondante soit choisie.

Hypothèse IV- 12 : *Les durées des opérations sont contrôlables.*

Hypothèse IV- 13 : *La préemption est interdite.*

Hypothèse IV- 14 : *L'objectif est d'ordonnancer chaque produit dès que sa commande est enregistrée de manière à maximiser la productivité du système à long terme.* Nous désignons par $T_{h,j}$ la date à laquelle correspond l'instant d'indice h. La date de début de l'opération $i=(h,k)$ effectuée sur le produit j est notée $B_{i,j}=T_{h,j}$ et sa date de fin $C_{i,j}=T_{h,k}$. Ordonnancer un produit revient à déterminer dans quelle période $Z_{i,j}$ doit être exécutée l'opération i sur le produit j, ainsi que ses dates de début $B_{i,j}$ et de fin $C_{i,j}=B_{i,j}+P_{i,j}$. Pour atteindre une productivité maximale du système à long terme, nous étendons les méthodes de gestion présentées dans les chapitres précédents :

a) **terminer chaque produit le plus tôt possible** (voir section 3),

b) **commencer la fabrication du produit au plus tard**, tout en respectant la date de livraison fixée au plus tôt (voir section 4),

c) **minimiser le coût d'utilisation des ressources de chaque produit**, tout en respectant la date de livraison fixée au plus tôt (voir section 5).

Hypothèse IV- 15 : *Dans une gamme alternative donnée, deux opérations ne peuvent avoir à la fois une même opération qui les précède, et une même opération qui les suit* . Cette hypothèse nous autorise à étendre la numérotation des sommets introduite dans le chapitre précédent. Nous numérotons l'ensemble des sommets du graphe G_j de manière à ce que :

a) Pour tout sommet d'indice h, il n'existe pas plus d'un sommet d'indice $k>h$ parmi les successeurs directs et les prédécesseurs de h.

b) Si pour un sommet d'indice h il existe un tel sommet $k>h$, alors tous les successeurs possibles du sommet h ont un indice inférieur à h.

Les sommets de la Figure IV- 3 sont numérotés de cette manière. Il est possible d'obtenir une telle numérotation par un algorithme de complexité égale à $O(|H_j|)$. **Cette numérotation est essentielle pour les algorithmes présentés dans la suite.**

Hypothèse IV- 16 : *Il n'existe pas de conflits de ressources entre les opérations pouvant être effectuées en parallèle sur un même produit.* Cette hypothèse ne concerne pas les opérations de deux alternatives différentes. En effet, seules les opérations de l'une des deux alternatives sont effectuées.

IV.3. MINIMISATION DU *MAKESPAN*

IV.3.1. Introduction

Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui ordonnance chaque produit au plus tôt, i.e. qui minimise le *makespan* de chaque produit. Nous formalisons le problème avant de montrer son niveau de difficulté. Ensuite, nous proposons un algorithme original.

Si le nombre de gammes alternatives pour réaliser un produit est réduit, nous pouvons tester les alternatives une par une et garder la "meilleure". Mais, quand le nombre de sous-gammes alternatives augmente, cette méthode devient combinatoire et n'est plus applicable. C'est pourquoi nous proposons une approche différente dans la suite.

IV.3.2. Formalisation

Nous voulons déterminer les dates de début des opérations effectuées sur le produit j sachant que la date de disponibilité est r_j (voir contraintes (IV-1)). Les dates $T_{h,j}$ et $T_{k,j}$ définissent les instants de début et de fin de chaque opération $(h,k) \in I_j$.

La durée d'exécution $P_{(h,k),j}=T_{k,j}-T_{h,j}$ de chaque opération $i=(h,k)$ est à déterminer et doit vérifier : $l_{i,j} \leq P_{i,j} \leq u_{i,j}$ (voir contraintes (IV-2)). Chaque opération i peut être exécutée dans l'une des $y_{i,j}$ fenêtres de temps de l'ensemble $W_{i,j}=\{[a_{z,i,j} \ ; \ b_{z,i,j}] \ / \ 1 \leq z \leq y_{i,j}\}$ (voir contraintes (IV-3)-(IV-4)). La fenêtre retenue est la fenêtre d'indice $Z_{i,j}$ (voir contraintes (IV-5)).

De même, la sous-gamme alternative d'indice $X_{e,j}$ est la sous-gamme retenue (voir contraintes (IV-6)), pour $e \in E_j$. Pour chaque sous-gamme alternative retenue, les dates des opérations doivent coïncider avec les dates des opérations extérieures à la sous-gamme (voir contraintes (IV-7)).

Notre objectif est de minimiser la date d'achèvement du produit, i.e. la date de fin de la dernière opération effectuée sur le produit. Nous notons $H_j \backslash H'_j$ l'ensemble des sommets du graphe qui n'appartiennent pas à une sous-gamme alternative, $H'_j \subseteq H_j$. Nous considérons que le début et la fin d'exécution du produit n'appartiennent pas à une alternative.

Les contraintes (IV-8) à (IV-10) donnent les intervalles de définition de toutes les variables de décisions $T_{h,j}$, $Z_{i,j}$. et $X_{e,j}$.

Ce problème s'écrit :

Problème IV- 1 : Minimisation du *makespan* de chaque produit

$$\min \quad \max_{h \in H_j \setminus H'_j} \left\{ T_{h,j} \right\}$$

$$\text{tel que} \begin{cases} r_j \le T_{h,j} & \forall h \in H_j & \text{(IV - 1)} \\ l_{(h,k),j} \le T_{k,j} - T_{h,j} \le u_{(h,k),j} & \forall (h,k) \in I_j & \text{(IV - 2)} \\ a_{Z_{(h,k),j},(h,k),j} \le T_{h,j} & \forall (h,k) \in I_j & \text{(IV - 3)} \\ T_{k,j} \le b_{Z_{(h,k),j},(h,k),j} & \forall (h,k) \in I_j & \text{(IV - 4)} \\ 1 \le Z_{i,j} \le y_{i,j} & \forall i \in I_j & \text{(IV - 5)} \\ 1 \le X_{e,j} \le v_{e,j} & \forall e \in E_j & \text{(IV - 6)} \\ T_{h,j} = T_{k,j} \quad \forall (h,k) \in \aleph_{e,j}\left(X_{e,j}\right) & \forall e \in E_j & \text{(IV - 7)} \\ T_{h,j} \in \Re^+ & \forall h \in H_j & \text{(IV - 8)} \\ Z_{i,j} \in \mathrm{N} & \forall i \in I_j & \text{(IV - 9)} \\ X_{e,j} \in \mathrm{N} & \forall e \in E_j & \text{(IV - 10)} \end{cases}$$

Avant de proposer une méthode de résolution, nous montrons le niveau de difficulté du Problème IV- 1. Pour cela, nous montrons qu'il appartient à une classe de problèmes pour lesquels on ne connaît pas d'algorithmes polynomiaux.

Théorème IV- 1

Trouver une solution réalisable au Problème IV- 1 est un problème NP-complet, et le Problème IV- 1 est un problème NP-difficile.

Démonstration

Pour n'importe quelle solution au Problème IV- 1 (i.e. $\{T_{h,j}\}_{h \in H_j}$, $\{Z_{i,j}\}_{i \in I_j}$ et $\{X_{e,j}\}_{e \in E_j}$), nous pouvons vérifier en un temps polynomial qu'une solution est réalisable ou non.

Considérons le problème suivant :

Soit un ensemble fini A de valeurs entières positives $A = \{a_1, a_2, \ldots, a_{|A|}\}$ et un entier positif B. Existe-t-il un sous-ensemble $A' \subseteq A$ tel que la somme des entiers de A' soit exactement égale à B ? Répondre à cette question (désigné en anglais par *subset sum*) est un problème NP-complet [Garey et Johnson, 1979].

Montrons que ce problème est un cas particulier du nôtre. Pour cela, considérons le produit j qui doit subir $m_j = |A| + 2$ transformations successives. La première opération de transformation a une durée minimale nulle $l_{1,j}=0$ et doit être réalisée pendant la période [0 ; 0]. La dernière opération a également une durée minimale nulle $l_{m_j,j}=0$ et doit être réalisée pendant

la période $[B\ ;\ B]$. Chaque transformation i intermédiaire (i.e. $2 \le i \le m_j - 1$) peut être réalisée par une machine en un temps $l_{i,j} = u_{i,j} = a_{i-1}$. Les transformations intermédiaires sont toutes facultatives. Aussi, pour chacune, nous définissons deux sous-gammes alternatives :

a) la première qui consiste à réaliser une opération à l'aide d'une machine en un temps $l_{i,j} = u_{i,j} = a_{i-1}$,

b) la seconde qui revient à réaliser une opération en un temps nul $l_{i,j} = u_{i,j} = 0$ (i.e. dans ce cas, aucune transformation n'est réalisée).

Déterminer un ordonnancement admissible pour ce produit j permet de résoudre simultanément le problème de type *subset sum* associé.

Par conséquent, trouver une solution réalisable au Problème IV- 1 est un problème NP-complet et le Problème IV- 1 est un problème NP-difficile.

CQFD

Dans la section suivante, nous proposons une approche directe de résolution du Problème IV- 1 en un temps pseudo polynomial.

IV.3.3. Sélection des fenêtres

Pour résoudre le Problème IV- 1, il faut déterminer quelle gamme alternative réalisée, dans quelle fenêtre $Z_{i,j}$ et entre quelles dates $T_{h,j}$ et $T_{k,j}$ l'opération $i=(h,k)$ doit être effectuée de manière à ce que le *makespan* du produit soit minimal.

L'approche proposée est une extension de l'Algorithme II- 6 du chapitre II et de l'Algorithme III- 5 du chapitre précédent. Elle consiste à discrétiser le temps. Nous considérons que les durées des opérations ainsi que les dates de début et de fin des fenêtres sont des valeurs rationnelles. En multipliant toutes ces valeurs par le plus petit multiple commun de leur dénominateur Q, nous transformons notre problème de manière à n'avoir que des dates et des durées entières. Nous admettons donc que ces valeurs sont entières.

L'approche est basée sur la programmation dynamique. Pour cela, nous définissons $|H_j|$ étapes correspondant aux $|H_j|$ états du produit j au cours de sa réalisation (y compris les états des gammes alternatives). Pour chaque état du produit, nous énumérons toutes les dates possibles pour lesquelles le produit peut être dans cet état. Nous recherchons l'ensemble des dates de chaque état qui correspondent à une fin au plus tôt du produit et, par conséquent, nous évaluons chaque date en fonction du *makespan* qu'elle permet d'atteindre.

Avant d'appliquer cette approche, nous calculons une borne supérieure du *makespan* optimal pour ce problème. Pour cela, nous choisissons arbitrairement les sous-gammes alternatives. Le Problème IV- 1 que l'on obtient sans gamme alternative est équivalent au Problème III- 1. Nous

le résolvons en appliquant l'Algorithme III- 2. Nous désignons par $\bar{d}_j$ la date de fin que l'on obtient pour le produit j. Cette date est évidemment une borne supérieure du *makespan* optimal du Problème IV- 1.

La date $T_{h,j}$ correspond à l'instant de fin d'exécution des opérations $(f,h)\in I_j$ et de début d'exécution des opérations $(h,k)\in I_j$ sur le produit j. Puisque le produit ne peut commencer avant la date r_j (voir définition des contraintes (IV-1)) et peut être terminé à la date $\bar{d}_j$, nous avons :

$r_j \le T_{h,j} \le \bar{d}_j$, pour $1 \le h \le m_j+1$. (IV-11)

La durée de chaque opération i est contrôlable et peut être choisie dans l'intervalle $[l_{i,j};u_{i,j}]$ (voir contrainte (IV-2)). Ainsi, les dates de début de l'opération (f,h) qui autorisent une fin à la date $T_{h,j}$ sont comprises entre $T_{h,j}$-$u_{(f,h),j}$ et $T_{h,j}$-$l_{(f,h),j}$. De même, les dates de fin de l'opération (h,k) qui autorisent un début d'exécution à la date $T_{h,j}$ sont comprises entre $T_{h,j}+l_{(h,k),j}$ et $T_{h,j}+u_{(h,k),j}$.

Si nous décidons de commencer l'opération $(f,h)\in I_j$ à l'instant $T_{f,j}$ et de la terminer à l'instant $T_{h,j}$, deux cas peuvent se présenter :

a) l'intervalle $[T_{f,j}\ ;\ T_{h,j}]$ est inclus dans une période de disponibilité des ressources, $[a_{z,(f,h),j};b_{z,(f,h),j}]$ où $1\le z\le y_{(f,h),j}$. Dans ce cas, nous en déduisons que le *makespan* du produit est supérieur à $T_{h,j}$ date à laquelle l'opération (f,h) se termine.

b) l'intervalle $[T_{f,j}\ ;\ T_{h,j}]$ n'est pas inclus dans une période de disponibilité des ressources ; dans ce cas, nous en déduisons que l'opération (f,h) ne peut se terminer à l'instant $T_{h,j}$ et nous lui associons un *makespan* de valeur infinie.

Parmi toutes les dates possibles $T_{f,j}$, nous retenons la meilleure date (i.e. celle qui permet d'atteindre le meilleur *makespan*) permettant de terminer l'opération (f,h) à la date $T_{h,j}$. Nous notons $S_{(f,h)}(T_{h,j})$ l'ensemble des dates permettant de terminer l'opération $(f,h)\in I_j$ à la date $T_{h,j}$. Formellement, nous avons :

$S_{(f,h)}(T_{h,j})=\{t\ /\ r_j\le t\le \bar{d}_j,\ T_{h,j}-l_{(f,h),j}\le t\le T_{h,j}-u_{(f,h),j},$ et $\exists 1\le z\le y_{(f,h),j}$ tq. $[t;T_{h,j}]\subseteq[a_{z,(f,h),j};b_{z,(f,h),j}]\}$
pour $r_j\le T_{h,j}\le \bar{d}_j$, $(h,k)\in I_j$, $1\le h\le m_j+1$. (IV-12)

Nous désignons par $C_{(f,h)}(T_{h,j})$ le meilleur *makespan* déduit de l'opération $(f,h)\in I_j$ se terminant à la date $T_{h,j}$:

$$C_{(f,h)}\left(T_{h,j}\right)=\begin{cases}+\infty & \text{si } S_{(f,h)}\left(T_{h,j}\right)=\varnothing \\ \min\limits_{T_{f,j}\in S_{(f,h)}\left(T_{h,j}\right)}\left\{\max\left\{C\left(T_{f,j}\right);T_{h,j}\right\}\right\} & \text{sinon}\end{cases}$$

pour $r_j\le T_{h,j}\le \bar{d}_j$, $(h,k)\in I_j$, $1\le h\le m_j+1$. (IV-13)

De même, si nous décidons de commencer l'opération (h,k) à l'instant $T_{h,j}$ et de la terminer à l'instant $T_{k,j}$, deux cas peuvent se présenter :

a) l'intervalle $[T_{h,j}\ ;\ T_{k,j}]$ est inclus dans une période de disponibilité des ressources, $[a_{z,(h,k),j};b_{z,(h,k),j}]$ où $1\le z\le y_{(h,k),j}$. Dans ce cas, nous en déduisons que le *makespan* du produit est supérieur à $T_{h,j}$ et à $T_{k,j}$.

b) dans l'autre cas, nous associons à (h,k) un *makespan* infini.

Nous notons $S_{(h,k)}(T_{h,j})$ l'ensemble des dates permettant de commencer l'opération $(h,k)\in I_j$ à la date $T_{h,j}$. Formellement, nous avons :

$$S_{(h,k)}(T_{h,j})=\{t/r_j\le t\le \bar{d}_j,\ T_{h,j}+l_{(h,k),j}\le t\le T_{h,j}+u_{(h,k),j},\ \text{et } \exists 1\le z\le y_{(h,k),j} \text{ tq. } [T_{h,j};t]\subseteq[a_{z,(h,k),j};b_{z,(h,k),j}]\}$$
pour $r_j\le T_{h,j}\le \bar{d}_j,\ (h,k)\in I_j,\ 1\le h\le m_j+1.$ (IV-14)

Nous désignons par $C_{(h,k)}(T_{h,j})$ le *makespan* induit par l'opération $(h,k)\in I_j$ commençant à la date $T_{h,j}$:

$$C_{(h,k)}(T_{h,j})=\begin{cases}+\infty & \text{si } S_{(h,k)}(T_{h,j})=\varnothing\\ \min\limits_{T_{k,j}\in S_{(h,k)}(T_{h,j})}\left\{\max\left\{C(T_{k,j});T_{h,j}\right\}\right\} & \text{sinon}\end{cases}$$
pour $r_j\le T_{h,j}\le \bar{d}_j,\ (h,k)\in I_j,\ 1\le h\le m_j+1.$ (IV-15)

Considérons, maintenant, un couple $(f,h)\in \mathfrak{I}_j$. L'état f du produit est un prédécesseur de l'état h : f est donc une des extrémités communes de sous-gammes alternatives et h est l'état qui coïncide avec f si la sous-gamme alternative correspondante est retenue. Même si la sous-gamme alternative associée à ce couple n'est pas retenue, nous voulons que l'état f coïncide avec l'état h. En contraignant l'état f à coïncider avec l'état h, nous ne modifions que les dates des états du produit utilisant cette sous-gamme alternative. Nous désignons par $C_{(f,h)}(T_{h,j})$ le *makespan* induit par l'état f :

$C_{(f,h)}(T_{h,j})=C(T_{f,j})$ pour $r_j\le T_{h,j}\le \bar{d}_j,\ (f,h)\in \mathfrak{I}_j,\ 1\le h\le m_j+1.$ (IV-16)

Considérons, enfin, un couple $(h,k)\in \mathfrak{I}_j$. L'état k du produit est un successeur possible de l'état h. Cela signifie que si la sous-gamme alternative correspondant à ce couple est retenue, les états h et k coïncideront. Nous désignons par $C_{(h,k)}(T_{h,j})$ le *makespan* induit par l'état k :

$C_{(h,k)}(T_{h,j})=C(T_{f,j})$ pour $r_j\le T_{h,j}\le \bar{d}_j,\ (h,k)\in \mathfrak{I}_j,\ 1\le h\le m_j+1.$ (IV-17)

Parmi tous les successeurs possibles de h, nous retenons celui qui nous permet de garantir le meilleur *makespan* et tel que l'état h du produit soit obtenu à la date $T_{h,j}$. Nous en déduisons que le meilleur *makespan* pour lequel l'état h est obtenu à la date $T_{h,j}$ est :

$$C(T_{h,j}) = \max\left\{ \max_{\substack{(h,k)\in I_j \\ k<h}} \left\{ C_{(h,k)}(T_{h,j}) \right\}; \max_{\substack{(f,h)\in I_j \cup \Im_j \\ f<h}} \left\{ C_{(f,h)}(T_{h,j}) \right\}; \min_{\substack{(h,k)\in \Im_j \\ k<h}} \left\{ C_{(h,k)}(T_{h,j}) \right\} \right\}$$

pour $r_j \le T_{h,j} \le \overline{d}_j$, $1 \le h \le m_j + 1$. (IV-18)

La Figure IV- 5 reprend l'exemple précédent. Les dates $T_{h,j}$ possibles de chaque état h du produit sont énumérées à chaque étape. Les états sont considérés dans l'ordre de la numérotation introduite à la suite de l'Hypothèse IV- 15. Les calculs relatifs à l'état h du produit tiennent compte des états f du produit proches (i.e. les successeurs et les prédécesseurs de l'état h) pour lesquels les calculs ont déjà été opérés. Les calculs relatifs à l'état h permettent de définir les dates $T_{h,j}$ pour lesquelles le produit peut être dans l'état h et de connaître le *makespan* minimal correspondant à cette date.

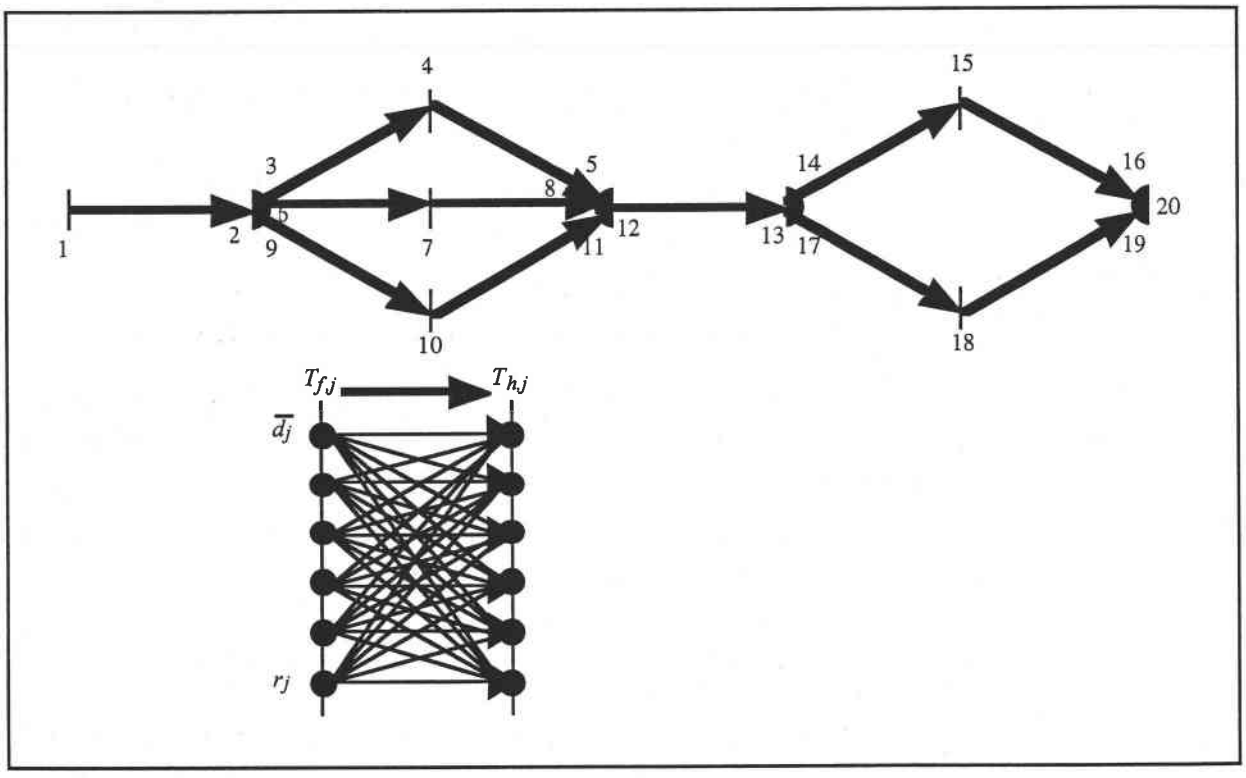

Figure IV- 5 : Les étapes de la programmation dynamique

Si la date $T_{h,j}$ ne correspond pas à une solution réalisable, il lui est associé un *makespan* infini. L'évaluation $C(T_{|H_j|,j})$ donne le *makespan* minimal pour lequel le produit est dans l'état $|H_j|$ à la date $T_{|H_j|,j}$. Si $\{T_{h,j}\}_{1 \le h \le m_j+1}$ est la solution optimale au Problème IV- 1, on a $\max_{h\in H_j}\{T_{h,j}\} = \min_{r_j \le T_{|H_j|,j} \le \overline{d}_j}\left\{C(T_{|H_j|,j})\right\}$. L'Algorithme IV- 1 découle des relations (IV-10) à (IV-18) et permet de résoudre le Problème IV- 1.

