N° d'ordre : Année 1999

THESE

présentée à

L'UNIVERSITE DE METZ, FACULTE DES SCIENCES
UFR MATHEMATIQUES, INFORMATIQUE, MECANIQUE

pour obtenir le titre de

DOCTEUR

Spécialité : Sciences de l'Ingénieur
Mention : Automatique

par

Berenice CAMARGO DAMASCENO

ORDONNANCEMENT DES SYSTEMES DE PRODUCTION MULTI-RESSOURCES AVEC LA PRISE EN COMPTE DE BLOCAGE

Soutenue le 4 mars 1999, devant le Jury :

M. Christian PROUST	Président	: Professeur
M. Pierre BAPTISTE	Rapporteur	: Professeur
M. Stephane DAUZERE-PERES	Rapporteur	: Maître Assist. à l'Ecole de Mines, HDR
M. Jean-Claude GENTINA	Rapporteur	: Professeur
M. Xiaolan XIE	Directeur de thèse	: Chargé de Recherche à l'INRIA, HDR
M. Yves DALLERY	Examinateur	: Directeur de Recherche au CNRS
M. Luis A. PEREIRA de OLIVEIRA	Examinateur	: Professeur

Thèse pr dustriel et de Production Mécanique
urs de Metz -

N° d'ordre : Année 1999

THESE

présentée à

L'UNIVERSITE DE METZ, FACULTE DES SCIENCES
UFR MATHEMATIQUES, INFORMATIQUE, MECANIQUE

pour obtenir le titre de

DOCTEUR

Spécialité : Sciences de l'Ingénieur
Mention : Automatique

par

Berenice CAMARGO DAMASCENO

ORDONNANCEMENT DES SYSTEMES DE PRODUCTION MULTI-RESSOURCES AVEC LA PRISE EN COMPTE DE BLOCAGE

Soutenue le 4 mars 1999, devant le Jury :

M. Christian PROUST	Président	: Professeur
M. Pierre BAPTISTE	Rapporteur	: Professeur
M. Stephane DAUZERE-PERES	Rapporteur	: Maître Assist. à l'Ecole de Mines, HDR
M. Jean-Claude GENTINA	Rapporteur	: Professeur
M. Xiaolan XIE	Directeur de thèse	: Chargé de Recherche à l'INRIA, HDR
M. Yves DALLERY	Examinateur	: Directeur de Recherche au CNRS
M. Luis A. PEREIRA de OLIVEIRA	Examinateur	: Professeur

Thèse préparée au sein du Laboratoire de Génie Industriel et de Production Mécanique
- Ecole Nationale d'Ingénieurs de Metz -

A ma mère
Benedicta Camargo Damasceno

pour son amour et son existence

A ma famille

pour sa présence

Remerciements

Les remerciements ont leur place quand nous sommes entourés de personnes amies qui grâce à leur soutien, leurs encouragements, leur compétence, leur "présence" ont nous permis d'arriver à la fin de notre travail. C'est exactement mon cas. Je souhaite remercier chacun et chacune de l'ensemble des personnes qui m'ont aidée à finaliser ma thèse. J'espère que les mots, même s'ils sont insuffisants pour exprimer mes sentiments, pourront montrer que tous et toutes que je vais mentionner ci-dessous (et éventuellement, les personnes qui, faute de mémoire ne seront pas là explicitement) auront toujours une place dans mes souvenirs et dans mon cœur.

D'abord, je tiens à remercier ceux qui m'ont fait l'honneur de faire partie de mon jury de thèse :

- *Monsieur Pierre Baptiste, Professeur à l'Université de Besançon : qu'il trouve ici toute ma reconnaissance d'avoir accepté la lourde tâche d'être rapporteur. Je le remercie également pour les discussions pendant la phase finale de mon travail.*

- *Monsieur Yves Dallery, Directeur de recherche au CNRS au Laboratoire d'Informatique de Paris 6 à l'Université Pierre et Marie Curie, qui a accepté de participer à ce jury.*

- *Monsieur Stephane Dauzère-Perez, Maître Assistant à l'Ecole de Mines de Nantes, qui a accepté d'être rapporteur. Je le remercie également pour son excellent travail de recherche, sa thèse et ses articles, qui m'ont beaucoup inspirée et aidée.*

- *Monsieur Jean-Claude Gentina, Directeur de l'Ecole Centrale de Lille, qui a accepté d'être rapporteur. Je le remercie pour le temps qu'il a pu consacrer à cette lourde tâche.*