Algorithme IV- 1 : Choix des fenêtres pour une fin du produit au plus tôt

Procédure Définition des fenêtres induisant un *makespan* minimal pour le produit j.

Entrée : j est l'indice du produit considéré.

r_j est la date à partir de laquelle on peut commencer à réaliser le produit j.

$\overline{d}_j$ est une borne supérieure du *makespan* minimal, i.e. date avant laquelle le produit j est terminé.

m_j est le nombre d'opérations à réaliser sur le produit j.

H_j est l'ensemble des instants marquant le début et la fin des opérations effectuées sur le produit j.

I_j est l'ensemble des opérations à réaliser sur le produit j.

E_j est l'ensemble des extrémités communes des sous-gammes alternatives du produit j.

$\mathcal{S}_j$ est l'ensemble des couples d'états qui coïncident suivant les sous-gammes alternatives retenues.

$\aleph_{e,j}(X)$ est l'ensemble des couples d'états qui coïncident si la sous-gamme alternative X est retenue parmi l'ensemble d'indice e, pour $e \in E_j$.

$e_{h,j}$ est l'ensemble de sous-gammes alternatives dont h est une extrémité commune, pour $h \in H_j$.

$d_{h,e,j}$ est l'indice de la sous-gamme alternative à laquelle appartient l'état h, pour $h \in H_j$, $e \in E_j$.

$l_{i,j}$ est la durée minimale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$u_{i,j}$ est la durée maximale d'exécution de l'opération i sur le produit j, pour $i \in I_j$.

$y_{i,j}$ est le nombre de fenêtres dans laquelle on peut réaliser l'opération i sur le produit j, pour $i \in I_j$.

$a_{z,i,j}$ est la date de début de la z -ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \leq z \leq y_{i,j}$, $i \in I_j$.

$b_{z,i,j}$ est la date de fin de la z -ième fenêtre de l'opération i effectuée sur le produit j, pour $1 \leq z \leq y_{i,j}$, $i \in I_j$.

Sortie : $T_{h,j}$ est la date correspondant à h, où $h \in H_j$.

$Z_{i,j}$ est la fenêtre dans laquelle on doit réaliser l'opération i sur le produit j, où $i \in I_j$.

$X_{e,j}$ est l'indice de la sous-gamme alternative retenue correspondant à l'ensemble e, où $e \in E_j$.

1. **Initialisation.**

1.1. **Pour** $(f,h) \in I_j$: *On examine chaque opération (f,h) une par une.*

1.1.1. **Si** $(f<h)$: *Les états de l'étape f sont examinées avant ceux de l'étape h.*

1.1.1.1. **Trier** les fenêtres $[a_{z,(f,h),j}\ ;\ b_{z,(f,h),j}]$ pour $1 \leq z \leq y_{(f,h),j}$ dans l'ordre croissant des $a_{z,(f,h),j}$ et, en cas d'égalité, dans l'ordre croissant des $b_{z,(f,h),j}$. *Dans la suite, la z -ième fenêtre est le z -ième élément dans cet ordre.* $1 \leq z \leq y_{z,(f,h),j}$

1.1.2. **Sinon** :

1.1.2.1. **Trier** les fenêtres $[a_{z,(f,h),j}\ ;\ b_{z,(f,h),j}]$ pour $1 \leq z \leq y_{(f,h),j}$ dans l'ordre croissant des $b_{z,(f,h),j}$ et, en cas d'égalité, dans l'ordre croissant des $a_{z,(f,h),j}$

2. **Pour** h allant de 1 à $|H_j|$: *On note h l'indice de l'étape examinée.*

2.1. **Pour** $T_{h,j}$ allant de r_j à $\overline{d}_j$: *On énumère chaque date possible $T_{h,j}$ de l'état h.*

2.1.1. **Pour** $C(T_{h,j}) := T_{h,j}$. *On initialise le coût de la date $T_{h,j}$ à $T_{h,j}$.*

2.1.2. **Pour** $f \in \{f<h \ /\ (f,h) \in I_j\}$: *On note f l'indice de l'étape précédente.*

2.1.2.1. **Poser** $C_{(f,h)}(T_{h,j}) := +\infty$. *On initialise le coût relatif à l'opération (f,h) à $+\infty$.*

2.1.2.2. **Poser** $z := \min\{z \ /\ 1 \leq z \leq y_{(f,h),j},\ b_{z,(f,h),j} \geq T_{h,j}\}$. *On note z la $1^{ère}$ fenêtre permettant de réaliser l'opération (f,h) qui se termine à la date $T_{h,j}$.*

2.1.2.3. **Pour** $T_{f,j}$ allant de $\max\{r_j\ ;\ a_{z,(f,h),j}\ ;\ T_{h,j} - u_{(f,h),j}\}$ à $\min\{\overline{d}_j\ ;\ T_{h,j} - l_{(f,h),j}\}$: *On examine chaque date $T_{f,j}$ permettant de terminer l'opération à la date $T_{h,j}$.*

2.1.2.4. **Si** $(C_{(f,h)}(T_{h,j}) > C(T_{f,j}))$ **alors** :

La date $T_{f,j}$ permet de diminuer le coût.

2.1.2.4.1. **Poser** $C_{(f,h)}(T_{h,j}) := C(T_{f,j})$.

2.1.2.4.2. **Poser** $Sopt_{(f,h)}(T_{h,j}) := T_{f,j}$. *On désigne par $Sopt_{(f,h)}(T_{h,j})$ la date permettant de terminer l'opération (f,h) à la date $T_{h,j}$ au coût le plus bas.*

2.1.2.5. **Poser** $C(T_{h,j}) := \max\{C(T_{h,j}); C_{(f,h)}(T_{h,j})\}$. *On prend en compte l'évaluation de $C_{(f,h)}(T_{h,j})$ dans celle de la date $T_{h,j}$.*

2.1.3. **Pour** $k \in \{k<h \ /\ (h,k) \in I_j\}$: *On note k l'indice de l'étape suivante.*

2.1.3.1. **Poser** $C_{(h,k)}(T_{h,j}):=+\infty$. *On initialise le coût relatif à l'opération* (h,k) *à* $+\infty$.

2.1.3.2. **Poser** $z:=\max\{z \ / \ 1\leq z\leq y_{(h,k),j},\ a_{z,(h,k),j}\leq T_{h,j}\}$. *On note* z *la* $1^{ère}$ *fenêtre permettant de réaliser l'opération* (h,k) *qui commence à la date* $T_{h,j}$.

2.1.3.3. **Pour** $T_{k,j}$ allant de $\max\{r_j \ ; \ T_{h,j}+l_{(f,h),j}\}$ à $\min\{\overline{d}_j \ ; \ b_{z,(f,h),j} \ ; T_{h,j}+u_{(f,h),j}\}$: *On examine chaque date* $T_{k,j}$ *permettant de commencer l'opération à la date* $T_{h,j}$.

2.1.3.4. **Si** ($C_{(h,k)}(T_{h,j})>C(T_{k,j})$) **alors** :

La date $T_{k,j}$ *permet de diminuer le coût.*

2.1.3.4.1. **Poser** $C_{(h,k)}(T_{h,j}):=C(T_{k,j})$.

2.1.3.4.2. **Poser** $Sopt_{(h,k)}(T_{h,j}):=T_{k,j}$. *On désigne par* $Sopt_{(h,k)}(T_{h,j})$ *la date permettant de commencer l'opération* (h,k) *à la date* $T_{h,j}$ *au coût le plus bas.*

2.1.3.5. **Poser** $C(T_{h,j}):=\max\{C(T_{h,j});C_{(h,k)}(T_{h,j})\}$. *On prend en compte l'évaluation de* $C_{(h,k)}(T_{h,j})$ *dans celle de la date* $T_{h,j}$.

2.1.4. **Pour** $f\in\{f<h \ / \ (f,h)\in\Im_j\}$: *On note* f *l'indice de l'étape considérée.*

2.1.4.1. **Poser** $C_{(f,h)}(T_{h,j}):=C(T_{f,j})$. *On identifie les deux coûts.*

2.1.4.2. **Poser** $C(T_{h,j}):=\max\{C(T_{h,j});C_{(f,h)}(T_{h,j})\}$. *On prend en compte l'évaluation de* $C_{(f,h)}(T_{h,j})$ *dans celle de la date* $T_{h,j}$.

2.1.5 **Poser** $C:=C(T_{h,j})$. *On initialise* C *le coût relatif à* $C(T_{h,j})$.

2.1.6. **Pour** $k\in\{k<h \ / \ (h,k)\in\Im_j\}$: *On note* k *l'indice de l'étape considérée.*

2.1.6.1. **Poser** $C_{(h,k)}(T_{h,j}):=C(T_{k,j})$. *On identifie les deux coûts.*

2.1.6.4. **Si** ($C>C_{(h,k)}(T_{h,j})$) **alors** :

La date $T_{k,j}$ *permet de diminuer le coût.*

2.1.6.4.1. **Poser** $C:=C_{(h,k)}(T_{h,j})$.

2.1.6.4.2. **Poser** $X_{e_{h,j}}(T_{h,j}):=d_{k,e_{h,j}j}$. *On désigne par* $X_{e_{h,j}}(T_{h,j})$ *l'indice de l'alternative retenue si le produit est dans l'état* h *à la date* $T_{h,j}$.

2.1.7. **Poser** $C(T_{h,j}):=\max\{C(T_{h,j});C\}$. *On prend en compte l'évaluation*

3. **Poser** $Copt:=+\infty$. *On note* $Copt$ *le meilleur coût trouvé pour un état* $T_{|H_j|,j}$ *de la dernière étape.*

4. **Pour** $T_{|H_j|,j}$ allant de r_j à $\overline{d}_j$: *On énumère chaque date possible* $T_{|H_j|,j}$.

4.1. **Si** ($Copt>C(T_{|H_j|,j})$) **alors** : *La date* $T_{|H_j|,j}$ *à un meilleur coût.*

4.1.1. **Poser** $Copt:=C(T_{|H_j|,j})$.

4.1.2. **Poser** $Topt:=T_{|H_j|,j}$. *On note par* $Topt$ *la date induisant le coût le plus bas.*

5. **Reconstitution de la solution.**

5.1. **Si** ($Copt$ est inchangé) **alors** : *Le coût* $Copt$ *est toujours égal à* $+\infty$.

5.1.1. **Quitter**. *Il n'existe aucune solution réalisable.*

5.2. **Sinon** :

5.2.1. **Poser** $T_{|H_j|,j}:=Topt$.

5.2.2. **Pour** h allant de $|H_j|$ à 2 : *On note* h *l'indice de l'étape considérée.*

5.2.2.1. **Pour** $f\in\{f<h \ / \ (f,h)\in I_j\}$: *On note* f *l'indice de l'étape considérée.*

5.2.2.1.1. **Poser** $T_{f,j}:=Sopt_{(f,h)}(T_{h,j})$.

5.2.2.1.2. **Poser** $Z_{(f,h),j}:=\min\{z \ / \ 1\leq z\leq y_{(f,h),j},\ b_{z,(f,h),j}\geq T_{h,j}\}$.

5.2.2.2. **Pour** $k\in\{k<h \ / \ (h,k)\in I_j\}$: *On note* k *l'indice de l'étape considérée.*

5.2.2.2.1. **Poser** $T_{k,j}:=Sopt_{(h,k)}(T_{h,j})$.

5.2.2.2.2. **Poser** $Z_{(h,k),j}:=\max\{z \ / \ 1\leq z\leq y_{(h,k),j},\ a_{z,(h,k),j}\leq T_{h,j}\}$.

5.2.2.3. **Pour** $f\in\{f<h \ / \ (f,h)\in\aleph_{e_{h,j}}(X_{e_{h,j}j})\}$: *On note* f *l'indice de l'étape considérée.*

5.2.2.3.1. **Poser** $T_{f,j}:=T_{h,j}$.

5.2.2.4. **Pour** $k\in\{k<h \ / \ (h,k)\in\Im_j\}$: *On note* k *l'indice de l'étape considérée.*

5.2.2.4.1. **Poser** $T_{k,j}:=T_{h,j}$.

5.2.2.4.2. **Poser** $X_{e_{h,j}j}:=X_{e_{h,j}}(T_{h,j})$.

Si l'Algorithme IV- 1 s'achève au pas 5.1.1, il n'existe aucune solution réalisable. Cela signifie qu'il est impossible de fabriquer le produit avec les contraintes imposées au système de production.

Exemple

Nous proposons un exemple d'application de l'Algorithme IV- 1. Les données relatives au produit j à réaliser sont fournies dans le Tableau IV- 1 et la Figure IV- 6. Le produit peut être réalisé à partir de la date r_j=0 et avant la date $\overline{d}_j$=11, calculée en choisissant arbitrairement deux sous-gammes alternatives.

Tableau IV- 1 : Données fournies

ENTREES					
Indice de l'opération i	Durée minimale $l_{i,j}$	Durée maximale $u_{i,j}$	Coût opératoire $s_{i,j}$	Instant de début et de fin (h,k)	Périodes durant lesquelles on peut réaliser l'opération i $\{[a_{z,i,j};b_{z,i,j}] / 1 \leq z \leq y_{i,j}\}$
1	2	4	4	(2,4)	[0;1], [2;6], [8;+∞[
2	3	5	1	(3,5)	[0;2], [4;7], [11;+∞[
3	1	4	7	(6,7)	[1;5], [6;+∞[
4	2	2	6	(8,10)	[0;6], [9;+∞[
m_j=5	3	5	2	(9,11)	[0;1], [2;5], [8;+∞[

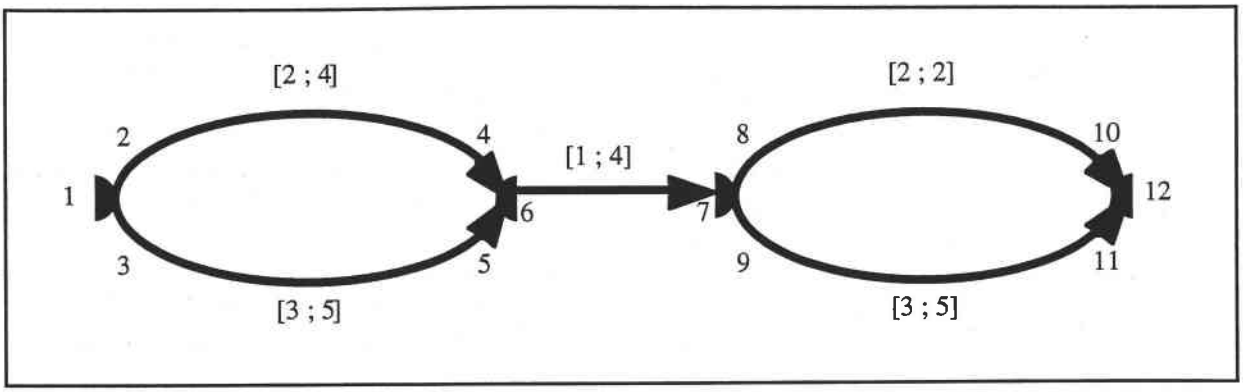

Figure IV- 6 : Représentation du produit à fabriquer

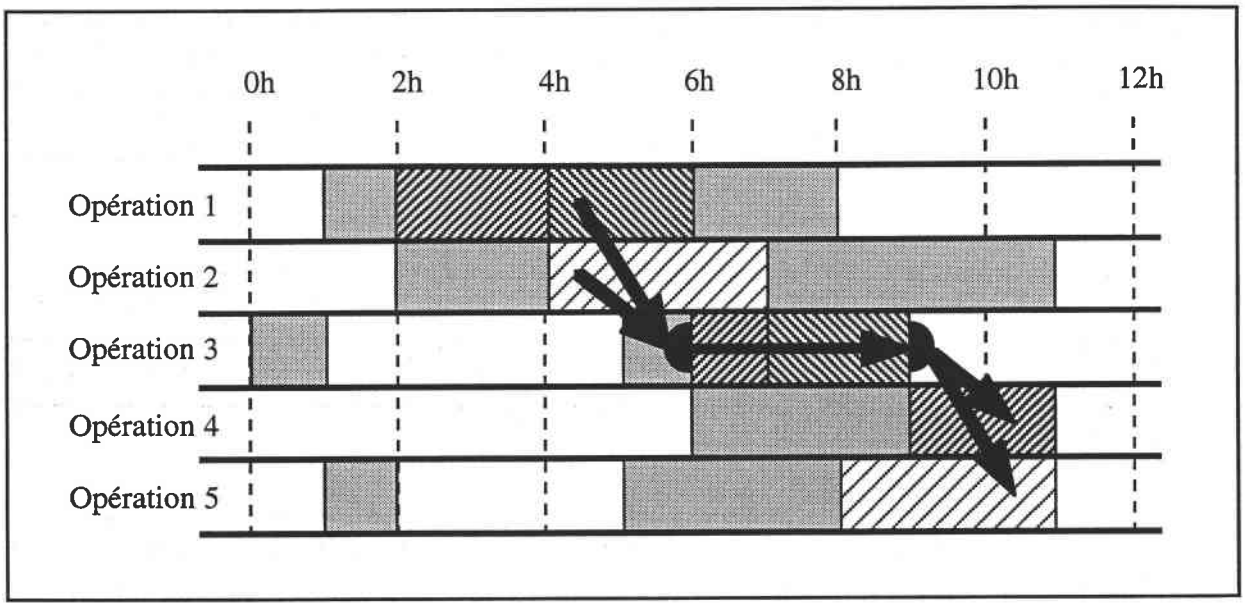

Figure IV- 7 : Résultats fournis par l'Algorithme IV- 1

Les résultats de l'exécution sont donnés dans la Figure IV- 7. Les opérations effectuées sur le produit à ordonnancer sont représentées par des zones dans lesquelles les hachures ont une densité importante, tandis que les opérations correspondant à des sous-gammes non retenues sont représentées par des hachures de faible densité.

Tableau IV- 2 : Résultats de l'exécution de l'Algorithme IV- 1

ENTREE	RESULTATS INTERMEDIAIRES Coûts associés à chaque date												SORTIE
	$T_{1,j}$	$T_{2,j}$	$T_{3,j}$	$T_{4,j}$	$T_{5,j}$	$T_{6,j}$	$T_{7,j}$	$T_{8,j}$	$T_{9,j}$	$T_{10,j}$	$T_{11,j}$	$T_{12,j}$	
r_j=0	0	0	0	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
1	1	1	1	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
2	**2**	**2**	2	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	$T_{1,j}$ $T_{2,j}$
3	3	3	3	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
4	4	4	4	4	+∞	4	+∞	+∞	+∞	+∞	+∞	+∞	
5	5	5	5	5	+∞	5	5	5	5	+∞	+∞	+∞	
6	6	6	6	**6**	+∞	**6**	+∞	+∞	+∞	+∞	+∞	+∞	$T_{4,j}$ $T_{6,j}$
7	7	7	7	+∞	7	7	7	7	7	+∞	+∞	+∞	
8	8	8	8	+∞	+∞	+∞	8	8	8	+∞	+∞	+∞	
9	9	9	9	+∞	+∞	+∞	**9**	**9**	9	+∞	+∞	+∞	$T_{7,j}$ $T_{8,j}$
10	10	10	10	10	+∞	10	10	10	10	+∞	+∞	+∞	
$\overline{d}_j$=11	11	11	11	11	+∞	11	11	11	11	**11**	11	**11**	$T_{10,j}$ $T_{12,j}$

Théorème IV- 2

Si le Problème IV- 1 a une solution réalisable, l'Algorithme IV- 1 détermine les fenêtres et les dates d'exécution des opérations qui permettent de réaliser le produit avec un coût minimal, i.e. la solution fournie par l'Algorithme IV- 1 est optimale pour le Problème IV- 1.

Démonstration

La formulation du Problème IV- 1 est équivalente aux équations de programmation dynamique (IV-10) à (IV-18). Il en découle que l'Algorithme IV- 1 fournit la solution optimale au Problème IV- 1.

CQFD

Complexité

Le principe de l'Algorithme IV- 1 étant similaire à l'Algorithme II- 6, **sa complexité est semblable et égale à** $O(y_j \log_2 y_j + y_j \ |H_j| \ (d_j - r_j + 1) + |H_j| \ (d_j - r_j + 1)^2)$. Rappelons que $|H_j|$ est le nombre de sommets du graphe (i.e. d'états du produit). Il est égal à la somme de m_j, le nombre d'opérations de la gamme et du nombre de gammes alternatives correspondant à l'ensemble de sous-gammes alternatives. Les remarques proposées pour réduire le temps de calcul de l'Algorithme II- 6 sont applicables à l'Algorithme IV- 1.

Place mémoire

Comme l'Algorithme III- 5, nous devons conserver les évaluations des dates de toutes les étapes précédentes. Par conséquent, **l'Algorithme IV- 1 requiert** $(2 \ |H_j|)(d_j - r_j + 1)$ **mots**.

IV.3.4. Conclusion

Dans cette section, la méthode de gestion en temps réel décrite ordonnance chaque produit de manière à minimiser le *makespan*. Cette méthode de gestion basée sur la programmation dynamique permet de choisir la meilleure sous-gamme alternative.

IV.4. MINIMISATION DU TEMPS TOTAL D'UTILISATION DES RESSOURCES

IV.4.1. Introduction

Dès qu'une commande arrive, nous pouvons fournir un délai minimum de fabrication (voir section précédente). A partir de ce délai minimum de fabrication pour l'ensemble des produits, et ainsi de négocier avec le client la date de livraison des produits. Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui conduit au lancement en fabrication au plus tard de chaque produit, tout en respectant la date de livraison fixée au plus tôt par l'Algorithme IV- 1 (voir section précédente). En retardant le plus possible le début de fabrication du produit, nous espérons laisser plus de liberté pour ordonnancer les produits futurs. Nous formalisons ce problème avant de le résoudre par une méthode similaire à celle développée dans la section précédente.

IV.4.2. Formalisation

Les contraintes du problème à résoudre incluent toutes celles du Problème IV- 1. De plus, nous voulons respecter la date de livraison d_j du produit j fixée au plus tôt (voir contrainte (IV-19)).

Notre objectif est de maximiser la date de lancement du produit. Nous notons $H_j \backslash H'_j$ l'ensemble des sommets du graphe qui n'appartiennent pas à une sous-gamme alternative. Nous considérons que le début et la fin d'exécution du produit n'appartiennent pas à une gamme alternative.

Problème IV- 2 : Minimisation du temps total d'utilisation des ressources

$$\max \quad \min_{h \in H_j \backslash H'_j} \{T_{h,j}\}$$

$$\text{tel que} \begin{cases} r_j \le T_{h,j} & \forall h \in H_j & \text{(IV - 1)} \\ r_j \le T_{h,j} & \forall h \in H_j & \text{(IV - 19)} \\ l_{(h,k),j} \le T_{k,j} - T_{h,j} \le u_{(h,k),j} & \forall (h,k) \in I_j & \text{(IV - 2)} \\ a_{Z_{(h,k),j},(h,k),j} \le T_{h,j} & \forall (h,k) \in I_j & \text{(IV - 3)} \\ T_{k,j} \le b_{Z_{(h,k),j},(h,k),j} & \forall (h,k) \in I_j & \text{(IV - 4)} \\ 1 \le Z_{i,j} \le y_{i,j} & \forall i \in I_j & \text{(IV - 5)} \\ 1 \le X_{e,j} \le v_{e,j} & \forall e \in E_j & \text{(IV - 6)} \\ T_{h,j} = T_{k,j} \quad \forall (h,k) \in \aleph_{e,j}(X_{e,j}) & \forall e \in E_j & \text{(IV - 7)} \\ T_{h,j} \in \Re^+ & \forall h \in H_j & \text{(IV - 8)} \\ Z_{i,j} \in \mathrm{N} & \forall i \in I_j & \text{(IV - 9)} \\ X_{e,j} \in \mathrm{N} & \forall e \in E_j & \text{(IV - 10)} \end{cases}$$

Théorème IV- 3

Trouver une solution réalisable au Problème IV- 2 est un problème NP-complet et le Problème IV- 2 est un problème NP-difficile.

Démonstration

D'après le Théorème IV- 1, trouver une solution vérifiant les contraintes (IV-1) à (IV-10) est un problème NP-complet. Nous en déduisons que trouver une solution réalisable au Problème IV- 2 est un problème NP-complet et le Problème IV- 2 est un problème NP-difficile.

CQFD

IV.4.3. Sélection des fenêtres

Nous proposons une approche directe de résolution du Problème IV- 2 en un temps pseudo polynomial. Cette approche est similaire à celle employée pour résoudre le Problème IV- 1. Aussi, nous ne mentionnons que ce qui diffère dans l'approche que nous proposons. Nous considérons également que les durées des opérations ainsi que les dates de début et de fin des fenêtres sont des valeurs rationnelles.

De la même manière, nous définissons $|H_j|$ étapes correspondant aux $|H_j|$ états du produit j au cours de sa réalisation. Pour chaque état du produit, nous énumérons toutes les dates possibles pour lesquelles le produit peut être dans cet état. Nous recherchons l'ensemble des dates de chaque état qui correspondent à un commencement au plus tard du produit et, par conséquent, nous évaluons chaque date en fonction de la date de lancement qu'elle permet d'atteindre.

Dans les définitions (IV-11), (IV-12), (IV-16) et (IV-17), nous remplaçons $\overline{d}_j$ par d_j la date de fin au plus tôt calculée dans la section précédente.

Nous désignons par $C_{(f,h)}(T_{h,j})$ la date de lancement au plus tard induite de l'opération $(f,h)\in I_j$ se terminant à la date $T_{h,j}$:

$$C_{(f,h)}\left(T_{h,j}\right)=\begin{cases}-\infty & \text{si } S_{(f,h)}\left(T_{h,j}\right)=\varnothing \\ \max\limits_{T_{f,j}\in S_{(f,h)}\left(T_{h,j}\right)}\left\{\min\left\{C\left(T_{f,j}\right);T_{h,j}\right\}\right\} & \text{sinon}\end{cases}$$

pour $r_j \le T_{h,j} \le d_j$, $(h,k)\in I_j$, $1\le h\le m_j+1$. (IV-20)

De même, nous désignons par $C_{(h,k)}(T_{h,j})$ la date de lancement au plus tard induite par l'opération $(h,k)\in I_j$ commençant à la date $T_{h,j}$:

$$C_{(h,k)}(T_{h,j}) = \begin{cases} -\infty & \text{si } S_{(h,k)}(T_{h,j}) = \varnothing \\ \max\limits_{T_{k,j} \in S_{(h,k)}(T_{h,j})} \left\{ \min\left\{ C(T_{k,j}); T_{h,j} \right\} \right\} & \text{sinon} \end{cases}$$

pour $r_j \le T_{h,j} \le d_j$, $(h,k) \in I_j$, $1 \le h \le m_j + 1$. (IV-21)

Parmi tous les successeurs possibles k de h où $(h,k) \in \mathfrak{I}_j$, nous retenons celui qui nous permet de commencer le plus tard possible et que l'état h du produit soit obtenu à la date $T_{h,j}$. Nous en déduisons que la date de lancement au plus tard pour lequel l'état h est obtenu à la date $T_{h,j}$ est :

$$C(T_{h,j}) = \min\left\{ \min_{\substack{(h,k) \in I_j \\ k<h}} \left\{ C_{(h,k)}(T_{h,j}) \right\}; \min_{\substack{(f,h) \in I_j \cup \mathfrak{I}_j \\ f<h}} \left\{ C_{(f,h)}(T_{h,j}) \right\}; \max_{\substack{(h,k) \in \mathfrak{I}_j \\ k<h}} \left\{ C_{(h,k)}(T_{h,j}) \right\} \right\}$$

pour $r_j \le T_{h,j} \le d_j$, $1 \le h \le m_j + 1$. (IV-22)

Si la date $T_{h,j}$ ne correspond pas à une solution réalisable, il lui est associé une date de lancement négative (et infinie). L'évaluation $C(T_{|H_j|,j})$ donne la date de lancement maximale pour lequel le produit est dans l'état $|H_j|$ à la date $T_{|H_j|,j}$. Si $\{T_{h,j}\}_{1 \le h \le m_j+1}$ est la solution optimale au Problème IV- 2, on a $\max\limits_{h \in H_j}\{T_{h,j}\} = \min\limits_{r_j \le T_{|H_j|,j} \le \bar{d}_j} \left\{ C(T_{|H_j|,j}) \right\}$. L'Algorithme IV-2 découle des relations (IV-11) à (IV-12), (IV-14), (IV-16) à (IV-17), et (IV-20) à (IV-22) et permet de résoudre le Problème IV- 2.

Algorithme IV- 2 : Choix des fenêtres pour un lancement au plus tard

Il est identique à l'Algorithme IV- 1, mises à part les modifications suivantes :

a) La borne $\bar{d}_j$ est remplacée par d_j.

b) Aux pas 2.1.2.1, 2.1.3.1 et 3, nous initialisons les coûts $C_{(f,h)}(T_{h,j})$, $C_{(h,k)}(T_{h,j})$ et $Copt$ à $-\infty$ (au lieu de $+\infty$).

c) Aux pas 2.1.2.4, 2.1.3.4, 2.1.6.4 et 4.1, les signes ">" sont remplacés par "<".

d) Aux pas 2.1.2.5, 2.1.3.5, 2.1.4.2 et 2.1.7, les opérateurs "max" sont remplacés par des opérateurs "min".

Exemple

Nous proposons un exemple d'application de l'Algorithme IV- 2. Les données relatives au produit j à réaliser sont fournies dans le Tableau IV- 1 et la Figure IV- 6. Le produit peut être réalisé à partir de la date $r_j = 0$ et avant la date $d_j = 11$.

Tableau IV- 3 : Résultats de l'exécution de l'Algorithme IV- 2

ENTREE	RESULTATS INTERMEDIAIRES Coûts associés à chaque date												SORTIE
	$T_{1,j}$	$T_{2,j}$	$T_{3,j}$	$T_{4,j}$	$T_{5,j}$	$T_{6,j}$	$T_{7,j}$	$T_{8,j}$	$T_{9,j}$	$T_{10,j}$	$T_{11,j}$	$T_{12,j}$	
r_j=0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	
1	1	1	1	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	
2	2	2	2	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	
3	3	3	3	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	
4	**4**	**4**	4	2	-∞	2	-∞	-∞	-∞	-∞	-∞	-∞	$T_{1,j}$ $T_{2,j}$
5	5	5	5	3	-∞	3	2	2	2	-∞	-∞	-∞	
6	6	6	6	**4**	-∞	**4**	-∞	-∞	-∞	-∞	-∞	-∞	$T_{4,j}$ $T_{6,j}$
7	7	7	7	-∞	4	4	4	4	4	-∞	-∞	-∞	
8	8	8	8	-∞	-∞	-∞	4	4	4	-∞	-∞	-∞	
9	9	9	9	-∞	-∞	-∞	**4**	**4**	4	-∞	-∞	-∞	$T_{7,j}$ $T_{8,j}$
10	10	10	10	8	-∞	8	4	4	4	-∞	-∞	-∞	
d_j=11	11	11	11	9	-∞	9	4	4	4	**4**	4	**4**	$T_{10,j}$ $T_{12,j}$

Les résultats de l'exécution sont donnés dans la Figure IV- 8 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Complexité

La complexité de l'Algorithme IV- 2 est égale à celle de l'Algorithme IV- 1, c'est-à-dire en $O(y_j \log_2 y_j + y_j \ |H_j| \ (d_j - r_j + 1) + |H_j| \ (d_j - r_j + 1)^2)$.

Place mémoire

La place mémoire nécessaire à l'Algorithme IV- 2 est égale à celle de l'Algorithme IV- 1, c'est-à-dire que l'Algorithme IV- 2 requiert $(2 \ |H_j|)(d_j - r_j + 1)$ mots.

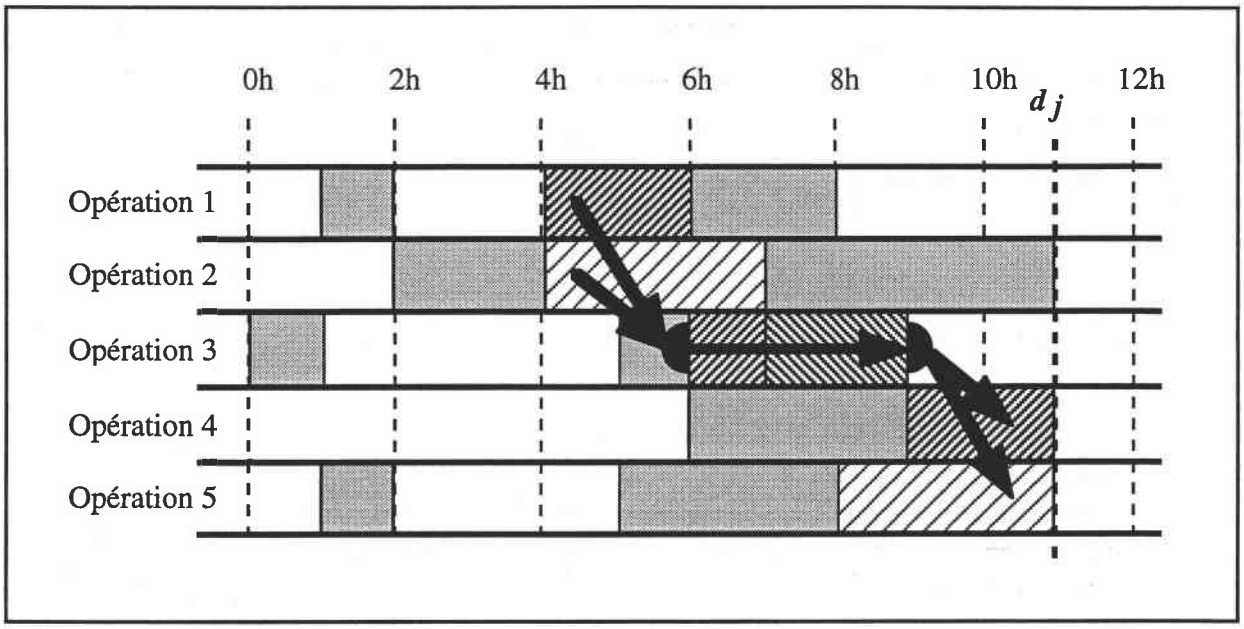

Figure IV- 8 : Résultats fournis par l'Algorithme IV- 2

Théorème IV- 4

Si le Problème IV- 2 a une solution réalisable, l'Algorithme IV- 2 détermine les fenêtres et les dates d'exécution des opérations qui permettent de réaliser le produit avec un coût minimal, i.e. la solution fournie par l'Algorithme IV- 2 est optimale pour le Problème IV- 2.

Démonstration

La formulation du Problème IV- 2 est équivalente aux équations de programmation dynamique (IV-11) à (IV-12), (IV-14), (IV-16) à (IV-17), et (IV-20) à (IV-22). Il en découle que l'Algorithme IV- 2 fournit la solution optimale au Problème IV- 2.

CQFD

IV.4.4. Conclusion

Dans cette section, la méthode de gestion en temps réel décrite ordonnance chaque produit de manière à commencer le produit au plus tard. Cette méthode de gestion basée sur la programmation dynamique permet de choisir la meilleure sous-gamme alternative.

IV.5. MINIMISATION DU COUT D'UTILISATION DES RESSOURCES

IV.5.1.Introduction

Dès qu'une commande arrive, nous pouvons fournir un délai minimum de fabrication (voir section IV.3), et ainsi négocier avec le client la date de livraison des produits. Dans cette section, notre objectif est de décrire une méthode de gestion en temps réel qui fabrique chaque produit au coût le plus bas, tout en respectant la date de livraison fixée au plus tôt par l'Algorithme IV- 1.

Nous formalisons ce problème avant de le résoudre par une méthode similaire à celle développée dans les sections précédentes.

IV.5.2.Formalisation

Les contraintes du problème à résoudre sont identiques à celles du Problème IV- 2. Nous voulons déterminer les dates de début $T_{h,j}$ et de fin $T_{k,j}$ de chaque opération $(h,k)\in I_j$ exécutée sur le produit j. Notre objectif est de minimiser le coût d'utilisation des ressources sachant que le coût induit en fabrication par l'opération i est $s_{i,j}$. Nous notons $I_j\{X_{e,j}\}$ l'ensemble des opérations de la gamme exécutées si les sous-gammes alternatives $\{X_{e,j}\}_{e\in E_j}$ sont retenues.

Problème IV- 3 :Minimisation du coût d'utilisation des ressources

$$\min \sum_{(h,k)\in I_j\{X_{e,j}\}} s_{(h,k),j}\left(T_{k,j}-T_{h,j}\right)$$

$$\text{tel que}\begin{cases}
r_j \le T_{h,j} & \forall h\in H_j & \text{(IV-1)}\\
r_j \le T_{h,j} & \forall h\in H_j & \text{(IV-19)}\\
l_{(h,k),j} \le T_{k,j}-T_{h,j} \le u_{(h,k),j} & \forall (h,k)\in I_j & \text{(IV-2)}\\
a_{Z_{(h,k),j},(h,k),j} \le T_{h,j} & \forall (h,k)\in I_j & \text{(IV-3)}\\
T_{k,j} \le b_{Z_{(h,k),j},(h,k),j} & \forall (h,k)\in I_j & \text{(IV-4)}\\
1\le Z_{i,j}\le y_{i,j} & \forall i\in I_j & \text{(IV-5)}\\
1\le X_{e,j}\le v_{e,j} & \forall e\in E_j & \text{(IV-6)}\\
T_{h,j}=T_{k,j} \quad \forall (h,k)\in \aleph_{e,j}\left(X_{e,j}\right) & \forall e\in E_j & \text{(IV-7)}\\
T_{h,j}\in \Re^{+} & \forall h\in H_j & \text{(IV-8)}\\
Z_{i,j}\in \mathrm{N} & \forall i\in I_j & \text{(IV-9)}\\
X_{e,j}\in \mathrm{N} & \forall e\in E_j & \text{(IV-10)}
\end{cases}$$

Théorème IV- 5

Trouver une solution réalisable au Problème IV- 3 est un problème NP-complet, et le Problème IV- 3 est un problème NP-difficile.

Démonstration

D'après le Théorème IV- 3, trouver une solution réalisable au Problème IV- 3 est un problème NP-complet, et le Problème IV- 3 est un problème NP-difficile.

CQFD

IV.5.3. Sélection des fenêtres

Nous proposons une approche directe de résolution du Problème IV- 3 en un temps pseudo polynomial. Cette approche est similaire à celle employée pour résoudre les Problèmes IV- 1 et IV- 2. Aussi, nous ne mentionnons que ce qui diffère par rapport à ces problèmes. Nous considérons également que les durées des opérations ainsi que les dates de début et de fin des fenêtres sont des valeurs rationnelles.

De la même manière, nous définissons $|H_j|$ étapes correspondant aux $|H_j|$ états du produit j au cours de sa réalisation. Pour chaque état du produit, nous énumérons toutes les dates possibles pour lesquelles le produit peut être dans cet état. Nous recherchons l'ensemble des dates de chaque état qui correspondent à un coût minimal de fabrication du produit et, par conséquent, nous évaluons chaque date en fonction du coût d'utilisation des ressources qu'elle induit.

Dans les définitions (IV-11), (IV-12), (IV-16) et (IV-17), nous remplaçons $\bar{d}_j$ par d_j, date de fin au plus tôt calculée dans la section précédente.

Nous désignons par $C_{(f,h)}(T_{h,j})$ le coût d'utilisation des ressources induit par l'opération $(f,h)\in I_j$ se terminant à la date $T_{h,j}$:

$$C_{(f,h)}\left(T_{h,j}\right)=\begin{cases}+\infty & \text{si } S_{(f,h)}\left(T_{h,j}\right)=\varnothing \\ \min\limits_{T_{f,j}\in S_{(f,h)}\left(T_{h,j}\right)}\left\{C\left(T_{f,j}\right)+s_{(f,h),j}\left(T_{h,j}-T_{f,j}\right)\right\} & \text{sinon}\end{cases}$$

pour $r_j \le T_{h,j} \le d_j$, $(h,k)\in I_j$, $1\le h\le m_j+1$. (IV-23)

De même, nous désignons par $C_{(h,k)}(T_{h,j})$ le coût d'utilisation des ressources induit par l'opération $(h,k)\in I_j$ commençant à la date $T_{h,j}$:

$$C_{(h,k)}(T_{h,j}) = \begin{cases} +\infty & \text{si } S_{(h,k)}(T_{h,j}) = \varnothing \\ \min\limits_{T_{k,j} \in S_{(h,k)}(T_{h,j})} \left\{ C(T_{k,j}) + s_{(h,k),j}(T_{k,j} - T_{h,j}) \right\} & \text{sinon} \end{cases}$$

pour $r_j \le T_{h,j} \le d_j$, $(h,k) \in I_j$, $1 \le h \le m_j + 1$. (IV-24)

Parmi tous les successeurs possibles k de h où $(h,k) \in \mathfrak{I}_j$, , nous retenons celui qui nous permet de commencer le plus tard possible et tel que l'état h du produit soit obtenu à la date $T_{h,j}$. Nous en déduisons que la date de commencement au plus tard pour lequel l'état h est obtenu à la date $T_{h,j}$ est :

$$C(T_{h,j}) = \sum_{\substack{(h,k) \in I_j \\ k<h}} C_{(h,k)}(T_{h,j}) + \sum_{\substack{(f,h) \in I_j \cup \mathfrak{I}_j \\ f<h}} C_{(f,h)}(T_{h,j}) + \min_{\substack{(h,k) \in \mathfrak{I}_j \\ k<h}} \left\{ C_{(h,k)}(T_{h,j}) \right\}$$

pour $r_j \le T_{h,j} \le d_j$, $1 \le h \le m_j + 1$. (IV-25)

Si la date $T_{h,j}$ ne correspond pas à une solution réalisable, il lui est associée un coût d'utilisation infini. L'évaluation $C(T_{|H_j|,j})$ donne le coût d'utilisation minimal pour lequel le produit est dans l'état $|H_j|$ à la date $T_{|H_j|,j}$. L'Algorithme IV- 3 découle des relations (IV-11) à (IV-12), (IV-14), (IV-16) à (IV-17), et (IV-23) à (IV-25) et permet de résoudre le Problème IV- 3.

Algorithme IV- 3 : Choix des fenêtres pour un coût d'utilisation minimal

Il est identique à l'Algorithme IV- 1, mises à part les modifications suivantes :

a) La borne $\overline{d}_j$ est remplacée par d_j.

b) Aux pas 2.1.1, nous initialisons le coût $C(T_{h,j})$ à 0.

c) Aux pas 2.1.2.4, 2.1.3.4, 2.1.2.4.1 et 2.1.3.4.1, les signes "$C(T_{f,j})$" et "$C(T_{k,j})$" sont remplacés par respectivement "$C(T_{f,j})+(T_{h,j}-T_{f,j})\ s_{(f,h),j}$" et "$C(T_{k,j})+(T_{k,j}-T_{h,j})\ s_{(k,h),j}$" (comme pour les pas 2.1.2.4 et 2.1.3.4 de l'Algorithme III- 5).

d) Aux pas 2.1.2.5, 2.1.3.5, 2.1.4.2 et 2.1.7, les opérateurs "max" sont remplacés par des opérateurs "somme".

Exemple

Nous proposons un exemple d'application de l'Algorithme IV- 3. Les données relatives au produit j à réaliser sont fournies dans le Tableau IV- 1 et la Figure IV- 6. Le produit peut être réalisé à partir de la date r_j=0 et avant la date d_j=7.

Tableau IV- 4 :Résultats de l'exécution de l'Algorithme IV- 3

ENTREE	RESULTATS INTERMEDIAIRES Coûts associés à chaque date												SORTIE
	$T_{1,j}$	$T_{2,j}$	$T_{3,j}$	$T_{4,j}$	$T_{5,j}$	$T_{6,j}$	$T_{7,j}$	$T_{8,j}$	$T_{9,j}$	$T_{10,j}$	$T_{11,j}$	$T_{12,j}$	
r_j=0	0	0	0	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
1	0	0	0	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
2	0	0	0	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
3	0	0	0	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	+∞	
4	**0**	0	**0**	8	+∞	8	+∞	+∞	+∞	+∞	+∞	+∞	$T_{1,j}$ $T_{3,j}$
5	0	0	0	8	+∞	8	15	15	15	+∞	+∞	+∞	
6	0	0	0	8	+∞	8	+∞	+∞	+∞	+∞	+∞	+∞	
7	0	0	0	+∞	**3**	**3**	15	15	15	+∞	+∞	+∞	$T_{5,j}$ $T_{6,j}$
8	0	0	0	+∞	+∞	+∞	**10**	10	**10**	+∞	+∞	+∞	$T_{7,j}$ $T_{9,j}$
9	0	0	0	+∞	+∞	+∞	17	17	17	+∞	+∞	+∞	
10	0	0	0	8	+∞	8	24	24	24	+∞	+∞	+∞	
d_j=11	0	0	0	8	+∞	8	10	10	10	19	**16**	**16**	$T_{11,j}$ $T_{12,j}$

Les résultats de l'exécution sont donnés dans la Figure IV- 9 où les opérations effectuées sur le produit à ordonnancer sont représentées par des zones hachurées.

Complexité

La complexité de l'Algorithme IV- 3 est égale à celle de l'Algorithme IV- 1, c'est-à-dire en $O(y_j \log_2 y_j + y_j \ |H_j| \ (d_j - r_j + 1) + |H_j| \ (d_j - r_j + 1)^2)$.

Place mémoire

La place mémoire nécessaire à l'Algorithme IV- 3 est égale à celle de l'Algorithme IV- 1, c'est-à-dire que l'Algorithme IV- 3 requiert $(2 \ |H_j|)(d_j - r_j + 1)$ mots.