- *Monsieur Christian Proust, Professeur de l'Université de Tours et Directeur de l'Ecole d'Ingénieurs en Informatique pour l'Industrie à l'Université de Tours, qui a accepté de présider ce jury et je le remercie pour sa gentillesse et l'intérêt qu'il a porté à mon travail.*

Je ne saurais oublier ici Monsieur Xiaolan Xie, Chargé de Recherche à l'INRIA et directeur de ma thèse. Mes remerciements seront toujours insuffisants. Je le remercie pour avoir apporté sa compétence dans le domaine où s'insère cette thèse. Je le remercie également pour tout le précieux temps qu'il m'a consacré, ses conseils et sa confiance. Cette thèse est arrivée à sa conclusion grâce à son aide indéniable. Je le remercie beaucoup.

Je pense aussi à Monsieur Jean-Marie Proth, Directeur de Recherche à l'INRIA qui m'a accueillie au sein du projet SAGEP. Je pense également à tous ceux qui ont fait partie de ce projet et je remercie tout particulièrement Christel Wiemmert pour son amitié et son soutien.

Je remercie Monsieur Pierre Padilla, Directeur de l'Ecole Nationale d'Ingénieurs de Metz (ENIM), pour m'avoir accueillie dans son établissement, pour son encouragement, sa présence et son extrême gentillesse.

Je remercie également Monsieur François Vernadat, Professeur à l'Université de Metz, pour m'avoir accueillie au sein du Laboratoire de Génie Industriel et Production Mécanique (LGIPM) et pour avoir lue minutieusement cette thèse.

Je tiens à remercier aussi les chercheurs du laboratoire LGIPM et, plus spécialement Brigitte Finel.

Je tiens tout spécialement à exprimer toute ma reconnaissance à Madame Marie-Noëlle Honnert, Enseignante à la Faculté des Lettres à l'Université de Metz, pour avoir accepté de faire des corrections grammaticales et orthographiques dans cette thèse et pour m'avoir apporté sa connaissance de la langue française.

Je remercie aussi les responsables de l'Ecole "Escola Técnica Estadual Rubens de Faria e Souza" (ETERFS), de le Centre "Centro Estadual de Educação Técnológica Paula Souza" (CEETEPS) et de le Conseil "Conselho Nacional de Desenvolvimento Científico e Tecnológico" - Brésil - (CNPq) pour avoir rendu possible mon séjour en France.

Il existe des personnes particulières qui grâce à leur présence, même à distance, nous soutiennent et nous aident à continuer notre travail. Pour moi, il y a d'un côté Anirio Salles Filho, un ami qui, même s'il est resté au Brésil, a toujours été à mes côtés. De l'autre Hamid Hamouche qui a été présent tout au long de mon séjour en France.

Je réserve une place spéciale dans mes remerciements aux amis, "non moins spéciaux", Giuseppe Berio, Fabrice Chauvet, Nevine Haffez, Mounira Harzallah et Cathy Michel pour leur soutien et leur amitié.

Je tiens tout spécialement à exprimer de tendres pensées à Regina Célia Farrabotti et à mon frère Deoclécio Damasceno pour m'avoir apporté leur aide indispensable pour rendre mon séjour ici possible.

Peu importe l'ordre de mes remerciements. Tous ceux que j'ai nommé (et tous ceux que je n'ai pas pu nommer par manque de place) m'ont apporté, à un moment ou un autre, un soutien important et décisif.

TABLE DES MATIERES

CHAPITRE II : JOB-SHOPS MULTI-RESSOURCES AVEC BLOCAGE : DÉFINITION ET ÉTUDE DE LA COMPLEXITÉ

CHAPITRE III : ORDONNANCEMENT DES SYSTÈMES À RESSOURCES UNITAIRES

LISTE DES FIGURES

CHAPITRE I : ORDONNANCEMENT DES SYSTEMES FLEXIBLES DE PRODUCTION : ETAT DE L'ART

CHAPITRE II : JOB-SHOPS MULTI-RESSOURCES AVEC BLOCAGE : DÉFINITION ET ÉTUDE DE LA COMPLEXITÉ

CHAPITRE III : ORDONNANCEMENT DES SYSTÈMES À RESSOURCES UNITAIRES

CHAPITRE IV : ORDONNANCEMENT DES SYSTÈMES À RESSOURCES MULTIPLES

CHAPITRE V : LE PROBLÈME D'ORDONNANCEMENT AVEC MOYENS DE TRANSPORT

GLOSSAIRE DES NOTATIONS UTILISEES

Les notations utilisées dans cette thèse sont définies brièvement ci-dessous.