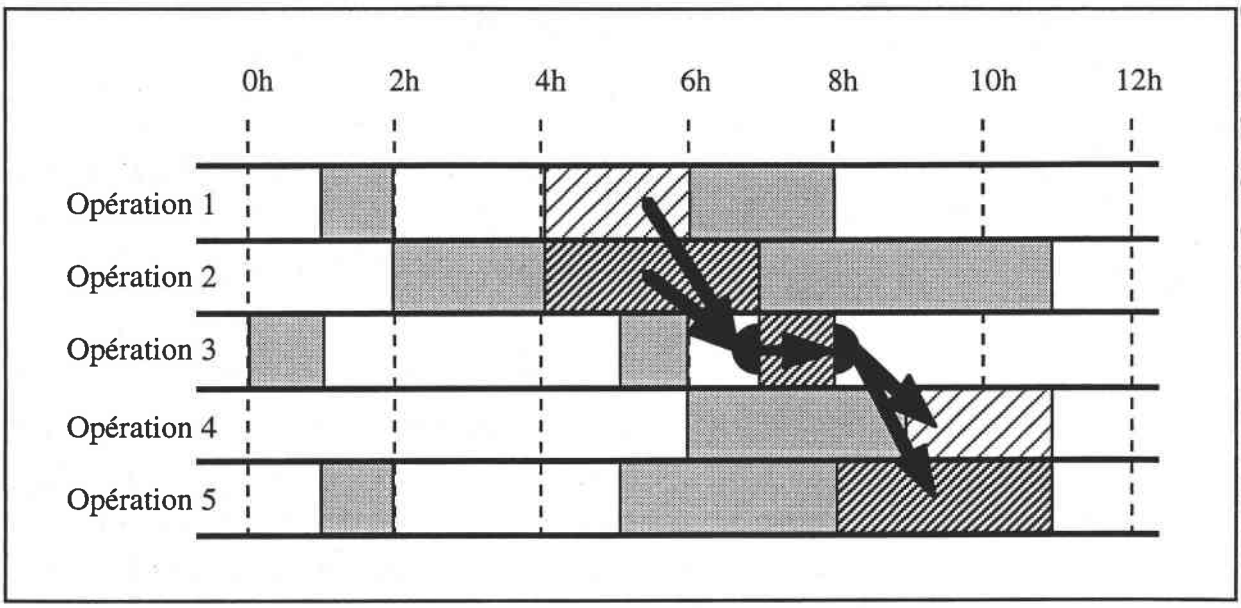

Figure IV- 9 : Résultats fournis par l'Algorithme IV- 3

Théorème IV- 6

Si le Problème IV- 3 a une solution réalisable, l'Algorithme IV- 3 détermine les fenêtres et les dates d'exécution des opérations qui permettent de réaliser le produit avec un coût minimal, i.e. la solution fournie par l'Algorithme IV- 3 est optimale pour le Problème IV- 3.

Démonstration

La formulation du Problème IV- 3 est équivalente aux équations de programmation dynamique (IV-11) à (IV-12), (IV-14), (IV-16) à (IV-17), et (IV-23) à (IV-25). Il en découle que l'Algorithme IV- 3 fournit la solution optimale au Problème IV- 3.

CQFD

IV.5.4. Conclusion

Dans cette section, la méthode de gestion en temps réel décrite ordonnance chaque produit de manière à commencer au plus tard le produit. Cette méthode de gestion basée sur la programmation dynamique permet de choisir la meilleure sous-gamme alternative.

IV.6. CONCLUSION

Dans ce chapitre, nous avons étendu les méthodes de gestion de production en temps réel proposées dans les chapitres précédents. Ces méthodes ont été adaptées aux situations où plusieurs sous-gammes alternatives permettent de passer d'un état du produit à un autre. Elles restent applicables pour les problèmes à durées opératoires mixtes.

La première méthode de gestion consiste **à terminer chaque produit le plus tôt possible** (voir Algorithme IV- 1). La deuxième méthode proposée revient **à fabriquer le produit au plus tard**, tout en respectant la date de livraison fixée au plus tôt (voir Algorithme IV- 2). L'intérêt est de réduire le plus possible le temps pendant lequel des ressources sont utilisées pour fabriquer le produit. Nous proposons une troisième méthode pour laquelle chaque produit est fabriqué de manière à **minimiser le coût d'utilisation des ressources de chaque produit**, sans dépasser la date de livraison fixée au plus tôt (voir Algorithme IV- 3).

Les algorithmes présentés ici sont nouveaux et leur complexité très réduite permet leur utilisation en temps réel.

CONCLUSION

Cette thèse s'intéresse à **l'ordonnancement en temps réel**. Dans ce contexte, les produits sont ordonnancés dès que leur commande est connue, sans attendre l'arrivée des futures demandes et sans remettre en question les produits déjà ordonnancés. Ce type de gestion est très répandu dans les entreprises parce qu'il permet, en particulier, de prendre les décisions rapidement face aux aléas de l'environnement.

Les méthodes de gestion présentées dans cette thèse concernent les systèmes de production pour lesquels les opérations réalisées sur les produits se suivent sans attente, et les durées opératoires sont contrôlables. Dans le premier chapitre, nous avons montré que ces deux caractéristiques sont présentes dans la plupart des systèmes de fabrication : ateliers de traitements de surface, entreprises manufacturières, usines automatisées... De plus, leur prise en compte offre de nouvelles possibilités de gestion : le **contrôle de la quantité** et de la **durée de séjour** des produits en cours de fabrication.

Dans les Chapitres II à IV, nous avons proposé des méthodes de gestion applicables aux systèmes de fabrication pour lesquels les produits ont des gammes linéaires (voir Chapitre II), des gammes d'assemblage et désassemblage (voir Chapitre III), des gammes alternatives (voir Chapitre IV). Les objectifs visés par ces méthodes sont les suivants :

fabriquer chaque produit en minimisant son *makespan*,

fabriquer chaque produit en minimisant le temps d'utilisation des ressources et en respectant la date de livraison,

fabriquer chaque produit au coût le plus bas tout en respectant la date de livraison.

La société SODETEG T.A.I. a déjà utilisé avec succès certains algorithmes pour le cas de produits linéaires. L'efficacité des méthodes proposées et la richesse de situations où elles sont utilisables, devraient contribuer à leur diffusion rapide dans le monde industriel.

Sur le plan théorique, l'analyse de la complexité des algorithmes proposés montre leur adéquation avec leur utilisation en temps réel. Ils sont polynomiaux en ce qui concerne les objectifs a) et b) pour les produits dont les gammes sont linéaires ou comportent des opérations d'assemblage et désassemblage. Dans les autres cas, ils sont pseudo polynomiaux. Nous avons également montré que les trois objectifs conduisent à des problèmes NP-difficiles pour des produits à gammes alternatives. Une question reste ouverte : **minimiser le coût de fabrication d'un produit** (dont la gamme est linéaire ou comporte des opérations d'assemblage et désassemblage) **est-il un problème NP-difficile** ?

Des améliorations des algorithmes pseudo polynomiaux proposées dans cette thèse sont à l'étude. Elles permettront de les rendre encore plus efficaces. Une des méthodes développées pour cela consiste en des raffinements successifs de la solution courante (voir l'étude de la complexité de l'Algorithme II- 6).

Nous avons également comparé la **performance dans le pire des cas** des méthodes en temps réel proposées avec la situation *off-line* où toutes les commandes futures sont connues. Nous avons montré en particulier que ces méthodes sont m -compétitives pour les problèmes de type *flow shop* et *job shop* à m machines. Ces résultats obtenus pour les produits à gammes linéaires peuvent être facilement étendus aux autres produits. Aucune étude ne permet encore de dire si des algorithmes plus compétitifs existent. Nous voulons améliorer les algorithmes proposés pour les situations où les durées des opérations réalisées à l'aide d'une même machine diffèrent d'un produit à l'autre. En particulier, nous espérons développer des algorithmes aléatoires (i.e. qui font intervenir le hasard dans la décision) à partir de l'Algorithme II- 6.

Nous avons aussi étudié les situations où des **conflits de ressources** existent pour un même produit, où les **commandes futures sont connues**, et où il est possible de **modifier les ordonnancements antérieurs**. Dans ces situations, nous avons montré qu'atteindre un des objectifs a), b) ou c) est un problème NP-difficile au sens fort. On ne connaît aucun algorithme polynomial (ou même pseudo polynomial) capable de résoudre ce type de problèmes. Aussi, dans le contexte en temps réel, nous sommes contraints d'appliquer des méthodes approchées rapides. Les méthodes présentées dans cette thèse peuvent évidemment être utilisées comme telles. Nous pensons développer des méthodes spécifiques à certaines situations.

Enfin, de nombreuses méthodes d'ordonnancement *off-line* (i.e. contexte où il est possible de remettre en question les produits déjà ordonnancés) utilisent les méthodes de recherche de plus long chemin sur le graphe PERT développé en gestion de projet. Ainsi, on dit parfois que la gestion de projet est le problème central de l'ordonnancement [Esquirol et Lopez, 1999]. De même, nous pensons que les approches proposées dans cette thèse seront utilisées pour résoudre des problèmes "centraux" de l'ordonnancement *off-line* pour lesquels les opérations réalisées sur les produits se suivent sans attente, et les durées opératoires sont contrôlables.

REFERENCES BIBLIOGRAPHIQUES

A

[Achugbue et Chin, 1982] ACHUGBUE J.O., CHIN F.Y. (1982), "Scheduling the open shop to minimize mean flow time", *SIAM Journal on Computing*, Vol. 11, pp. 709-720.

[Adiri et Amit, 1984] ADIRI I., AMIT N. (1984), "Openshop and flowshop scheduling to minimize sum of completion times", *Computer and Operations Research*, Vol. 11, pp. 275-284.

[Albers et Brucker, 1993] ALBERS S., BRUCKER P. (1993), "The complexity of one-machine batching problems", *Discrete Applied Mathematics*, Vol. 47, pp. 87-107.

[Antonio et *al.*, 1999] ANTONIO A., CHAUVET F., CHU C., PROTH J.-M. (1999), "The cutting stock problem with mixed objectives: two heuristics based on dynamic programming", *European Journal of Operational Research*, Vol. 114, No. 2, pp. 395-402.

B

[Baker, 1974] BAKER K.R. (1974), "Introduction of sequencing and scheduling", Wiley, New York, Etats-Unis.

[Beauquier et *al.*, 1992] BEAUQUIER D., BERSYEL J., CHRETIENNE P. (1992), "Eléments d'algorithmique", Masson, Paris, France.

[Bellman, 1958] BELLMAN R.E. (1958), "On a routing problem", *Quarterly Applied Mathematics*, Vol. 16, pp. 87-90.

[Blackstone et *al.*, 1982] BLACKSTONE J.H. Jr, PHILIPPS D.T., HOGG D.L. (1982), "A state-of-the-art survey of dispatching rules for manufacturing job-shop operations", *International Journal of Production Research*, Vol. 20, pp. 27-45.

[Ben-David et *al.*, 1994] BEN-DAVID S., BORODIN A., KARP R.M., TARDOS G., WIGDERSON A. (1994), " On the power of randomization in on-line algorithms ", *Algorithmica*, Vol. 11, pp. 2-14.

[Blazewicz, 1987] BLAZEWICZ J. (1987), "Selected topics in scheduling theory", *Annals of Discrete Mathematics*, Vol. 31, pp. 1-60.

[Blazewicz et *al.*, 1983] BLAZEWICZ J., LENSTRA J.K., RINNOOY KAN A.H.G. (1983), "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol. 475, pp. 11-24.

[Blazewicz et *al.*, 1993] BLAZEWICZ J., ECKER K., SCHMIDT G., WEGLARZ J. (1993), "Scheduling in computer and manufacturing systems", Springer-Verlag, Berlin, Allemagne.

[Blazewicz et *al.*, 1996] BLAZEWICZ J., DOMSCHKE W., PESCH E. (1996), "The job shop scheduling problem: conventional and new solution techniques", *European Journal of Operational Research*, Vol. 93, pp. 1-33.

[Bloch, 1999] BLOCH C. (1999), "Contribution à l'ordonnancement dynamique de lignes de traitement de surface", Thèse de doctorat, Université de Besançon, France.

[Bolignano et *al.*, 1983] BOLIGNANO D., PROTH J.-M., VOYIATZIS K. (1983), "Ordonnancement des pièces dans un système de traitement de surface", Rapport de recherche INRIA, No. 234, Le Chesnay, France.

[Brauner et *al.*, 1997] BRAUNER N., FINKE G., KUBIAK W. (1997), "A proof of the Lei and Wang claim", Technical Note, Laboratoire Leibniz, Institut Imag, Grenoble, France.

[Brucker, 1995] BRUCKER P. (1995), "Scheduling algorithms", Springer-Verlag, Berlin, Allemagne.

[Brucker et Knust, 1998] BRUCKER P., KNUST S. (1998), "Complexity results for single-machine problems with positive finish-start time-lags", Osnabruecker Schriften zur Mathematik, Reihe P, No. 202.

[Brucker et *al.*, 1977] BRUCKER P., GAREY M.R., JOHNSON (1977), "Scheduling equal-length tasks under tree-like precedence constraints to minimize maximum lateness", *Mathematics of Operations Research*, Vol. 2, pp. 275-284.

[Brucker et *al.*, 1993] BRUCKER P., JURISCH B., JURISCH M. (1993), "Open shop problems with unit time operations", *Zeitschrift fuer of Operations Research*, Vol. 37, pp. 59-73.

[Brucker et *al.*, 1995] BRUCKER P., KRAVCHENKO S.A., SOTSKOV Y.N. (1995), "On the complexity of two machine job-shop scheduling with regular objective functions", Osnabruecker Schriften zur Mathematik, Reihe P, No. 172.

[Brucker et *al.*, 1997] BRUCKER P., JURISCH B., KRAEMER A. (1997), "Complexity of scheduling problems with multi-purpose machines", *Annals of Operations Research*, Vol. 70, pp. 57-73.

[Bruno et *al.*, 1980] BRUNO J., JONES J.W., SO K. (1980), " Deterministic scheduling with pipelined processors", *IEEE Transactions on Computers*, Vol. C-29, pp. 308-316.

C

[Callahan, 1971] CALLAHAN J.R. (1971), "The nothing hot delay problems in the production of steel", PhD. Thesis, Department of Industrial Engineering, University of Toronto, Canada.

[Carlier et Chrétienne, 1988] CARLIER J., CHRETIENNE P. (1988), "Les problèmes d'ordonnancement", Masson, Paris, France.

[Chauvet et Proth, 1998a] CHAUVET F., PROTH J.-M. (1998), "Real-time scheduling for assembly processes", papier invité, *SAE Aerospace Manufacturing Technology Conference & Exposition* (AMTC'98), Long Beach, Californie, Etats-Unis, n°981871, publié par Society of Automative Engineers, Inc., Warrendale, Michigan, Etats-Unis, pp. 187-195.

[Chauvet et Proth, 1998b] CHAUVET F., PROTH J.-M. (1998), "Scheduling heuristics for minimizing the makespan", papier invité, 9th SIAM-Discrete Mathematics'98, Totonto, Canada.

[Chauvet et Proth, 1999] CHAUVET F., PROTH J.-M. (1999) "The PERT problem with alternatives: modelisation and optimisation", Rapport de Recherche I.N.R.I.A. n°3651, INRIA, Le Chesnay (France), mars 1999, proposé pour publication dans *Operations Research.*

[Chauvet et *al.*, 1998a] CHAUVET F., LEVNER E., PROTH J.-M. (1998) "On PERT networks with alternatives", Rapport de Recherche I.N.R.I.A. n°3583, INRIA, Le Chesnay (France), décembre 1998, proposé pour publication dans *Networks.*

[Chauvet et *al.*, 1998b] CHAUVET F., PROTH J.-M., WARDI Y. (1998) "On-line scheduling with WIP regulation", Proceedings of the *Rensselaer's International Conference on Agile, Intelligent, and Computer-Integrated Manufacturing*, Troy, New York, Etats-Unis, CD-ROM, proposé pour publication dans *IEEE Transactions on Robotics and Automation.*

[Chauvet et *al.*, 1999a] CHAUVET F., LEVNER E., MEYZIN L., PROTH J.-M. (1999) "On-line scheduling in a surface treatment system", *European Journal of Operational Research*, à paraître.

[Chauvet et *al.*, 1999b] CHAUVET F., OULD HAOUBA A., PROTH J.-M. (1999), "Cutting problem: a fast algorithm for computing a near-optimal solution", *International Journal of Production Economics*, à paraître.

[Chen et Woeginger, 1995] CHEN B., WOEGINGER G.H. (1995), "A study of on-line scheduling two-stage shops", dans DU D.Z., PARDALOS P.M., eds., "Minimax and applications", Kluwer Academic Publishers, pp. 97-107.

[Chen et *al.*, 1996] CHEN H., CHU C., PROTH J.-M. (1996), "Cyclic scheduling of a hoist with time window constraints", *IEEE Transactions on Robotic and Automation*, Vol. 12, No. 6, pp. 144-152.

[Cheng et Smith, 1995] CHENG C.-C., SMITH S.F. (1995), "A constraint-posting framework for scheduling under complex constraints", Proceedings of the symposium on *Emerging Technologies and Factory Automation* (ETFA'95), Paris, France, Vol. 1, pp. 269-280.

[Chrétienne et *al.*, 1995] CHRETIENNE P., COFFMAN E.G., Jr., LENSTRA J.K., LIU Z. (1995), (eds.) "Scheduling theory and its applications", Wiley, New York, Etats-Unis.

[Chu et Proth, 1996] CHU C., PROTH J.M. (1996), "L'ordonnancement et ses applications", Masson, Paris, France.

[Coffman, 1976] COFFMAN E.G.Jr. (1976), "Computer and job shop scheduling theory", Wiley, New York, Etats-Unis.

[Coffman et Graham, 1972] COFFMAN E.G.Jr., GRAHAM R.L. (1972) "Optimal scheduling for two-processor systems", *Acta Informatica*, Vol. 1, pp. 200-213.

[Conway et *al.*, 1967] CONWAY R.N., MAXWELL W.L., MILLER L.W. (1967), "Theory of scheduling", Addison-Wesley, Reading, Massachusetts, Etats-Unis.

[Crama, 1997] CRAMA Y. (1997) "Combinatorial optimization models for production scheduling in automated manufacturing systems", *European Journal of Operational Research*, Vol. 99, pp. 136-153.

[Crama et van de Klundert, 1997] CRAMA Y., VAN DE KLUNDERT J. (1997), "Cyclic scheduling of identical parts in robotic cell", *Operations Research*, Vol. 45, No. 6.

[Crama et *al.*, 1998] CRAMA Y., KATS V., VAN DE KLUNDERT J., LEVNER E. (1998) "Cyclic scheduling in robotic flowshops", Working paper, No. 9802, Gemme, Faculté d'économie, de gestion et de sciences sociales, Liège, Belgique.

D

[Dempster et *al.*, 1982] DEMPSTER M.A.H., LENSTRA J.K., RINNOOY KAN A.H.G. (1982), (eds.) "Deterministic and stochastic scheduling", Reidel, Dordrecht, Pays-Bas.

E

[Egli et Rippin, 1981] EGLI U.M., RIPPIN D.W.T. (1981), "Short-term scheduling for multi-product batch chemicals plants", American Institute of Chemical Engineers Meeting, Houston, Texas, Etats-Unis.

[Esquirol et Lopez, 1999] ESQUIROL P., LOPEZ P. (1999), (eds.) "L'ordonnancement", Economica, Paris, France.

F

[Feldmann et *al.*, 1994] FELDMANN A., SGALL J., TENG S.H. (1994), "Dynamic scheduling on parallel machines", *Theoretical Computer Science*, Vol. 130, No. 1, pp. 49-72.

[Ford, 1956] FORD L.R.Jr (1956), "Network flow theory", The Rand Corporation, pp. 293.

[French, 1982] FRENCH S. (1982), "Sequencing and scheduling: an introduction to the mathematics of the job shop", Ellis Horwood, Chichester, Grande-Bretagne.

G

[Gandadharan et Rajendran, 1993] GANDADHARAN R., RAJENDRAN C. (1993), "Heuristic algorithms for scheduling in the no-wait flowshop", *International Journal of Production Economics*, Vol. 32, No. 3, pp. 285-290.

[Ganesharajah et *al.*, 1998] GANESHARAJAH T., HALL N.G., SRISKANDARAJAH C. (1998), "Design and operational issues in AGV-served manufacturing systems", *Annals of Operations Research*, Vol. 76, pp. 109-154.

[Garey et Johnson, 1977] GAREY M.R., JOHNSON D.S. (1977), " Two-processor scheduling with start-times and deadlines", *SIAM Journal on Computing*, Vol. 6, pp. 416-426.

[Garey et Johnson, 1979] GAREY M.R., JOHNSON D.S. (1979), "Computers and intractability. A guide to the theory of NP-Completeness", W. H. Freeman And Company, New York, Etats-Unis.

[Garey et *al.*, 1976] GAREY M.R., JOHNSON D.S., SETHI R. (1976), "The complexity of flow-shop and job-shop scheduling", *Mathematics of Operations Research*, Vol. 1, pp. 117-129.

[Gaudel et *al.*, 1987] GAUDEL M.C., SORIA M., FROIDEVAUX C. (1987), "Types de données et algorithmes. Volume 2 : recherche, tri, algorithmes sur les graphes", Collection Didactique, D-004, Inria, Rocquencourt, France.

[Ge et Yih, 1995] GE Y., YIH Y. (1995), "Crane scheduling with time windows in circuit board production lines", *International Journal of Production Research*, Vol. 33, No. 5, pp. 1187-1199.

[Gilmore et Gomory, 1964] GILMORE P.C., GOMORY R.E. (1964) "Sequencing a one-state variable machine: a solvable case of the traveling salesman problem", *Operations Research*, Vol. 12, pp. 655-679.

[Gladky, 1997] GLADKY A.A. (1997), "On the complexity of minimizing weighted number of late jobs in unit time open shops", *Discrete Applied Mathematics*, Vol. 74, pp. 197-201.

[Glass et Potts, 1996] GLASS C.A., POTTS C.N. (1996), "A comparison of local search methods for flow shop scheduling", *Annals of Operations Research*, Vol. 63.

[Golenko-Ginzburg et Kats, 1995] GOLENKO-GINZBURG D., KATS V. (1995), "Priority rules in job shop scheduling", Chapitre 9 dans LEVNER E., eds., "Intelligent scheduling of robots and flexible manufacturing systems", Center for Techological Education Holon, pp. 177-190.

[Gondran et Minoux, 1979] GONDRAN M., MINOUX M. (1979), "Graphes et algorithmes", Eyrolles, Paris, France.

[Gonzalez et Sahni, 1976] GONZALEZ T., SAHNI S. (1976), "Open shop scheduling to minimize finish time", *Journal of the Association of Computing Machinery*, Vol. 23, pp. 665-679.

[Gonzalez et Sahni, 1978] GONZALEZ T., SAHNI S. (1978), "Flow shop and job shop schedules: complexity and approximation", *Operations Research*, Vol. 26, pp. 36-52.

[Goyal et Sriskandarajah, 1988] GOYAL S.K., SRISKANDARAJAH C. (1988), "No-wait shop scheduling: computational complexity and approximate algorithms", *Operations Research*, Vol. 25, No. 4, pp. 220-244.

[Graham et *al.*, 1979] GRAHAM R.L., LAWLER E.L., LENSTRA J.K., RINNOOY KAN A.H.G. (1979), "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics*, Vol. 5, pp. 287-326.

[Gupta et *al.*, 1990] GUPTA J.N.D., STRUSEVICH V.A., ZWANAVELD C.M. (1990), "Two-stage no-wait scheduling models with setup and removal times separated", *Computers and Operations Research*, Vol. 24, No. 11, pp. 1025-1056.

H

[Hall et Sriskandarajah, 1996] HALL N.G., SRISKANDARAJAH C. (1996), "A survey of machine scheduling problems with blocking and no-wait in process", *Operations Research*, Vol. 44, pp. 510-525.

[Hall et *al.*, 1997] HALL N.G., KAMOUN H., SRISKANDARAJAH C. (1997), "Scheduling in robotic cells: classification, two and three machine cells", *Operations Research*, Vol. 45, No. 3, pp. 421-439.

[Hartmann et Orlin, 1993] HARTMANN M., ORLIN J.B., (1993), "Finding minimum cost to time ratio cycles with small integer transit times", *Networks*, Vol. 23, pp. 567-574.

[Hefetz et Adiri, 1982] HEFETZ N., ADIRI I., (1982), "An efficient optimal algorithm for the two-machine, unit-time, job-shop, schedule-length problem", *Mathematical Operations Research*, Vol. 113, pp. 354-360.

[Hertz et *al.*, 1996] HERTZ A., MOTTET Y., ROCHAT Y., (1996), "On a scheduling problem in a robotized analytical system", *Discrete Applied Mathematics*, Vol. 65, No. 1-3, pp. 285-318.

[Hohzaki et *al.*, 1996] HOHZAKI R., FUJII S., SANDOH H. (1996), "A shortest path problem on the network with AGV-type time-windows", *Journal of the Operations Research Society of Japan*, Vol. 39, No. 1, pp. 77-87.

[Hoogeveen et *al.*, 1994] HOOGEVEEN J.A., van de VELDE S.L., VELTMAN B. (1994), "Complexity of scheduling multiprocessor tasks with prespecified processor allocations", *Discrete Applied Mathematics*, Vol. 55, pp. 259-272.

[Hu, 1961] HU T.C. (1961), "Parallel sequencing and assembly line problems", *Operations Research*, Vol. 9, pp. 841-848.

J

[Jackson, 1956] JACKSON J.R. (1956) "An extension of Johnson's results on job lot scheduling", *Naval Research Logistics Quarterly*, Vol. 3, pp. 201-203.

[Jain et Meeran, 1999] JAIN A.S., MEERAN S. (1999) "Deterministic job-shop scheduling: past, present and future", *European Journal of Operational Research*, Vol. 113, pp. 390-434.

[Johnson, 1954] JOHNSON S.M. (1954) "Optimal two-and-three-stage production schedules with set-up times included", *Naval Research Logistics Quarterly*, Vol. 1, pp. 61-68.

K

[Karabati et Kouvelis, 1997] KARABATI S., KOUVELIS P. (1997), "Flow-line scheduling problem with controllable processing times", *IIE Transactions*, Vol. 29, pp. 1-14.

[Kawaguchi et Kyan, 1988] KAWAGUCHI T., KYAN S. (1988), "Deterministic scheduling in computer systems: a survey", *Journal of Operations Research Society of Japan*, Vol. 31, No. 2, pp. 190-217.

[Kise et *al.*, 1991] KISE H., SHIOYAMA T., IBARAKI T. (1991), "Automated two-machine flowshop scheduling: a solvable case" *IEEE Transactions*, Vol. 23, No. 1, pp. 10-16.

[Knuth, 1968-1973] KNUTH D.E. (1968-1973), "The art of computer programming", Vol. I-III, Addison-Wesley, Reading, Massachusetts, Etats-Unis.

[Koole, 1996] KOOLE G.M. (1996), "Stochastic scheduling and dynamic programming", *Journal of the Operational Research Society*, Vol. 47, No. 10, pp. 1313.

[Kravchenko, 1998] KRAVCHENKO S.A. (1998), "A polynomial algorithm for a two-machine no-wait job-shop scheduling problem", *European Journal of Operational Research*, Vol. 106, No. 1, pp. 101-107.

[Kubiak, 1989] KUBIAK W. (1989) "A pseudopolynomial algorithm for a two-machine no-wait job shop problem", *European Journal of Operational Research*, Vol. 43, pp. 267-270.

[Kubiak et Timkovsky, 1996] KUBIAK W., TIMKOVSKY V.G. (1996) "A polynomial-time algorithm for total completion time minimization in two-machine job-shop with unit-time operations", *European Journal of Operational Research*, Vol. 94, pp. 310-320.

[Kubiak et *al.*, 1991] KUBIAK W., SRISKANDARAJAH C., ZARAS K. (1991) "A note on the complexity of open shop scheduling problems", *Infor*, Vol. 29, pp. 284-294.

[Kubiak et *al.*, 1995] KUBIAK W., SETHI W., SRISKANDARAJAH C. (1995) "An efficient algorithm for a job shop problem", *Mathematics of Industrial Systems*, Vol. 1, pp. 203-216.

L

[Lageweg et *al.*, 1981] LAGEWEG B.J., LAWLER E.L., LENSTRA J.K., RINNOOY KAN A.H.G. (1981), "Computer aided complexity classification of deterministic scheduling problems", Report BM 138, Centre for Mathematics and Computer Science.

[Lai et *al.*, 1997] LAI T.C., SOTSKOV Y.N., SOTSKOVA N.YU., WERNER F. (1997), "Optimal makespan scheduling with given bounds of processing times", *Mathematical and Computer Modelling*, Vol. 26, No. 3, pp. 67-86.

[Lamothe, 1996] LAMOTHE J. (1996), "Une approche pour l'ordonnancement dynamique de traitement de surface", Thèse de doctorat, Ecole Nationale de l'Aéronautique et de l'Espace, Toulouse, France.

[Lawler, 1976] LAWLER E.L. (1976), "Sequencing to minimize the weighted number of tardy jobs", *Revue Française d'Automatique, d'Informatique et de Recherche Opérationnelle*, Vol. 10, Supplément 5, pp. 27-33.

[Lawler, 1983] LAWLER E.L. (1983), "Recent results in the theory of machine scheduling", A. Bachem, M. Groetschel, B. Korte, (eds.) "Mathematical programming: the state of the art", Bonn, 1982, Springer-Verlag, Berlin, Allemagne, pp. 202-234.

[Lawler et *al.*, 1981] LAWLER E.L., LENSTRA J.K., RINNOOY KAN A.H.G. (1981), "Minimizing maximum lateness in a two-machine open shop", *Mathematics of Operations Research*, Vol. 6, pp. 153-158, Erratum, *Mathematics of Operations Research*, Vol. 7, pp. 635.

[Lawler et *al.*, 1989] LAWLER E.L., LENSTRA J.K., RINNOOY KAN A.H.G., SHMOYS D.B. (1989), "Sequencing and scheduling: algorithms and complexity", *Operations Research and Managment Science*, Vol. 4.

[Lei et Wang, 1989] LEI L., WANG T.-J. (1989), "A proof: the cyclic hoist scheduling problem is NP-complete", Working paper, No. 89-0016, Rutgers University.

[Lenstra et Rinnooy Kan, 1978] LENSTRA J.K., RINNOOY KAN A.H.G. (1978) "Complexity of scheduling under precedence constraints", *Operations Research*, Vol. 26, pp. 22-35.

[Lenstra et Rinnooy Kan, 1979] LENSTRA J.K., RINNOOY KAN A.H.G. (1979), "Computational complexity of discrete optimization problems", *Annals of Discrete Mathematics*, Vol. 4, pp. 121-140.

[Liu et Bulfin, 1985] LIU C.Y., BULFIN R.L. (1985), "On the complexity of preemptive open shop scheduling problems", *Operations Research Letters*, Vol. 4, pp. 71-74.

[Logendra et Sriskandarajah, 1996] LOGENDRA R., SRISKANDARAJAH C. (1996), "Sequencing of Robot Activities and Parts in Two Machine Robotic Cells", *International Journal of Production Research*, à paraître.

M

[Malcom et *al.*, 1959] MALCOLM D.G., ROSEBOOM J.H., CLARK C.E., FAZAR W., (1959), "Application of a technique for research and development program evaluation", *Operations Research*, Vol. 7, No. 5.

[Matsuo, 1990] MATSUO H. (1990), "Cyclic sequencing problems in the two-machine permutation flowshop: Complexity, worst-case and average-case analysis", *Naval Research Logistics*, Vol. 37, pp. 679-694.

[McCormick et Rao, 1994] MCCORMICK S.T., RAO U.S. (1994), "Some complexity results in cyclic scheduling", *Mathematical and Computer Modeling*, Vol. 20, pp. 107-122.

[Monma, 1982] MONMA C.L. (1982), "Linear-time algorithms for scheduling on parallel processors", *Operations Research*, Vol. 30, pp. 792-798.

[Morton et Pentico, 1993] MORTON T.E., PENTICO D. (1993), "Heuristics Scheduling Systems", Wiley, New York, Etats-Unis.

N

[Nowicki, 1994] NOWICKI E. (1994), "An approximation algorithm for a single-machine scheduling problem with release times, delivery times and controllable processing times", *European Journal of Operational Research*, Vol. 72, No. 1, pp. 74-82.

[Nowicki et Zdrzalka, 1990] NOWICKI E., ZDRZALKA S. (1990), "A survey of results for sequencing problems with controllable job processing times", *Discrete Applied Mathematics*, Vol. 26, pp. 271-287.

O

[Orman et *al.*, 1996] ORMAN A.J., POTTS C.N., SHAHANI A.K., MOORE A.R. (1996), "Scheduling for a multifunction phased array radar system", *European Journal of Operational Research*, Vol. 90, pp. 13-25.

[Orman et *al.*, 1998] ORMAN A.J., SHAHANI A.K., MOORE A.R. (1998), "Modelling for the control of a complex radar system", *Computers and Operations Research*, Vol. 25, No. 3, pp. 239.

P

[Panwalkar et Rajagopalan, 1992] PANWALKAR S.S., RAJAGOPALAN R. (1992), "Single-machine scheduling with controllable processing times", *European Journal of Operational Research*, Vol. 59, pp. 298-302.

[Panwalker et Iskander, 1977] PANWALKER R., ISKANDER W. (1977), "A survey of scheduling rules", *Operations Research*, Vol. 25, No. 1, pp. 45-59.

[Papadimitriou et Kanellakis, 1980] PAPADIMITRIOU C.H., KANELLAKIS P.C. (1980), "Flowshop scheduling with limited temporary storage", *Journal of the ACM-Association for Computing Machinery*, Vol. 27, No. 3, pp. 533-549.

[Pekny et Miller, 1991] PEKNY J.F., MILLER D.L. (1991), "Exact solution of the no-wait flowshop scheduling problem with a comparison to heuristic methods", *Computers and Chemical Engineering*, Vol. 15, No. 11, pp. 741-749.

[Piehler, 1960] PIEHLER J. (1960), "Ein beitrag zum reinhenfolgeproblem", *Unternehmensforschung*, Vol. 4, pp. 138-142.

[Pinedo, 1995] PINEDO M. (1995), "Scheduling: theory, algorithms, and systems", Prentice-Hall, Englewood Cliffs, New Jersey, Etats-Unis.

R

[Rajendran, 1994] RAJENDRAN C. (1994), "A no-wait flowshop scheduling heuristic to minimize makespan", *Journal of the Operational Research Society*, Vol. 45, No. 4, pp. 472-480.

[Ramudhin et Ratliff, 1992] RAMUDHIN A., RATLIFF H.D. (1992), "Generation daily production schedules in process industries", Working Paper 92-51, Faculté des Sciences de l'Administration, Université Laval, Québec, Canada.

[Ramudhin et Ratliff, 1994] RAMUDHIN A., RATLIFF H.D. (1994), "On the complexity of the process shop", *Infor*, Vol. 32, No. 2, pp. 99-109.

[Reddi et Ramamoorthy, 1972] REDDI S.S., RAMAMOORTHY C.V (1972) "On the flow shop sequencing problem with no wait in process", *Operations Research Quarterly*, Vol. 23, pp. 323-331.

[Reklaitis, 1982] REKLAITIS G.V. (1982), "Review of scheduling of process operations", *AIChE Symposium Series*, Vol. 78, pp. 119-133.

[Rinnooy Kan, 1976] RINNOOY KAN A.H.V. (1976), "Machine scheduling problems: classification, complexity and computations", Nijhoff, The Hague.

[Röck, 1984a] ROCK H. (1984a), "The three-machine no-wait flow shop problem is NP-complete", *Journal of the ACM-Association for Computing Machinery*, Vol. 51, pp. 336-345.

[Röck, 1984b] ROCK H. (1984b), "Some new results in flow shop scheduling", *Zeitschrift für Operations Research*, Vol. 28, pp. 1-16.

[Roy, 1960] ROY B. (1960), "Contribution de la théorie des graphes à l'étude des problèmes d'ordonnancement", Communication à la deuxième *conférence internationale de recherche opérationnelle*, Aix-en-Provence, France, septembre 1960, paru dans : "Les Problèmes d'ordonnancement, applications et méthodes", Dunod, Paris, France, 1964, pp. 109-125.

S

[Sahni et Cho, 1979] SAHNI S., CHO Y. (1979) "Complexity of scheduling jobs with no-wait in process", *Mathematics of Operations Research*, Vol. 4, pp. 448-457.

[Salvador, 1973] SALVADOR M.S. (1973), "A solution of a special class of flowshop scheduling problems", In Proceedings of the Symposium on the *Theory of scheduling and its applications*, Springer-Verlag, Berlin, Allemagne.

[Schuurman et Woeginger, 1997] SCHUURMAN P., WOEGINGER G.J. (1997), "A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem", Rapport de recherche Woe-01, START Project Y43-MAT, Graz, Autriche.

[Sethi et *al.*, 1992] SETHI R., SRISKANDARAJAH C., SORGER G., BLAZEWICZ J., KUBIAK W. (1992), "Sequencing of parts and robot moves in a robotic cell", *International Journal of Flexible Manufacturing Systems*, Vol. 4, pp. 331-358.

[Sethi, 1976] SETHI R. (1976) "Scheduling graphs on two processors", *SIAM Journal on Computing*, Vol. 5, pp. 73-82.

[Sgall, 1998] SGALL J. (1998), "On-line algorithms-The state of the art", Chapitre 9 dans FIAT A., WOEGINGER G., eds., "Lectures notes in computer science", Springer-Verlag, pp. 196-231.

[Shmelev, 1997] SHMELEV V.V. (1997), "Dynamic Problems of Scheduling", *Automation and Remote Control*, Vol. 58, No. 1-2, pp. 98-101.

[Sidney et Sriskandarajah, 1999] SIDNEY J.B., SRISKANDARAJAH C. (1999), "A heuristic for the two-machine no-wait openshop scheduling problem", *Naval Research Logistics*, Vol. 46, No. 2, pp. 129-146.

[Simons et Warmuth, 1989] SIMONS B., WARMUTH M. (1989), "A fast algorithm for multiprocessor scheduling of unit-length jobs", *SIAM Journal on Computing*, Vol. 18, pp. 690-710.

[Song et *al.*, 1993] SONG W., ZABINSKY Z.B., STORCH R.L. (1993), "An algorithm for scheduling a chemical processing tank line", *Production Planning and Control*, Vol. 4, No. 4, pp. 323-332.

[Sotskov, 1991] SOTSKOV Y.N., (1991), "The complexity of shop scheduling problems with two or three jobs", *European Journal of Operational Research*, Vol. 53, pp. 326-336.

[Sotskov et *al.*, 1997] SOTSKOV Y.N., SOTSKOVA N.Y., WERNER F. (1997), "Stability of an Optimal Schedule in a Job Shop", *Omega*, Vol. 25, No. 4, pp. 397-414.

[Souilah, 1990] SOUILAH A. (1990), "La séquence globale minimale", Rapport de recherche INRIA, No. 1328, Le Chesnay, France.

[Sriskandarajah et Ladet, 1986] SRISKANDARAJAH C., LADET P. (1986), "Some no-wait shops scheduling problems: complexity aspect", *European Journal of Operational Research*, Vol. 24, No. 3, pp. 424-438.

[Sriskandarajah et Wagneur, 1994] SRISKANDARAJAH C., WAGNEUR E. (1994), "On the complexity of preemptive open shop scheduling problems", *European Journal of Operational Research*, Vol. 77, pp. 404-414.

[Strusevich, 1991] STRUSEVICH V.A. (1991), "Complexity aspects of shop scheduling problems", Doctoral Thesis, Erasmus University, Rotterdam, Pays-Bas.

[Strusevich, 1995] STRUSEVICH V.A. (1995), "Two machine flow shop scheduling problem with no-wait in process: controllable machine speeds", *Discrete Applied Mathematics*, Vol. 59, No. 1, pp. 75-86.

T

[Tanaev et *al.*, 1994a] TANAEV V.S., GORDON V.S., SHAFRANSKY Y.M. (1994a), "Scheduling theory: single-stage systems", Kluwer, Dordrecht, Pays-Bas.

[Tanaev et *al.*, 1994b] TANAEV V.S., SOTSKOV Y.N., STRUSEVICH V.A. (1994b), "Scheduling theory: multi-stage systems", Kluwer, Dordrecht, Pays-Bas.

[Tautenhahn et Woeginger, 1997] TAUTENHAHN T., WOEGINGER G.J. (1997), " Minimizing the total completion time in a unit time open shop with release times ", *Operations Research Letters*, Vol. 20, No. 5, pp. 207-212.

[Timkovsky, 1985a] TIMKOVSKY V.G. (1985a), "On the complexity of scheduling an arbitrary system", *Soviet Journal of Computer and System Sciences*, Vol. 5, pp. 46-52.

[Timkovsky, 1985b] TIMKOVSKY V.G. (1985b), "Polynomial-time algorithm for the Lenstra-Rinnooy Kan two-machine scheduling problem", (en Russe), *Kibernetika*, Vol. 2, pp. 102-109.

[Timkovsky, 1993] TIMKOVSKY V.G. (1993), "The complexity of unit-time job-shop scheduling", Technical Report 93-09, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada.

[Timkovsky, 1997] TIMKOVSKY V.G. (1997), "A polynomial-time algorithm for the two-machine unit-time release-date job-shop schedule-length problem", *Discrete Applied Mathematics*, Vol. 77, pp. 185-200.

[Timkovsky, 1998a] TIMKOVSKY V.G. (1998a), "Is a unit-time job shop not easier than identical parallel machines?", *Discrete Applied Mathematics*, Vol. 85, pp. 149-162.

[Timkovsky, 1998b] TIMKOVSKY V.G. (1998b), "Identical parallel machines vs. unit-time shops, preemptions vs. chains, and other offsets in scheduling complexity", Technical Report, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada.

[Timkovsky, 1998c] TIMKOVSKY V.G. (1998c), "Scheduling unit-time operation jobs on identical parallel machines and in a flow shop: complexity and correlation", Technical Report, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada.

[Trick, 1994] TRICK M.A. (1994), "Scheduling multiple variable-speed machines", *Operations Research*, Vol. 24, No. 2, pp. 234-248.

U

[Ullman, 1975] ULLMAN J.D. (1975), "NP-complete scheduling problems", *Journal of Computer and System Sciences*, Vol. 10, pp. 384-393.

V

[Varnier, 1996] VARNIER C. (1996), "Extensions du hoist scheduling problem cyclique", Thèse de doctorat, Université de Franche-Comté, France.

[van Hoesel, 1991] VAN HOESEL C.P.M. (1991), "An $O(n \log n)$ algorithm for the two machine flow shop scheduling problem with controllable machine speeds", Report 9112/A, Econometric Institute, Erasmus University, Rotterdam, Pays-Bas.

[van Vielt, 1991] VAN VIELT M. (1991), "Optimization of manufacturing system design", Ph.D Thesis, Erasmus University, Rotterdam, Pays-Bas.

[Vickson, 1980] VICKSON R.G. (1980), "Two single-machine sequencing problems involving controllable processing times", *IIE Transactions*, Vol. 12, pp. 258-262.

Y

[Yih, 1988] YIH Y. (1988), "Trace driven knowledge acquisition (TDKA) for rule based real time scheduling systems", *Journal of Intelligence Manufacturing*, Vol. 1, No. 4, pp. 217-230.

[Yih et Thesen, 1991] YIH Y., THESEN A. (1991), "Semi-Markov decision models for real time scheduling", *International Journal of Production Research*, Vol. 29, No. 11, pp. 2331-2346

[Yih, 1994] YIH Y. (1994), "An algorithm for hoist scheduling problems", *International Journal of Production Research*, Vol. 32, No. 3, pp. 501-516.

[Yin et Yih, 1992] YIN N.-C., YIH Y. (1992), "Crane scheduling in a flexible electroplating line: a tolerance-based approach", *Journal of Electronics Manufacturing*, Vol. 2, pp. 137-144.

Z

[Zweben et Fox, 1994] ZWEBEN M., FOX M. (1994), "Intelligent scheduling", Morgan & Kaufmann, San Mateo, Californie, Etats-Unis.

ANNEXE I

GESTION DES PERIODES DE DISPONIBILITE

Sommaire

Cette annexe décrit les méthodes de gestion des périodes de disponibilité. Ces méthodes sont définies de deux manières suivant que les ressources sont interchangeables en cours d'opération ou non. Cette distinction est originale, tout particulièrement dans le contexte temps réel. Celui-ci impose, en effet, une attention toute particulière à la complexité des algorithmes proposés.

Plan

I.1. INTRODUCTION

Les ressources sont utilisées soit pour des tâches extérieures à la fabrication proprement dite, soit pour des tâches de fabrication de produits préalablement ordonnancés. Après ordonnancement des $(j\text{-}1)$ premiers produits et affectation des ressources nécessaires aux opérations correspondantes, nous devons déduire les fenêtres de temps durant lesquelles chaque opération du produit j peut être exécutée. Au cours de la période consacrée à l'exécution de l'opération i du produit j, il faut que le nombre de ressources de type r soit supérieur ou égal à $q_{r,i,j}$ qui est le nombre de ressources requises pour cette opération. Une fois le produit j ordonnancé, il faut également remettre à jour les disponibilités des ressources en fonction des utilisations par le produit j.

Ces procédures de gestion des fenêtres de disponibilité des ressources sont relatives à chaque opération : elles sont indépendantes des types de gammes de produits (linéaire, assemblage...). En effet, les gammes définissent l'ordre dans lequel les opérations doivent être exécutées et cet ordre n'a aucune influence sur les procédures de gestion des fenêtres.

L'objectif de cette annexe est de présenter les structures supportant les données relatives aux disponibilités des ressources, les algorithmes permettant de définir les périodes durant lesquelles chaque opération peut être exécutée, et les méthodes de mise à jour des disponibilités des ressources.

I.2. STRUCTURES DE DONNEES

I.2.1. Introduction

Dans cette section, nous définissons les structures supportant les périodes de disponibilité des ressources. Nous proposons de définir ces périodes de façons différentes suivant que les exemplaires de la ressource sont interchangeables en cours d'opération ou non. Cette distinction est possible puisque nous faisons l'hypothèse que nous savons quelles ressources sont interchangeables en cours d'opération.

I.2.2. Cas des ressources interchangeables

Les exemplaires d'une ressource interchangeable en cours d'opération peuvent être remplacés par d'autres en cours d'opération. C'est le cas des opérateurs qui peuvent se relayer sans que l'opération ne s'arrête. L'ensemble des ressources interchangeables en cours d'opération est noté R_{in}. Nous supposons que les exemplaires d'un même type de ressource sont identiques et que le temps de changement d'un exemplaire à l'autre en cours d'opération est nul.