m : le nombre de types de ressources

M : le nombre de types de machines

L : le nombre des AGV

$R = \{1, 2, ..., r, ..., m\}$: l'ensemble des types de ressources

H_r : le nombre de ressources du type r

$\mathbf{H} = [H_1, H_2, ..., H_r, ..., H_m]^T$: le vecteur du nombre de ressources

n : le nombre de travaux

$J = \{J_1, J_2, ..., J_j, ..., J_n\}$: l'ensemble des travaux

O_{jk} ou (j,k) : la $k^{ème}$ opération du travail j

$\{O_{j1}, O_{j2}, ..., O_{jk}, ..., O_{jNj}\}$: la séquence d'opérations du travail j

N_j : le nombre d'opérations du travail j

h_{jkr} : le nombre de ressources du type r nécessaires pour l'opération (j,k). Evidement, $h_{jkr} \leq H_r$

$\mathbf{h}_{jk} = [h_{jk1}, h_{jk2}, ..., h_{jkr}, ..., h_{jkm}]$: le vecteur de besoins en ressources de l'opération (j,k)

$R_{jk} = \{r \in R / h_{jkr} > 0\}$: l'ensemble des types de ressources nécessaires pour l'opération (j,k)

$M_{jk} \in \{1, 2, \ldots, M\}$: la machine utilisée pour l'opération (j,k)

p_{jk} : le temps opératoire de l'opération (j,k)

S_{jk} : la date de début de l'opération (j,k)

C_{jk} : la date de fin de l'opération (j,k)

C_j : la date de fin du travail j

d_j : la date échue du travail j

H : l'horizon de temps (notation utilisée dans le chapitre V)

$J_j = \{(R_{j1}, \mathbf{h}_{j1}, p_{j1}) , (R_{j2}, \mathbf{h}_{j2}, p_{j2}) , ..., (R_{jNj}, \mathbf{h}_{jNj}, p_{jNj})\}$: la gamme de fabrication d'un travail j d'un job-shop MRB

I_{kt} : le nombre de ressources nécessaires pour l'opération (j,k) qui sont retenues dans la période t par les travaux déjà ordonnancés

B(k,t) : l'indicateur pour vérifier la possibilité de passer de l'opération (j,k) à l'opération (j, k+1) sans risque de blocage au début de la période t

J : le critère à optimiser

w_j : la pondération du travail j

T_j : le retard du travail j, défini par : $T_j = \max(0, C_j - d_j)$

$$1\{x\} = \begin{cases} 1 & \text{si } x \text{ est vraie,} \\ 0 & \text{sinon.} \end{cases}$$

$$\delta_{jk}^{m} = \begin{cases} 1 & \text{si } Mjk = m, \\ 0 & \text{sinon.} \end{cases}$$

LISTE DES SIGLES UTILISEES

Les sigles utilisées dans ce mémoire sont définies brièvement ci-dessous.

AGV : *Automated Guided Vehicles*

Job-shops MRB : Job-shops Multi-Ressources avec Blocage

SRU : Système à Ressources Unitaires

SRM : Système à Ressources Multiples

Opération ZD : Opération *Zero Delay*

Opération ZRZD : Opération *Zero Resource Zero Delay*

Opération ZRD : Opération *Zero Resource with Delay*

Opération SRSU : Opération *Single Resource Single Unit*

Opération SRMU : Opération *Single Resource Multi-Units*

Opération MRSU : Opération *Multi-Resource Single Unit*

Opération MRMU : Opération Multi-Ressources et Multi-Unités

Opération SU : Opération *Single Unit*

Opération MU : Opération *Multi-Units*

G-opération : opération *get*

R-opération : opération *release*

RdP : Réseau de Petri

Introduction générale

Introduction générale

La compétition à l'échelle mondiale a changé les donnes de l'économie. En production industrielle, le mot d'ordre est la compétitivité. Par ailleurs, le marché a sensiblement évolué et on est passé d'une économie d'échelle à une économie d'envergure. Pour être compétitive, une entreprise doit automatiser, intégrer et être flexible afin d'améliorer sa productivité, la qualité de ses produits, le délai et le coût de la production.