Les intervalles de temps durant lesquels ces ressources sont disponibles sont décrits par les quantités disponibles de chaque ressource. Formellement, on sait que la ressource de type r, $r \in R_{in}$, est disponible en quantité positive $d_{v,r}$ durant tout l'intervalle $[\alpha_{v,r} ; \beta_{v,r}]$, où $v \in \{1, 2, \ldots, \omega_r\}$. ω_r désigne le nombre d'intervalles relatifs à la ressource de type r. Ces intervalles sont non vides. On a : $d_{v,r} \in \Re^+ \backslash \{0\}$, $\alpha_{v,r} < \beta_{v,r}$, $\alpha_{v,r} \in \Re^+$ et $\beta_{v,r} \in \Re^+ \cup \{+\infty\}$, pour $v \in \{1, 2, \ldots, \omega_r\}$ et $r \in R_{in}$. $\Re^+$ désigne l'ensemble des nombres réels positifs.

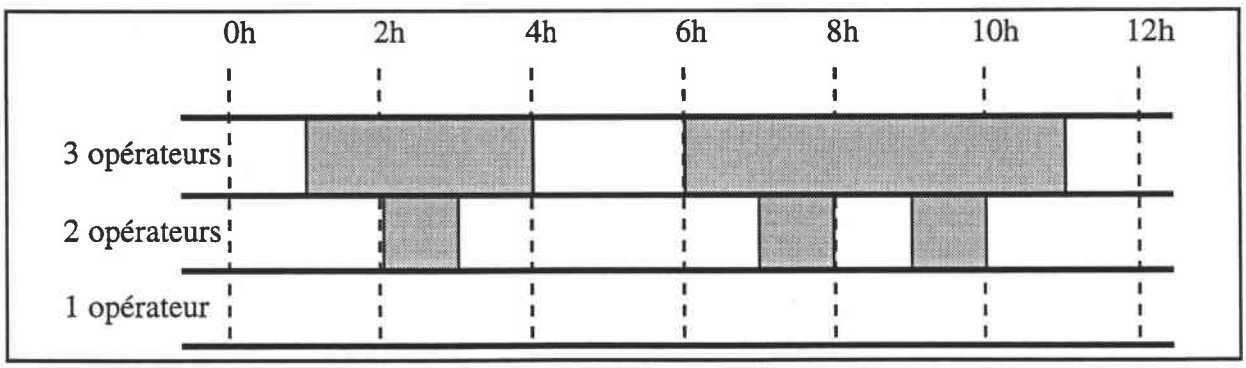

Figure A.1- 1 : Disponibilité d'une ressource interchangeable

Par exemple, nous considérons un groupe de 3 opérateurs ayant les mêmes compétences qui peuvent se succéder au cours de certaines opérations. Nous supposons que l'instant initial est la date 0. Nous pourrions avoir les ω_r=11 intervalles suivants : [0 ; 1h] avec $d_{1,r}$=3, [1h ; 2h] avec $d_{2,r}$=2, [2h ; 3h] avec $d_{3,r}$=1, [3h ; 4h] avec $d_{4,r}$=2, [4h ; 6h] avec $d_{5,r}$=3,

[6h ; 7h] avec $d_{6,r}$=2, [7h ; 8h] avec $d_{7,r}$=1, [8h ; 9h] avec $d_{8,r}$=2, [9h ; 10h] avec $d_{9,r}$=1, [10h ; 11h] avec $d_{10,r}$=2, [11h ; +∞[avec $d_{11,r}$=3. La Figure A.1- 1 représente cette situation avec des périodes d'indisponibilité grisées. Puisque les 3 opérateurs sont interchangeables à tout instant, il est possible de distribuer le travail entre les opérateurs à tout moment. Il est alors aisé d'équilibrer leur charge de travail.

Nous définissons les intervalles $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ relatifs à une ressource de type r de telle façon :

- qu'ils ne se chevauchent pas ou aient leur extrémité commune, et
- qu'ils soient ordonnées chronologiquement à savoir $\beta_{v,r} \leq \alpha_{v+1,r}$, pour $v \in \{1, 2, \ldots, \omega_r - 1\}$.

I.2.3. Cas des ressources non-interchangeables

Les exemplaires d'une ressource non-interchangeable (en cours d'opération) ne peuvent être remplacés par d'autres en cours d'opération. Par exemple, une machine choisie au début d'une opération ne peut être remplacée par une autre. L'ensemble des ressources non-interchangeables en cours d'opération est noté R_{en}. On a : $R_{en} = R / R_{in}$. Nous supposons que les exemplaires d'un même type de ressource sont identiques.

Les intervalles de temps durant lesquels ces ressources sont disponibles sont décrits pour chaque exemplaire. K_r désigne le nombre total d'exemplaires de la ressource de type r, $r \in R_{en}$. On a : $K_r \in$ N où N est l'ensemble des entiers naturels. Plus précisément, pour chaque exemplaire $k \in \{1, 2, \ldots, K_r\}$ de la ressource de type r, $r \in R_{en}$, nous connaissons les périodes $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ durant lesquelles l'exemplaire k est disponible, $u \in \{1, 2, \ldots, \Omega_{k,r}\}$. Nous notons $\Omega_{k,r}$ le nombre de périodes relatives au k-ième exemplaire de la ressource de type r. Ces périodes sont non vides. On a : $\zeta_{u,k,r} < \xi_{u,k,r}$, $\zeta_{u,k,r} \in \Re^+$ et $\xi_{u,k,r} \in \Re^+ \cup \{+\infty\}$, pour $u \in \{1, 2, \ldots, \Omega_{k,r}\}$, $k \in \{1, 2, \ldots, K_r\}$ et $r \in R_{en}$, où $\Re^+$ désigne l'ensemble des nombres réels positifs.

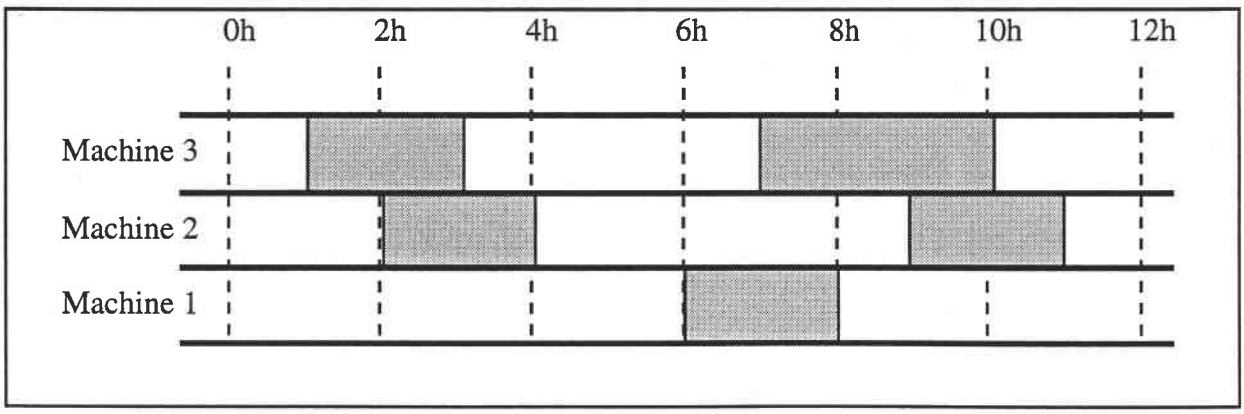

Figure A.1- 2 : Disponibilité d'une ressource non-interchangeable

Par exemple, dans un atelier, les machines identiques sont capables d'exécuter les mêmes opérations. Dans ces ateliers, on s'interdit le plus souvent de changer de machines en-cours d'opération. Nous supposons que l'instant initial est la date 0. Les périodes de disponibilité pour un tel atelier composé de K_r=3 machines pourraient être : [0 ; 6h] et [8h ; +∞[pour la machine 1 ; [0 ; 2h], [4h ; 9h] et [11h ; +∞[pour la machine 2 ; [0 ; 1h], [3h ; 7h] et [10h ; +∞[pour la machine 3. La Figure A.1- 2 présente cet exemple avec des périodes d'indisponibilité grisées. La dernière période de chaque exemplaire est illimitée puisque les ressources sont toutes disponibles au-delà de 11h.

Nous définissons les périodes $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ relatives à un exemplaire k d'une ressource de type r de telle façon qu'ils **ne se chevauchent pas, n'aient pas d'extrémité commune** et **soient données dans l'ordre chronologique** à savoir $\xi_{u,k,r}<\zeta_{u+1,k,r}$, pour $u\in\{1, 2, \ldots, \Omega_{k,r}-1\}$.

I.2.4. Conclusion

Nous avons défini dans cette section les structures supportant les périodes de disponibilité des ressources, suivant que les exemplaires de la ressource sont interchangeables en cours d'opération ou non. Il est possible de maintenir ces périodes en mémoire autrement que de la façon indiquée ci-dessous. Néanmoins, elle offre un bon compromis entre souplesse d'utilisation et temps de gestion, comme nous le verrons dans les sections suivantes.

Nous indiquons à l'intention du programmeur que les périodes de disponibilité des ressources peuvent être maintenues en mémoire dans des structures de données simples de type liste chaînée [Beauquier et al., 1992]. Ce type de structure permet des suppressions ou des ajouts immédiats. L'accès à tout élément d'une liste de $|L|$ éléments coûte $O(|L|)$. Puisque les périodes sont ordonnées chronologiquement, il est également possible d'utiliser des structures du type arbres binaires de recherche permettant de conserver cet ordre : arbres AVL, arbres a-b, arbres bicolores [Beauquier et al., 1992]. Ces structures permettent d'améliorer les accès à chaque élément qui coûtent alors seulement $O(\log_2 |L|)$ au détriment des suppressions et des ajouts qui requièrent dans ce cas $O(\log_2 |L|)$.

I.3. DEFINITION DES FENETRES D'EXECUTION

I.3.1. Introduction

L'ordonnancement des $(j\text{-}1)$ premiers produits ne peut être remis en cause quand la j-ième demande apparaît. Les disponibilités des ressources sont alors connues de façon précise. A partir de ces disponibilités, nous définissons les fenêtres de temps durant lesquelles peut être exécutée chaque opération. Pour cela, nous définissons, tout d'abord, les périodes durant lesquelles les disponibilités des ressources interchangeables sont suffisantes. Ensuite, nous dérivons de ces périodes, l'ensemble $W_{i,j}$ des $y_{i,j}$ fenêtres durant lesquelles toutes les ressources nécessaires (interchangeables et non-interchangeables) à l'opération i du produit j sont présentes en quantité suffisante : $W_{i,j}=\{[a_{z,i,j}\ ;\ b_{z,i,j}]\ /\ 1\leq z\leq y_{i,j}\}$.

I.3.2. Définition des périodes de disponibilité des ressources interchangeables

Introduction

Nous connaissons pour chaque ressource interchangeable de type r, $r\in R_{in}$, les intervalles de temps $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ durant lesquels elle est disponible, et la quantité disponible : $d_{v,r}\in\Re^{+}\backslash\{0\}$, $v\in\{1,\ 2,\ \ldots,\ \omega_r\}$. ω_r désigne le nombre d'intervalles relatifs à la ressource de type r. Nous recherchons les périodes $[\zeta_{u,1,r}\ ;\ \xi_{u,1,r}]$ durant lesquelles la ressource de type r est disponible en quantité suffisante (i.e. supérieure ou égale à $q_{r,i,j}$) pour que l'opération considérée puisse être exécutée, $u\in\{1,\ 2,\ \ldots,\ \Omega_{1,r}\}$. Nous désignons par $\Omega_{1,r}$ le nombre de ces périodes. Pour chaque période $[\zeta_{u,1,r}\ ;\ \xi_{u,1,r}]$, nous définissons un pointeur $V_{u,r,i,j}$. Ce pointeur permet de savoir quels intervalles $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ correspondent à la période u. Le but est d'obtenir des périodes de disponibilité pour la ressource interchangeable de type r qui puissent être traitées comme celles d'une ressource non interchangeable ayant un unique exemplaire : $K_r=1$.

Algorithme

Pour définir les périodes $[\zeta_{u,1,r}\ ;\ \xi_{u,1,r}]$, il suffit de rechercher les unions d'intervalles de disponibilité $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ durant lesquels le nombre d'exemplaires de ressources atteint ou dépasse $q_{r,i,j}$. C'est le principe de l'Algorithme A.1- 1. Rappelons que les intervalles $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ sont donnés dans l'ordre chronologique : on a $\beta_{v,r}\leq\alpha_{v+1,r}$, pour $v\in\{1,\ 2,\ \ldots,\ \omega_r\text{-}1\}$.

Algorithme A.1- 1 : Définition des périodes pour chaque ressource interchangeable

Procédure Définition des périodes durant lesquelles la ressource de type r est disponible pour l'opération i du produit j.

Entrée : r est la ressource interchangeable considérée.
i et j sont respectivement les indices de l'opération et du produit considérés.
$q_{r,i,j}$ est le nombre de ressources de type r nécessaires pour exécuter l'opération.
ω_r est le nombre de fenêtres relatives à la ressource de type r.
$[\alpha_{v,r} ; \beta_{v,r}]$ est la v-ième fenêtre, pour $v \in \{1, 2, \ldots, \omega_r\}$.
$d_{v,r}$ est le nombre d'unités de ressource de type r disponibles au cours de la v-ième fenêtre, pour $v \in \{1, 2, \ldots, \omega_r\}$.

Sortie : K_r=1 est le nombre d'exemplaires de la ressource considérée comme non interchangeable.
$\Omega_{1,r}$ est le nombre de périodes durant lesquelles la ressource est en nombre suffisant.
$[\zeta_{u,1,r} ; \xi_{u,1,r}]$ est la u-ième période, où $u \in \{1, 2, \ldots, \Omega_{1,r}\}$.
$V_{u,r,i,j}$ est le pointeur qui permet de savoir quels intervalles $[\alpha_{v,r} ; \beta_{v,r}]$ correspondent à la u-ième période, où $u \in \{1, 2, \ldots, \Omega_{1,r}\}$.

1. **Initialisation.**
   1.1. **Poser** K_r:=1. *La ressource est ensuite considérée comme en un exemplaire unique.*
   1.1. **Poser** u:=0. *On note u l'indice de la dernière période définie.*
   1.2. **Poser** $suff$:=0. *On note $suff$ l'indicateur qui vaut :*
   0 *si le nombre de ressources est insuffisant sur l'intervalle* $[\alpha_{v-1,r} ; \beta_{v-1,r}]$.
   1 *sinon.*
2. **Si** ($q_{r,i,j}$:=0) **alors** : *L'opération ne nécessite pas cette ressource qui est alors toujours disponible.*
   2.1. **Poser** $\Omega_{1,r}$:= .1
   2.2. **Poser** $\zeta_{1,1,r}$:=0 ; $\xi_{1,1,r}$:=+∞ . *La période de disponibilité correspondante est* [0 ; +∞[.
   2.3. **Poser** $V_{1,r,i,j}$:=∅ .
3. **Sinon** :
   3.1. **Pour** v allant de 1 à ω_r : *On examine les intervalles* $[\alpha_{v,r} ; \beta_{v,r}]$ *un par un.*
      3.1.1. **Si** ($q_{r,i,j} \leq d_{v,r}$ et $suff$=0) **alors** : *Le nombre d'unités de ressource est suffisant sur* $[\alpha_{v,r} ; \beta_{v,r}]$.
         3.1.1.1. **Poser** u:=u+1. *Une nouvelle période est créée.*
         3.1.1.2. **Poser** $\zeta_{u,1,r}$:=$\alpha_{v,r}$. *La date de début de la période est définie.*
         3.1.1.3. **Poser** w:=v . *L'indice de l'intervalle est sauvegardé dans la variable w.*
         3.1.1.4. **Poser** $suff$:=1.
      3.1.2. **Sinon, si** ($q_{r,i,j} > d_{v,r}$ et $suff$=1) **alors** : *Le nombre d'unités de ressource est insuffisant sur* $[\alpha_{v,r} ; \beta_{v,r}]$.
         3.1.2.1. **Poser** $\xi_{u,1,r}$:=$\beta_{v-1,r}$. *La date de fin de la période est définie.*
         3.1.2.2. **Poser** $V_{u,r,i,j}$:=(w ; v-1). *Les indices des intervalles de début et de fin correspondant à la période sont définis.*
         3.1.2.3. **Poser** $suff$:=0.
   3.2. **Si** ($q_{r,i,j} \leq d_{v,r}$) **alors** : *Le nombre d'unités de ressource est suffisant sur le dernier intervalle.*
      3.2.1. **Poser** $\xi_{u,1,r}$:=$\beta_{v,r}$. *La date de fin de la dernière période est définie.*
      3.2.2. **Poser** $V_{u,r,i,j}$:=(w ; v). *Les indices des intervalles de début et de fin relatifs à la dernière période sont définis.*
   3.3. **Poser** $\Omega_{1,r}$:=u . *Le nombre de périodes est défini.*

Exemple

Considérant les disponibilités de la ressource interchangeable de type r données par la Figure A.1- 1, l'Algorithme A.1- 1 fournit les périodes durant lesquelles cette ressource est disponible

en quantité $q_{r,i,j} \geq 2$. Ces $\Omega_{1,r}=4$ périodes sont les suivantes : [0h ; 2h], [3h ; 7h], [8h ; 9h], [10h ; +∞[. Elles sont représentés en blanc dans la Figure A.1- 3.

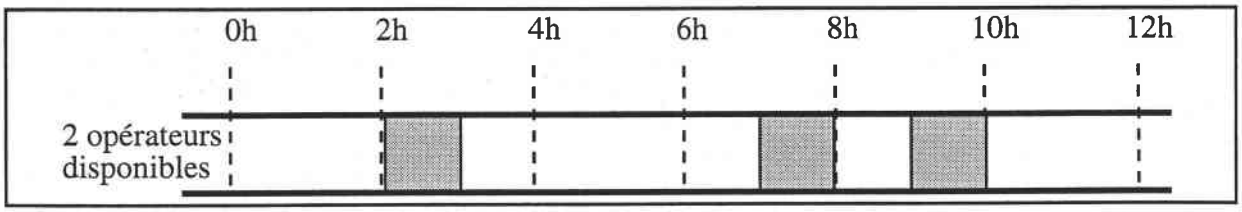

Figure A.1- 3 : Disponibilité d'une ressource interchangeable

Remarques

L'Algorithme A.1- 1 requiert $O(\omega_r)$ opérations et le nombre de périodes qu'il génère est $\Omega_{r,i,j} \leq (\omega_r+1)/2$, où ω_r est le nombre de fenêtres initiales relatives à la ressource de type r. Les intervalles $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ étant non-vides et donnés dans l'ordre chronologique, les périodes $[\zeta_{u,1,r}\ ;\ \xi_{u,1,r}]$ sont non vides, ne se chevauchent pas, n'ont pas d'intersection commune et sont répertoriées dans l'ordre chronologique.

Supposons que deux opérations i et i' du produit j, $i \in I_j$ et $i' \in I_j$, nécessitent le même nombre $q_{r,i,j}=q_{r,i',j}$ de ressources de type r, $r \in R_{in}$. Dans ce cas, l'Algorithme A.1- 1 est appliqué une seule fois et les 2 opérations font référence au même ensemble de périodes.

I.3.3.Définition des fenêtres durant lesquelles toutes les ressources sont disponibles

Introduction

Pour chaque exemplaire $k \in \{1, 2, \ldots, K_r\}$ de la ressource de type r, $r \in R = R_{in} \cup R_{en}$, nous connaissons les périodes $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ durant lesquelles cet exemplaire k est disponible, $u \in \{1, 2, \ldots, \Omega_{k,r}\}$. $\Omega_{k,r}$ est le nombre de périodes relatives au k-ième exemplaire de la ressource de type r. Nous recherchons les fenêtres de temps $[a_{z,i,j}\ ;\ b_{z,i,j}]$ durant lesquelles l'opération i du produit j peut être exécutée, $z \in \{1, 2, \ldots, y_{i,j}\}$. $y_{i,j}$ désigne le nombre de ces fenêtres. Pour chaque fenêtre $[a_{z,i,j}\ ;\ b_{z,i,j}]$, nous définissons un pointeur $U_{z,i,j}$. Ce pointeur permet de savoir quelles périodes $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ correspondent à la fenêtre z.

Nous supposons que l'ordonnancement des produits précédents n'est pas remis en question par l'ordonnancement du produit j. En particulier, si un exemplaire d'une ressource interchangeable est affecté à une opération, il n'est pas possible de modifier cette affectation au profit d'un autre exemplaire (même si cet exemplaire est disponible pendant toute la durée de l'opération).

Algorithme générant toutes les fenêtres

Algorithme A.1- 2 : Définition des fenêtres pour chaque opération

Procédure Définition des fenêtres durant lesquelles l'opération i du produit j peut être exécutée.

Entrée : i et j sont respectivement les indices de l'opération et du produit considéré.

R est l'ensemble des types de ressources.

R_{in} est l'ensemble des ressources interchangeables.

$l_{i,j}$ est la durée minimale de l'opération, et donc des périodes durant lesquelles l'opération peut être exécutée.

$q_{r,i,j}$ est le nombre d'exemplaires de ressources de type r nécessaires pour l'opération, pour $r \in R$.

K_r est le nombre d'exemplaires de la ressource de type r, pour $r \in R$.

$\Omega_{k,r}$ est le nombre de périodes relatives à la ressource de type r pour $k \in \{1, 2, \ldots, K_r\}$, $r \in R$.

$[\zeta_{u,k,r} ; \xi_{u,k,r}]$ est la u-ième période de l'exemplaire k, pour $u \in \{1, 2, \ldots, \Omega_{k,r}\}$, $k \in \{1, 2, \ldots, K_r\}$ et $r \in R$.

Sortie : $y_{i,j}$ est le nombre de fenêtres durant lesquelles l'opération peut être exécutée.

$[a_{z,i,j} ; b_{z,i,j}]$ est la z-ième fenêtre, où $z \in \{1, 2, \ldots, y_{i,j}\}$.

$U_{z,i,j}$ est le pointeur définissant les indices des périodes $[\zeta_{u,k,r} ; \xi_{u,k,r}]$ correspondant à la z-ième fenêtre $[a_{z,i,j} ; b_{z,i,j}]$, où $z \in \{1, 2, \ldots, y_{i,j}\}$.