La réponse technologique tend vers les systèmes flexibles alliant la productivité et la flexibilité. De plus en plus, les entreprises automatisent partiellement, voire entièrement, leurs moyens de production clés. Les moyens de production sont aujourd'hui très riches et variés. On trouve, dans un système de production moderne, des machines à commande numérique, une grande variété de moyens de transport automatisés (convoyeurs, AGV, ponts roulants), des robots pour la manutention, l'usinage, l'assemblage, etc., les installations automatisées très sophistiquées pour le stockage des produits et un nombre impressionnant d'outils permettant la flexibilité des machines.

L'investissement souvent très lourd dans ces moyens de production serait vain sans un système de gestion efficace. En effet, la mise en jeu d'une grande variété de ressources pose le délicat problème de la coordination de l'ensemble des ressources. Dans cette thèse, nous nous donnons comme objectif de proposer de méthodes pour une gestion efficace et intégrée de l'ensemble des ressources.

Comme Ramaswany et Joshi (96) l'ont constaté, la planification et le contrôle d'un système automatisé de production diffèrent considérablement des problèmes de planification et d'ordonnancement étudiés dans la littérature traditionnelle. La plupart des travaux existants ne prennent en compte que les produits et les machines et ignorent la variété des ressources mises en jeu dans un système de production. On espère que l'ordonnancement établi avec les seules contraintes des machines peut être réalisé par un système de pilotage sans trop dégrader les performances. La réalité est loin d'être si simple. En pratique, la gestion des moyens de manutention est très délicate et une mauvaise gestion peut perturber, de manière significative, la production. Dans certains systèmes avec des stocks tampons de capacité très faible, l'ordonnancement obtenu avec les seules contraintes des machines peut conduire au blocage mortel du système.

Dans cette thèse, nous proposons une approche intégrée d'ordonnancement qui prend en compte simultanément la variété des ressources et le problème de blocage. Les deux caractéristiques saillantes du modèle que nous proposons sont : (i) les opérations nécessitant simultanément des ressources de différents types que nous appelons opérations à ressources multiples ; (ii) la contrainte "retenir et attendre" pour le passage d'une opération à l'opération suivante du même travail, c'est-à-dire que les ressources nécessaires pour une opération ne sont libérées qu'au début de l'opération suivante. La caractéristique (i) permet la prise en considération de l'ensemble de ressources. La caractéristique (ii) correspond mieux à la réalité industrielle que l'hypothèse utilisée dans les travaux existants supposant que les

ressources sont libérées à la fin de chaque opération, ceci quel que soit l'état des ressources nécessaires pour l'opération suivante.

Cette thèse est composée de cinq chapitres. Le premier chapitre est une étude bibliographique de l'ordonnancement des systèmes flexibles de production. Après une brève introduction des problèmes d'ordonnancement des systèmes de production, nous exposons, de manière concise, différentes techniques classiques d'ordonnancement. Ensuite, nous présentons un échantillon représentatif des travaux sur l'ordonnancement des systèmes de production tels que l'ordonnancement des opérations à ressources multiples, l'ordonnancement sans blocage d'un FMS, l'ordonnancement des Réseaux de Petri (RdP) et ses applications aux systèmes de production, l'ordonnancement de robots de manutention en galvanoplastie, l'ordonnancement sans attente et sans en-cours et l'ordonnancement des cellules robotisées. Nous abordons également les travaux sur la détection, la prévention, l'élimination et la correction des blocages.

Le deuxième chapitre est consacré à la définition des systèmes considérés dans cette thèse et à l'étude de la complexité du problème d'ordonnancement correspondant. D'abord, nous introduisons les job-shops Multi-Ressources avec Blocage (job-shops MRB), une extension des job-shops enrichis des deux caractéristiques mentionnées ci-dessus, c'est-à-dire des opérations à multi-ressources et la propriété "Retenir et Attendre" pour la succession des opérations. Nous montrons que les job-shops MRB fournissent un cadre général pour étudier l'ordonnancement des systèmes industriels. La dernière partie de ce chapitre est une étude détaillée de la complexité du problème d'ordonnancement des job-shops MRB, utilisant les résultats existants pour des cas étudiés dans la littérature et développant des résultats nouveaux pour les autres cas.