1. **Initialisation.**
   - 1.1. **Poser** $z := 0$. *On note z l'indice de la dernière fenêtre définie.*
   - 1.2. **Poser** $R^* := \varnothing$. *On note R^* l'ensemble des types de ressources requis.*
2. **Pour** $r \in R$: *On examine les types de ressources, un par un.*
   - 2.1. **Si** $(q_{r,i,j} \neq 0)$ **alors** : *L'opération ne nécessite pas cette ressource.*
     - 2.1.1. **Poser** $Q_r := 0$. *On note Q_r le nombre d'exemplaires de ressource de type r requis.*
   - 2.2. **Sinon** : *L'opération nécessite cette ressource.*
     - 2.2.1. **Si** $(r \in R_{in})$ **alors** : *La ressource est interchangeable.*
       - 2.2.1.1. **Poser** $Q_r := 1$.
     - 2.2.2. **Sinon** : *La ressource est interchangeable.*
       - 2.2.2.1. **Poser** $Q_r := q_{r,i,j}$. *Le nombre d'exemplaires requis est celui de l'opération.*
     - 2.2.3. **Si** $(Q_r > K_r)$ **alors** : *L'opération nécessite plus d'exemplaires que le nombre existant.*
       - 2.2.3.1. **Poser** $y_{i,j} := 0$. *Aucune fenêtre n'est retenue.*
       - 2.2.3.2. **Quitter**. *L'algorithme est arrêté puisque la ressource ne peut être pourvue.*
     - 2.2.4. **Sinon** : *La ressource de type r est requise.*
       - 2.2.4.1. **Poser** $R^* := R^* \cup \{r\}$.
3. **Si** $(R^* = \varnothing)$ **alors** : *Aucune ressource n'est requise.*
   - 3.1. **Poser** $z := 1$.
   - 3.2. **Poser** $a_{z,i,j} := 0$; $b_{z,i,j} := +\infty$. *L'opération peut être exécutée dans la fenêtre* $[0; +\infty[$.
   - 3.3. **Poser** $U_{z,i,j} := \varnothing$.
4. **Sinon** : *Les ressources de R^* sont nécessaires à l'opération. On note $|R^*|$ le cardinal de l'ensemble R^*.*
   - 4.1. **Pour** $(k_{1,1} ; k_{2,1} ; \ldots ; k_{Q_1,1})$ tels que $1 \leq k_{1,1} < k_{2,1} < \ldots < k_{Q_1,1} \leq K_1$,
     $(k_{1,2} ; k_{2,2} ; \ldots ; k_{Q_2,2})$ tels que $1 \leq k_{1,2} < k_{2,2} < \ldots < k_{Q_2,2} \leq K_2$,
     ...
     $(k_{1,|R^*|} ; k_{2,|R^*|} ; \ldots ; k_{Q_{|R^*|},|R^*|})$ tels que $1 \leq k_{1,|R^*|} < \ldots < k_{Q_{|R^*|},|R^*|} \leq K_{|R^*|}$:
     Les Q_r exemplaires choisis pour la ressource de type r sont $(k_{1,r} ; k_{2,r} ; \ldots ; k_{Q_r,r})$.
     - 4.1.1. **Pour** $r \in R^*$, **pour** q allant de 1 à Q_r :
       - 4.1.1.1. **Poser** $u_{q,r} := 1$. *On note $u_{q,r}$ l'indice de la période de l'exemplaire q choisie.*
     - 4.1.2. **Tant que** $(u_{q,r} \leq \Omega_{k_{q,r},r}$, pour $q \in \{1, 2, \ldots, Q_r\}$ et $r \in R^*)$:
       - 4.1.2.1. **Poser** $a := \max\{\zeta_{u_{q,r},k_{q,r},r} \ / \ q \in \{1, \ldots, Q_r\}, r \in R^*\}$.
       - 4.1.2.2. **Poser** $b := \max\{\xi_{u_{q,r},k_{q,r},r} \ / \ q \in \{1, \ldots, Q_r\}, r \in R^*\}$.

       L'intersection des périodes sélectionnées est la fenêtre $[a;b]$.

4.1.2.3. **Si** $(b-a \geq l_{i,j})$ et $(b-a>0)$ **alors** : *L'opération peut être exécutée dans* $[a;b]$.
4.1.2.3.1. **Poser** $z:=z+1$.
4.1.2.3.2. **Poser** $a_{z,i,j}:=a$; $b_{z,i,j}:=b$. *La fenêtre* $[a;b]$ *est retenue.*
4.1.2.3.3. **Poser** $U_{z,i,j}:=\{(u_{q,r}, k_{q,r}, r) \ / \ q \in \{1, 2, \ldots, Q_r\}, r \in R^*\}$.
4.1.2.4. **Pour** $r \in R^*$, **pour** q allant de 1 à Q_r :
4.1.2.4.1. **Si** $(\xi_{u_{q,r},k_{q,r},r} \leq b+l_{i,j})$ **alors** : *La période sélectionnée ne permetra pas d'obtenir une fenêtre.*
4.1.2.4.1.1. **Poser** $u_{q,r}:=u_{q,r}+1$. *On sélectionne la période suivante.*
5. **Poser** $y_{i,j}:=z$. *Le nombre de fenêtres est définie.*

Pour définir les fenêtres $[a_{z,i,j}\ ;\ b_{z,i,j}]$, il faut rechercher toutes les intersections de période de disponibilité $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ durant lesquelles les besoins en ressource sont satisfaits. C'est le principe de l'Algorithme A.1- 2. La seule difficulté est de générer toutes les combinaisons pour chaque ressource de type r pour laquelle $q_{r,i,j}$ exemplaires parmi les K_r sont disponibles. Rappelons que les périodes $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ relatives à un exemplaire k n'ont pas d'intersection commune et sont répertoriées dans l'ordre chronologique. On a : $\xi_{u,k,r}<\zeta_{u+1,k,r}$, pour $u \in \{1, 2, \ldots, \Omega_{k,r}-1\}$.

Exemple

A titre d'exemple, nous exécutons l'Algorithme A.1- 2 à partir des disponibilités des $K_{r'}=3$ exemplaires de la ressource non-interchangeable de type $r' \in R_{en}$ données par la Figure A.1- 2. L'Algorithme A.1- 2 utilise les disponibilités en quantité $q_{r,i,j} \geq 2$ de l'exemplaire $K_r=1$ de la ressource interchangeable de type $r \in R_{in}$ fourni par l'Algorithme A.1- 1. L'Algorithme A.1- 2 peut alors donner les fenêtres durant lesquelles les ressources de type r et r' sont disponibles en quantité $q_{r,i,j} \geq 2$ et $q_{r',i,j} \geq 1$. Ces $y_{i,j}=11$ fenêtres sont les suivantes : [0h ; 1h], [3h ; 7h], [10h ; +∞[, [0h ; 2h], [4h ; 7h], [8h ; 9h], [11h ; +∞[, [0h ; 2h], [3h ; 6h], [8h ; 9h], [10h ; +∞[. Elles sont représentées en blanc sur la Figure A.1- 4.

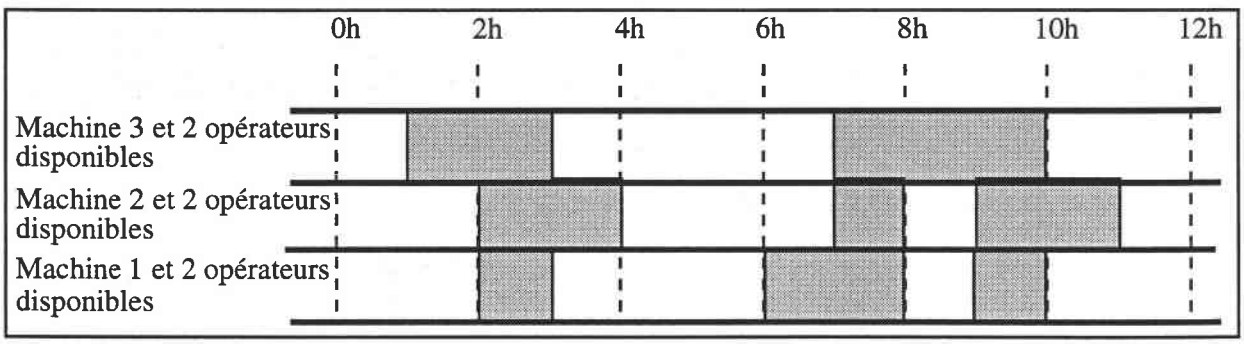

Figure A.1- 4 : Fenêtres d'exécution d'une opération

Complexité quand on génère toutes les fenêtres

Les pas 1 à 4 de l'Algorithme A.1- 2 requièrent $O(|R|)$ opérations, où $|R|$ est le cardinal de l'ensemble R. Au pas 4.1, on énumère toutes les combinaisons possibles de $q_{r,i,j}$ exemplaires parmi les K_r pour chaque ressource de type r. Le nombre de ces combinaisons est

$\sum_{\substack{1\le k_{1,1}<k_{2,1}<\ldots<k_{q_{1,i,j},1}\le K_1 \\ 1\le k_{1,2}<k_{2,2}<\ldots<k_{q_{2,i,j},2}\le K_2 \\ \ldots \\ 1\le k_{1,|R^*|}<k_{2,|R^*|}<\ldots<k_{q_{|R^*|,i,j},r}\le K_{|R^*|}}} 1 = \prod_{\substack{r'\in R^*_{en} \\ q_{r',i,j}>0}} \frac{K_{r'}!}{q_{r',i,j}!\left(K_{r'}-q_{r',i,j}\right)!}$. Seules les ressources non-interchangeables requises pour l'opération interviennent dans ce calcul, puisque pour chaque ressource interchangeable de type $r\notin R_{en}$, on a $K_r=1$. Pour chaque combinaison, le pas 4.1.1 est exécuté $O(|R^*|)$ fois. Quant au pas 4.1.2, il est exécuté au plus $1+\sum_{r\in R^*}\left(\Omega_{k_{1,r},r}+\Omega_{k_{2,r},r}+\ldots+\Omega_{k_{q_{r,i,j},r},r}-q_{r,i,j}\right)$ fois, puisque au moins un des indices $u_{q,r}$ est incrémenté au pas 4.1.2.4.1.1. R^* et $\Omega_{k,r}$ désignent respectivement le nombre de ressources requises pour l'opération et le nombre de périodes initiales relatives à l'exemplaire k de la ressource de type r. Les pas 4.1.2.1 à 4.1.2.4.1.1 nécessitent à chaque passage $O(|R^*|)$ opérations. Finalement, la complexité de l'Algorithme A.1- 2 est en $O\left(|R|+\sum_{\substack{1\le k_{1,1}<k_{2,1}<\ldots<k_{q_{1,i,j},1}\le K_1 \\ \ldots \\ 1\le k_{1,|R^*|}<k_{2,|R^*|}<\ldots<k_{q_{|R^*|,i,j},r}\le K_{|R^*|}}} \sum_{r\in R^*}\left(\Omega_{k_1,r}+\Omega_{k_2,r}+\ldots+\Omega_{k_{q_{r,i,j}},r}\right)\right)$. On appelle q_0 la plus grande des quantités nécessaires pour un type de ressource. De même, K_0 est le plus grand nombre d'exemplaires d'une ressource non-interchangeable et Ω_0 est le plus grand nombre de fenêtres pour un exemplaire d'un type de ressource. Une borne supérieure de la complexité de l'Algorithme A.1- 2 est alors $O\left(|R|+\left(\frac{K_0(K_0+1)\ldots(K_0-q_0+1)}{q_0!}\right)^{|R^*_{en}|}|R^*|q_0\Omega_0\right)$. Cette borne est atteinte si les besoins sont identiques pour chaque type de ressource et si tous les types de ressources ont le même nombre d'exemplaires et, pour chacun d'eux, le même nombre de fenêtres. Ce temps peut paraître important. En fait, le nombre d'exemplaires de ressources interchangeables requis pour une opération est très limité. Dans la plupart des cas pratiques qui nous intéressent, on a besoin d'une seule machine pour réaliser une opération et on a $\sum_{r\in R_{en}} q_{r,i,j}=1$. Dans ce cas, la complexité de l'Algorithme A.1- 2 est en $O(|R|+K_0|R^*|q_0\Omega_0)$.

Nombre de fenêtres générées

Une borne supérieure du nombre $y_{i,j}$ de fenêtres générées par l'Algorithme A.1- 2 est donné par le temps de calcul : $\left(\sum_{\substack{1\le k_{1,1}<k_{2,1}<\ldots<k_{q_{1,i,j},1}\le K_1 \\ \ldots \\ 1\le k_{1,|R^*|}<k_{2,|R^*|}<\ldots<k_{q_{|R^*|,i,j},r}\le K_{|R^*|}}} \left(1+\sum_{r\in R^*}\left(\Omega_{k_1,r}+\Omega_{k_2,r}+\ldots+\Omega_{k_{q_{r,i,j}},r}-q_{r,i,j}\right)\right)\right)$. Cette évaluation par excès correspond au cas où à chaque itération une nouvelle fenêtre est construite. Il est possible de créer des exemples pour lesquels cette borne est atteinte comme le cas de la Figure A.1- 5. Sur la Figure A.1- 5b, la ressource désignée par "A1" est l'exemplaire "1" de la ressource "A" de la Figure A.1- 5a. Les ressources "B1", "C1", "D1", "C2" et "D2" sont

définies de manière analogue. A partir de 6 exemplaires de 4 ressources différentes disponibles chacun pendant 3 périodes, $y_{i,j}=\left(\sum_{\substack{1\le k_{1,1}\le 2\\ 1\le k_{1,2}\le 2}}\left(1+\sum_{r\in\{1,\,2,\,3,\,4\}}(3-1)\right)\right)$=4x9=36 fenêtres sont générées. Finalement, dans le cas le plus courant où $\sum_{r\in R_{en}} q_{r,i,j}=1$, l'Algorithme A.1- 2 peut générer $y_{i,j}=K_0\left(1+|R^*|q_0\left(\Omega_0-1\right)\right)$ fenêtres différentes $[a_{z,i,j}\ ;\ b_{z,i,j}]$, où $1\le z\le y_{i,j}$.

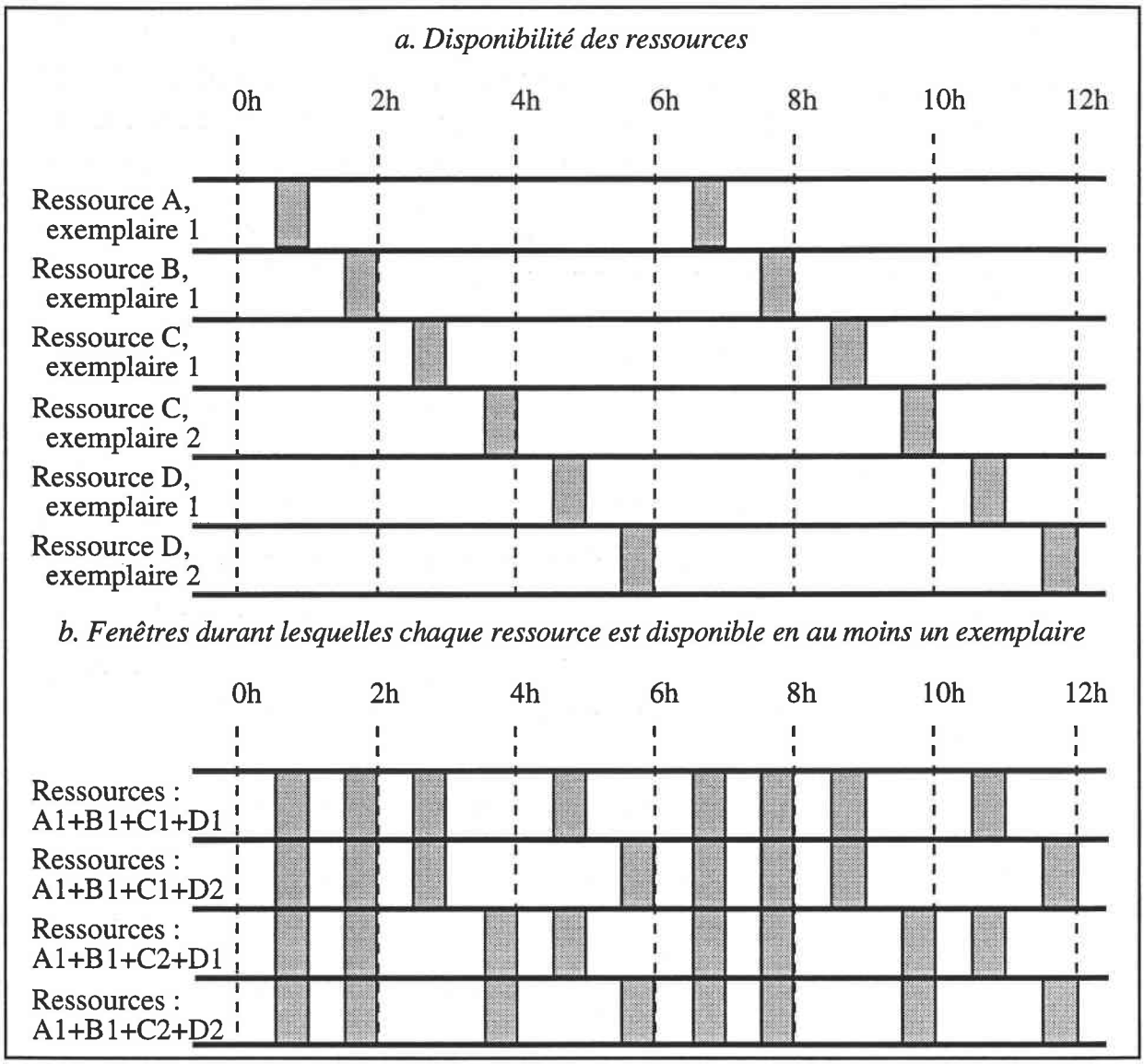

Figure A.1- 5 : Cas extrême de création de fenêtres d'exécution

Remarque

En recherchant les périodes durant lesquelles le nombre d'exemplaires est suffisant, l'Algorithme A.1- 2 peut générer des fenêtres incluses les unes dans les autres comme le montre la Figure A.1- 5. Si on cherche les plages de temps les plus longues possibles dans lesquelles exécuter chaque opération (c'est le cas pour tous les algorithmes présentés dans cette thèse),

nous supprimerons les fenêtres contenues dans d'autres. Pour cela, une méthode simple et rapide consiste à ordonner les fenêtres suivant les $a_{z,i,j}$ croissants et, en cas d'égalité, suivant les $b_{z,i,j}$ décroissants. Dans cet ordre σ, nous pouvons supprimer les fenêtres $[a_{\sigma(z),i,j} ; b_{\sigma(z),i,j}]$ pour lesquelles il existe une fenêtre $[a_{\sigma(z^*),i,j} ; b_{\sigma(z^*),i,j}]$ telle que $b_{\sigma(z),i,j} \leq b_{\sigma(z^*),i,j}$ et $1 \leq z^* < z \leq y_{i,j}$. Un tel algorithme nécessite $O(y_{i,j} \log_2 y_{i,j})$ opérations.

Algorithme générant les fenêtres non-incluses

Plutôt que de générer des périodes qui seront ensuite supprimées, l'Algorithme A.1- 3 fournit directement l'ensemble des fenêtres durant lesquelles le nombre d'exemplaires est suffisant et qui ne sont pas incluses dans d'autres fenêtres. Les données entrées sont identiques à celles de l'Algorithme A.1- 2 et les sorties donnent les fenêtres non-incluses dans l'ordre chronologique.

Algorithme A.1- 3 : Définition de fenêtres non-incluses pour chaque opération

Procédure Définition des fenêtres non-incluses durant lesquelles l'opération i du produit j peut être exécutée.

Entrée : i et j sont respectivement les indices de l'opération et du produit considéré.

R est l'ensemble des types de ressources.

R_{in} est l'ensemble des ressources interchangeables.

$l_{i,j}$ est la durée minimale de l'opération, et donc des périodes durant lesquelles l'opération peut être exécutée.

$q_{r,i,j}$ est le nombre d'exemplaires de ressources de type r nécessaires pour l'opération, pour $r \in R$.

K_r est le nombre d'exemplaires de la ressource de type r, pour $r \in R$.

$\Omega_{k,r}$ est le nombre de périodes relatives à la ressource de type r pour $k \in \{1, 2, \ldots, K_r\}$, $r \in R$.

$[\zeta_{u,k,r} ; \xi_{u,k,r}]$ est la u-ième période de l'exemplaire k, pour $u \in \{1, 2, \ldots, \Omega_{k,r}\}$, $k \in \{1, 2, \ldots, K_r\}$ et $r \in R$.

Sortie : $y_{i,j}$ est le nombre de fenêtres durant lesquelles l'opération peut être exécutée.

$[a_{z,i,j} ; b_{z,i,j}]$ est la z-ième fenêtre, où $z \in \{1, 2, \ldots, y_{i,j}\}$.

$U_{z,i,j}$ est le pointeur définissant les indices des périodes $[\zeta_{u,k,r} ; \xi_{u,k,r}]$ correspondant à la z-ième fenêtre $[a_{z,i,j} ; b_{z,i,j}]$, où $z \in \{1, 2, \ldots, y_{i,j}\}$.

1. **Initialisation.**
   - 1.1. **Poser** z:=0. *On note z l'indice de la dernière fenêtre définie.*
   - 1.2. **Poser** R^*:=$\varnothing$. *On note R^* l'ensemble des types de ressources requis.*
2. **Pour** $r \in R$: *On examine les types de ressources, un par un.*
   - 2.1. **Si** ($q_{r,i,j} \neq 0$) **alors** : *L'opération ne nécessite pas cette ressource.*
     - 2.1.1. **Poser** Q_r:=0. *On note Q_r le nombre d'exemplaires de ressource de type r requis.*
   - 2.2. **Sinon** : *L'opération nécessite cette ressource.*
     - 2.2.1. **Si** ($r \in R_{in}$) **alors** : *La ressource est interchangeable.*
       - 2.2.1.1. **Poser** Q_r:=1.
     - 2.2.2. **Sinon** : *La ressource est interchangeable.*
       - 2.2.2.1. **Poser** Q_r:=$q_{r,i,j}$. *Le nombre d'exemplaires requis est celui de l'opération.*
     - 2.2.3. **Si** ($Q_r > K_r$) **alors** : *L'opération nécessite plus d'exemplaires que le nombre existant.*
       - 2.2.3.1. **Poser** $y_{i,j}$:=0. *Aucune fenêtre n'est retenue.*
       - 2.2.3.2. **Quitter**. *L'algorithme est arrêté puisque la ressource ne peut être pourvue.*
     - 2.2.4. **Sinon** : *La ressource de type r est requise.*
       - 2.2.4.1. **Poser** R^*:=$R^* \cup \{r\}$.
3. **Si** ($R^* = \varnothing$) **alors** : *Aucune ressource n'est requise.*
   - 3.1. **Poser** z:=1.
   - 3.2. **Poser** $a_{z,i,j}$:=0 ; $b_{z,i,j}$:=+∞. *L'opération peut être exécutée dans la fenêtre* [0; +∞[.

3.3. **Poser** $U_{z,i,j}:=\emptyset$.

4. **Sinon** : *Les ressources de R^* sont nécessaires à l'opération.*

*La suite de l'algorithme diffère de l'*Algorithme A.1- 2.

4.1. **Pour** $r\in R^*$:

4.1.1. **Poser** $L_r:=\{1, 2, \ldots, K_r\}$. *On note L_r l'ensemble des exemplaires de la ressource r.*

4.2. **Pour** $r\in R^*$, **pour** $k\in L_r$:

4.2.1. **Poser** $u_{k,r}:=1$. *On note $u_{k,r}$ l'indice de la période choisie de l'exemplaire k.*

4.3. **Tant que** ($|L_r|\geq Q_r$, pour $r\in R^*$) : *Le nombre $|L_r|$ de ressource de type r est suffisant.*

4.3.1. **Pour** $r\in R^*$: *On note $u_{k,r}$ l'indice de la période choisie de l'exemplaire k.*

4.3.1.1. **Trier** les périodes $[\zeta_{u_{k,r},k,r}\ ;\ \xi_{u_{k,r},k,r}]$ pour $k\in L_r$,

dans l'ordre croissant des $\zeta_{u_{k,r},k,r}$ et, en cas d'égalité, décroissant des $\xi_{u_{k,r},k,r}$.

L'ordre ainsi défini est σ et $\sigma(k)$ est le k-ième élément dans cet ordre, $k\in L_r$.

4.3.1.2. **Poser** $U_r:=\{(u_{\sigma(k),r}, \sigma(k), r)\ /\ k\in\{1, 2, \ldots, Q_r\}\}$.

Les Q_r premières périodes sont retenues et on recherche leur intersection.

4.3.2. **Poser** $a:=\max\{\zeta_{u,k',r}\ /\ (u,k',r)\in U_r,\ r\in R^*\}$.

4.3.3. **Poser** $b:=\min\{\xi_{u,k',r}\ /\ (u,k',r)\in U_r,\ r\in R^*\}$. *La fenêtre $[a;b]$ est cette intersection.*

4.3.4. **Si** ($b-a\geq l_{i,j}$) et ($b-a>0$) **alors** : *L'opération peut être exécutée dans la fenêtre $[a;b]$.*

4.3.4.1. **Poser** $z:=z+1$.

4.3.4.2. **Poser** $a_{z,i,j}:=a$; $b_{z,i,j}:=b$. *La fenêtre $[a;b]$ est retenue.*

4.3.4.3. **Poser** $U_{z,i,j}:=\{U_r\ /\ r\in R^*\}$.

4.3.5. **Pour** $r\in R^*$, **pour** $k\in L_r$:

4.3.5.1. **Tant que** ($\xi_{u_{k,r},k,r}\leq b$ et $u_{k,r}\leq\Omega_{k,r}$) : *La période $u_{k,r}$ est contenue dans $[a;b]$.*

4.3.5.1.1. **Poser** $u_{k,r}:=u_{k,r}+1$.

4.3.5.1.2. **Si** ($u_{k,r}>\Omega_{k,r}$) **alors** : *Toute période de l'exemplaire k a été visitée.*

4.3.5.1.2.1. **Poser** $L_r:=L_r\backslash\{k\}$.

5. **Poser** $y_{i,j}:=z$. *Le nombre de fenêtres est défini.*

Exemple

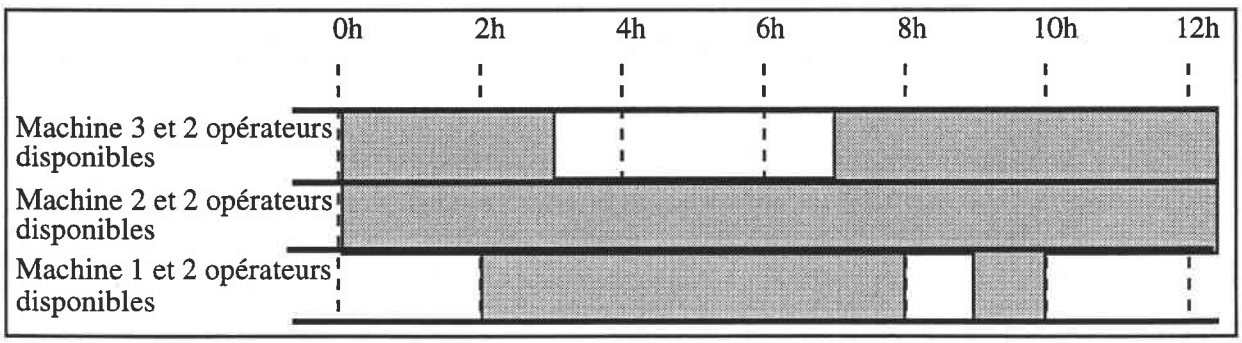

Figure A.1- 6 : Fenêtres non-incluses d'exécution d'une opération

L'Algorithme A.1- 3 se base sur les disponibilités des exemplaires $K_r=3$ de la ressource non-interchangeable de type $r'\in R_{en}$ présentées par la Figure A.1- 2. Il utilise également les disponibilités en quantité $q_{r,i,j}\geq 2$ de l'exemplaire $K_r=1$ fourni par l'Algorithme A.1- 1. L'Algorithme A.1- 3 peut alors donner les fenêtres non-incluses dans d'autres et durant lesquelles les ressources de type r et r' sont disponibles en quantité $q_{r,i,j}\geq 2$ et $q_{r',i,j}\geq 1$.

Ces $y_{i,j}$=4 fenêtres sont les suivantes : [0h ; 2h], [3h ; 7h], [8h ; 9h] et [10h ; +∞[.Elles sont représentées en blanc sur la Figure A.1- 6.

Complexité quand on génère les fenêtres non-incluses

Les pas 1 à 4 de l'Algorithme A.1- 3 requièrent $O(|R|)$ opérations, où $|R|$ est le cardinal de l'ensemble R. Les pas 4.1 à 4.2.1 requièrent $O(\sum_{r\in R^*} K_r)$ opérations, où R^* est l'ensemble de types de ressources requis. Quant au pas 4.3, il est exécuté au plus $1+\sum_{r\in R^*}\left(\Omega_{1,r}+\Omega_{2,r}+\ldots+\Omega_{K_r,r}-q_{r,i,j}\right)$ fois, puisque au moins un des indices $u_{k,r}$ est incrémenté au pas 4.3.5.1.1. Le tri au pas 4.3.1.1 nécessite $O(K_r \log_2 K_r)$ opérations [Beauquier et al., 1992] et la sélection au pas suivant nécessite $O(q_{r,i,j})$ opérations, pour tout $r\in R^*$. En fait, puisqu'on recherche les $q_{r,i,j}$ premiers éléments, ces 2 pas peuvent être exécutés en $O(q_{r,i,j} K_r)$ opérations, pour tout $r\in R^*$. Les pas 4.3.2 à 4.3.4.3 requièrent $O(\sum_{r\in R^*} q_{r,i,j})$ opérations. Les pas 4.3.5 à 4.3.5.1.2.1 nécessitent $O(\sum_{r\in R^*} K_r)$ opérations et sont exécutés au plus $1+\sum_{r\in R^*}\left(\Omega_{1,r}+\Omega_{2,r}+\ldots+\Omega_{K_r,r}-q_{r,i,j}\right)$ fois. Finalement, la complexité de l'Algorithme A.1-3 est en $O\left(|R|+\left[\sum_{r\in R^*}\left(K_r q_{r,i,j}\right)\right]\left[\sum_{r\in R^*}\left(\Omega_{1,r}+\Omega_{2,r}+\ldots+\Omega_{K_r,r}-q_{r,i,j}\right)\right]\right)$. On appelle q_0 le plus grand de ressources nécessaires pour l'ensemble des opérations. De même, K_0 est le plus grand nombre d'exemplaires de ressource non-interchangeable et Ω_0 est le plus grand nombre de périodes pour un exemplaire d'une ressource. Une borne supérieure de la complexité de l'Algorithme A.1- 3 est alors $O\left(|R|^2 K_0{}^2 q_0 \Omega_0 - |R|^2 K_0 q_0{}^2\right)$. Cette borne est atteinte si les besoins sont identiques pour chaque ressource et si tous les types de ressources comportent le même nombre d'exemplaires et, pour chacun d'eux, le même nombre de fenêtres.

Nombre de fenêtres non-incluses

Une borne supérieure du nombre $y_{i,j}$ de fenêtres générées par l'Algorithme A.1- 3 est directement donné par le nombre d'itérations du pas 4.3.4 : $1+\sum_{r\in R^*}\left(\Omega_{1,r}+\Omega_{2,r}+\ldots+\Omega_{K_r,r}-q_{r,i,j}\right)$. Cette évaluation par excès correspond au cas où, à chaque itération, une nouvelle fenêtre serait construite. Il est possible de créer des exemples pour lesquels cette borne est atteinte, c'est le cas de l'exemple de la Figure A.1- 7, où sont $y_{i,j}$=1+(3-1)+(3-1)+(3+3-1)=10 fenêtres sont générées. Sur la Figure A.1- 7b, la ressource "A1" correspond à l'exemplaire "1" de la ressource "A" de la Figure A.1- 7a. Les ressources "B1", "C1" et "C2" sont définies de manière similaire.

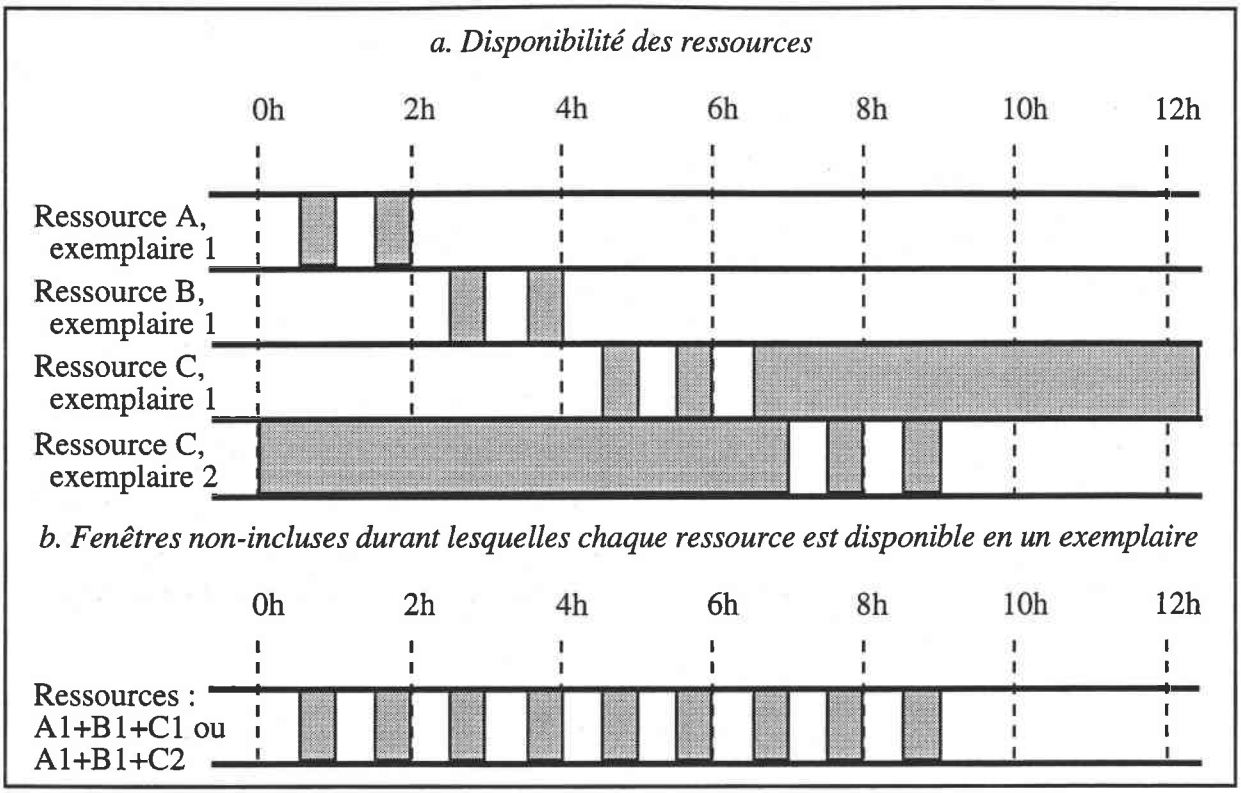

Figure A.1- 7 : Cas extrême de création de fenêtres d'exécution non-incluses

Remarques

Les périodes $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ ne se chevauchant pas et n'ayant pas d'extrémité commune, les fenêtres $[a_{z,i,j}\ ;\ b_{z,i,j}]$ sont non vides et se chevauchent éventuellement. Dans le cas où l'Algorithme A.1- 3 est appliqué, ces fenêtres sont non-incluses les unes dans les autres et répertoriées dans l'ordre chronologique. Supposons que deux opérations i et i' du produit j, $i \in I_j$ et $i' \in I_j$, nécessitent le même nombre d'exemplaires $q_{r,i,j}=q_{r,i',j}$ de chaque ressource de type r, $r \in R_{in}$. Dans ce cas, l'Algorithme A.1- 2 et l'Algorithme A.1- 3 sont appliqués une seule fois et les 2 opérations font référence au même ensemble de fenêtres.

I.3.4. Conclusion

L'ordonnancement des (j-1) premiers produits ne peut être remis en cause quand la j-ième demande apparaît. Les disponibilités des ressources sont connues précisément. Nous avons montré dans cette section comment en déduire l'ensemble des fenêtres $W_{i,j}=\{[a_{z,i,j}\ ;\ b_{z,i,j}]\ /\ 1 \le z \le y_{i,j}\}$ durant lesquelles l'opération i du produit j peut être exécutée.

I.4. MISE A JOUR DES DISPONIBILITES DES RESSOURCES

I.4.1. Introduction

Les méthodes présentées dans chaque chapitre permettent de déterminer dans quelle fenêtre $Z_{i,j}$ de type $[a_{Z_{i,j},i,j}\ ;\ b_{Z_{i,j},i,j}]$ chaque opération i du produit j doit être exécutée, ainsi que les dates de début $B_{i,j}$ et de fin $C_{i,j}$ d'exécution, $i \in I_j$. Une fois déterminé, l'ordonnancement du produit j ne peut être remis en cause. Ainsi, les disponibilités des ressources peuvent être mises à jour dès que l'ordonnancement du produit j est défini. Ce sont ces mises à jour qui sont décrites dans cette section. Les algorithmes étant de conception simple, ils ne sont pas décrits de manière formelle. En plus de ces mises à jour, il est possible de supprimer toutes les périodes de disponibilité passées (i.e. les périodes qui se terminent avant la date de commande du produit j, r_j).

I.4.2. Cas des ressources non-interchangeables

Pour mettre à jour les disponibilités de chaque ressource de type r non-interchangeable, $r \in R_{en}$, il suffit de remplacer la période $[\zeta_{u,k,r}\ ;\ \xi_{u,k,r}]$ par les périodes $[\zeta_{u,k,r}\ ;\ B_{i,j}]$ et $[C_{i,j}\ ;\ \xi_{u,k,r}]$ pour tout $(u,k,r) \in U_{Z_{i,j},i,j}$, où $r \in R_{en}$. $U_{Z_{i,j},i,j}$ a été défini par l'Algorithme A.1- 2 et l'Algorithme A.1- 3. Seules les périodes non vides sont insérées. Le nombre de périodes $\Omega_{k,r}$ est incrémenté si nécessaire. Si ces périodes sont gérées sous forme de liste chaînée, les mises à jour sont immédiates et permettent de conserver les propriétés de la structure : les périodes sont non vides, ne se chevauchent pas, n'ont pas d'intersection commune et sont répertoriées dans l'ordre chronologique. Le coût de la mise à jour est alors en $O(\sum_{r \in R^*_{en}} q_{r,i,j})$ et le nombre de périodes créées est au plus $\sum_{r \in R^*_{en}} q_{r,i,j}$, où R^*_{en} est l'ensemble des ressources non-interchangeables nécessaires à l'opération.

Supposons que seules les fenêtres non-incluses aient été retenues (i.e. les fenêtres fournies par l'Algorithme A.1- 3) avant l'ordonnancement du produit. Il est encore possible de réaffecter à chaque opération i du produit, un exemplaire $k' \in \{1, \ldots, k-1, k+1, \ldots, K_r\}$ disponible pendant la période $[B_{i,j}\ ;\ C_{i,j}]$ autre que l'exemplaire k prévu initialement. Rechercher tous ces exemplaires et les périodes disponibles coûte alors $O\left(\sum_{r \in R^*_{en}} \sum_{\substack{1 \le k' \le K_r \\ k' \ne k}} \Omega_{k',r}\right)$. Nous ne discuterons pas ici de la manière d'effectuer le choix entre les différents exemplaires disponibles. Un tel choix permet de contrôler la répartition des charges de travail entre les exemplaires.

I.4.3. Cas des ressources interchangeables

Pour mettre à jour les disponibilités de chaque ressource de type r interchangeable, $r \in R_{en}$, trois manipulations sont nécessaires.

Premièrement, il faut remplacer l'intervalle $[\alpha_{v1,r}\ ;\ \beta_{v1,r}]$, par les intervalles $[\alpha_{v1,r}\ ;\ B_{i,j}]$ de disponibilité $d_{v1,r}$ et $[B_{i,j}\ ;\ \beta_{v1,r}]$ de disponibilité $d_{v1,r}$-$q_{r,i,j}$ pour tout $(u, 1, r) \in U_{z_{i,j},i,j}$, $(v1, v2) \in V_{u,r,i,j}$, et $r \in R_{in}$. $U_{z_{i,j},i,j}$ a été défini par l'Algorithme A.1- 2 et l'Algorithme A.1- 3, tandis que $V_{u,r,i,j}$ est fourni par l'Algorithme A.1- 1.

Deuxièmement, l'intervalle $[\alpha_{v2,r}\ ;\ \beta_{v2,r}]$ est remplacé par les intervalles $[\alpha_{v2,r}\ ;\ C_{i,j}]$ de disponibilité $d_{v2,r}$-$q_{r,i,j}$ et $[B_{i,j}\ ;\ \beta_{v2,r}]$ de disponibilité $d_{v2,r}$ pour tout $(u, 1, r) \in U_{z_{i,j},i,j}$, $(v1, v2) \in V_{u,r,i,j}$, et $r \in R_{in}$.

Troisièmement, la disponibilité des intervalles $[\alpha_{v,r}\ ;\ \beta_{v,r}]$ est réduite à $d_{v,r}$-$q_{r,i,j}$ pour tout $(u, 1, r) \in U_{z_{i,j},i,j}$, $v1 < v < v2$, $(v1, v2) \in V_{u,r,i,j}$ et $r \in R_{in}$.

Seuls les intervalles non vides et de disponibilité non nulle sont conservés. Le nombre d'intervalles ω_r est incrémenté si nécessaire. Si ces intervalles sont gérés sous forme de liste chaînée, les mises à jour nécessitent au plus $O(\omega_r)$ opérations et permettent de conserver les propriétés de la structure : les périodes sont non vides, ne se chevauchent pas, ont au plus leur extrémité commune et sont répertoriées dans l'ordre chronologique. Le coût de la mise à jour est alors en $O(\sum_{r \in R^*_{in}} \omega_r)$ et le nombre de périodes créées est au plus $2|R^*_{in}|$, où R^*_{in} est l'ensemble des ressources interchangeables nécessaires à l'opération.

I.4.4. Conclusion

Ainsi, les disponibilités des ressources sont mises à jour à partir de l'ordonnancement du produit j. Quand la $(j+1)$-ième demande apparaîtra, on utilisera l'Algorithme A.1- 1, l'Algorithme A.1- 2 et l'Algorithme A.1- 3 pour déterminer à nouveau les fenêtres de temps durant lesquelles les opérations de ce produit peuvent être exécutées.

I.5. CONCLUSION

Cette annexe présente les structures supportant les données relatives aux disponibilités des ressources, les algorithmes permettant de définir les périodes durant lesquelles chaque opération peut être exécutée et les méthodes de mise à jour des disponibilités des ressources.

En effet, à partir des disponibilités de chaque ressource restant après les $(j\text{-}1)$ premiers produits ordonnancés, nous pouvons déduire les fenêtres de temps durant lesquelles chaque opération i du produit j peut être exécutée. Il s'agit des périodes durant lesquelles les disponibilités de la ressource de type r sont supérieures ou égales aux $q_{r,i,j}$ unités requises pour l'exécution de l'opération i du produit j. Une fois le produit j ordonnancé, il faut également remettre à jour les disponibilités des ressources selon les utilisations définies pour le produit j.

Répétons que ces procédures de gestion des fenêtres de disponibilité des ressources sont relatives à chaque opération. Par conséquent, elles sont indépendantes des types de gammes des produits et peuvent être utilisées indifféremment que la gamme du produit soit linéaire ou non.

ORDONNANCEMENT EN TEMPS RÉEL DANS LES PROBLÈMES À EN-COURS LIMITÉS

Thèse de Doctorat soutenue par Fabrice CHAUVET

Résumé - Dans cette thèse, nous montrons l'intérêt d'étudier des systèmes de production présentant les deux caractéristiques suivantes. Tout d'abord, les durées des opérations à réaliser sont choisies dans des intervalles donnés - de tels systèmes de gestion de fabrication sont dits à temps opératoires contrôlables -. De plus, les opérations successives réalisées sur chaque produit se suivent sans temps d'attente - de tels systèmes sont dits sans attente -.

Ces deux caractéristiques permettent de gérer en temps réel un large éventail de systèmes de fabrication. En outre, cette approche permet le contrôle des en-cours et des temps de fabrication.

La productivité du système de fabrication est optimisée grâce à des algorithmes originaux. L'analyse de la complexité des dix-sept algorithmes proposés prouve leur compatibilité avec leur utilisation en temps réel.

Mots-clefs - Gestion de production, ordonnancement, temps réel, systèmes automatisés, stockage interdit, sans attente, stockage limité, ressources multiples.

CONSTRAINED WORK-IN-PROCESS IN ON-LINE SCHEDULING

PhD Thesis by Fabrice CHAUVET

Abstract - In this thesis, a production system with two characteristics is considered. First, the processing time of an operation to be performed in a given time interval can be selected. Such a production system is said to have controllable processing times. Second, the operations of a product follow each other without waiting. A production system having this characteristic is said to be no-wait.

The two characteristics are key in scheduling of many different types of industrial systems. The proposed methodology allows one to control the work-in-process and production cycle time.

The system productivity is optimised with original algorithms developed in this research. The analysis of computational complexity of the seventeen algorithms shows that they can be used for on-line scheduling.

Key words - Production management, on-line scheduling, automated guided vehicle, zero WIP system, no-wait system, limited storage, multiple resources.

INRIA-Lorraine
Université de Metz - Faculté des sciences
UFR Mathématiques, informatique, mécanique
Ile du Saulcy - 57000 Metz

UNIVERSITE DE METZ
U.F.R. SCIENTIFIQUES

DOCTORAT
DE L'UNIVERSITE DE METZ

Thèse soutenue le 20 septembre 1999
par Monsieur CHAUVET Fabrice

Signatures des membres du jury :

- M. CHU C. Excusé
- M. CLAVER J.F. Excusé
- M. FINKE G.
- M. HAOUBA A.
- M. PRINS C.
- M. PROTH J.M.
- M. WAGNEUR E.

RAPPORT APRES SOUTENANCE

sur la thèse ayant pour sujet :

« Ordonnancement en temps réel dans les problèmes à en-cours limité. »

Les membres du jury, à l'unanimité, tiennent à souligner l'excellence de la présentation et la clarté de l'exposé très didactique mettant en valeur une approche innovante.

Les membres du jury tiennent également à souligner l'ampleur du champ exploré et la qualité des résultats obtenus, qualité reconnue pour le grand nombre de publications dans des revues de premier plan.

La pertinence des réponses aux questions du jury a fait la preuve d'une grande maîtrise du sujet de la part du candidat.

Cette thèse présente des résultats importants tant par leurs aspects théoriques que par leur potentiel d'utilisation industrielle.

Pour toutes ces raisons, le jury a décidé que Monsieur Fabrice CHAUVET est digne du Diplôme de Docteur de l'Université de Metz, lequel diplôme lui a été décerné avec la mention très honorable avec félicitations.

UNIVERSITE DE METZ

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Nom et Prénom de l'auteur : Monsieur CHAUVET Fabrice

Titre de la thèse : « Ordonnancement en temps réel dans les problèmes à en-cours limité. »

Date de soutenance : le 20 septembre 1999

Président du jury : Edouard Wagneur

Membres du jury : M.M. CHU, CLAVER, FINKE, HAOUBA, PRINS, PROTH et WAGNEUR

Reproduction de la thèse soutenue :

☒ thèse pouvant être reproduite en l'état

☐ thèse ne pouvant être reproduite

☐ thèse pouvant être reproduite après corrections suggérées au cours de la soutenance

Le Président du Jury

Corrections effectuées le :

Le Directeur de Thèse