Le troisième chapitre est dédié à l'ordonnancement des Systèmes à Ressources Unitaires (SRU), les job-shops MRB dans lesquels les ressources sont toutes différentes, c'est-à-dire dans lesquels il n'y a pas de ressources identiques. Nous introduisons la notion de systèmes simples, des job-shops MRB dans lesquels aucun travail (ou produit) n'attend la disponibilité de nouvelles ressources pour libérer les ressources qui ne sont plus nécessaires. Nous montrons que les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage pour un système simple alors que ceci est faux pour un SRU général. Ensuite, nous modélisons les SRU à l'aide des réseaux de Petri et utilisons le modèle RdP pour la vérification de blocage. Deux méthodes heuristiques gloutonnes utilisant le modèle RdP pour la vérification de blocage sont proposées pour construire des ordonnancements sans blocage. Nous proposons également une méthode d'ordonnancement d'un SRU général en passant par un système simple "équivalent". L'avantage de cette méthode est qu'elle n'a pas besoin de la vérification de blocage. Les résultats numériques présentés dans ce chapitre sont encourageants.

Le quatrième chapitre est consacré à l'ordonnancement des job-shops MRB généraux. Comme pour les SRU, nous montrons que les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage pour un système simple. Nous modélisons les job-shops MRB à l'aide des Réseaux de Petri généralisés avec poids. Nous montrons que la vérification de blocage est combinatoire pour un job-shop MRB général alors qu'elle est de complexité linéaire pour un SRU. Ceci montre l'importance des systèmes simples. Le modèle RdP permet la détermination d'un ordonnancement au plus tôt étant donné l'ordre de passage

des opérations dans les ressources; ce résultat sera exploité dans l'ordonnancement d'un job-shop MRB général. Ensuite, nous généralisons les méthodes heuristiques du chapitre III pour obtenir l'ordonnancement sans blocage des job-shops MRB généraux. Dans la suite, les résultats numériques sont présentés.

Le cinquième chapitre considère une classe particulière de job-shops MRB : les job-shops classiques dans lesquels chaque travail a besoin d'un moyen de transport (AGV) pour l'accompagner tout au long de sa réalisation. L'objectif de ce chapitre est l'étude de l'efficacité de la méthode Relaxation Lagrangienne pour l'ordonnancement des job-shops MRB selon la méthodologie proposée par Luh et Hoitomt (93). Nous proposons une méthode Programmation Dynamique pour la résolution du problème relaxé, une méthode de sous-gradients pour la résolution du problème dual et des heuristiques pour la construction d'un ordonnancement réalisable à partir de la solution du problème dual. Les résultats numériques actuels montrent que la méthode Relaxation Lagrangienne est une méthode à prendre en compte pour l'ordonnancement des job-shops MRB.

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

Chapitre I

Ordonnancement des systèmes flexibles de production : état de l'art

I.1 Introduction aux problèmes d'ordonnancement

I.1.1 La production

La production peut être définie comme une opération de transformation qui convertit des matières premières et/ou composants en produits finis.

Dans une entreprise, la production dépend d'autres fonctions. Nous présentons, dans la figure I.1, un schéma simplifié de la production dans son environnement immédiat (Bénassy, 87).

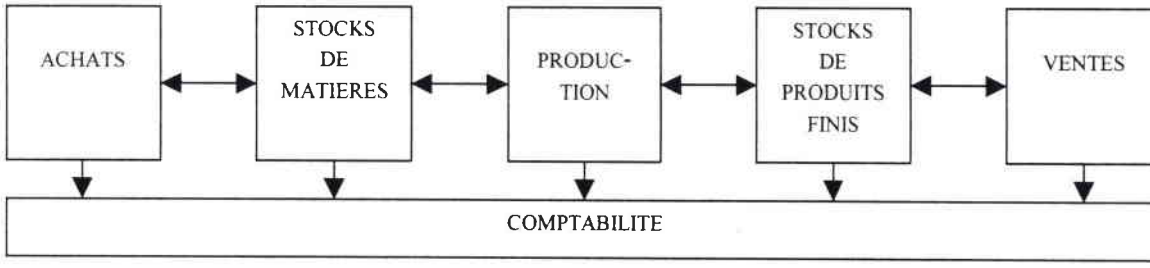

Figure I.1 : Position de la production

La production a quatre objectifs :

- un objectif de volume qui consiste à satisfaire la demande,
- un objectif de qualité sur les produits,
- un objectif de délai,
- un objectif de coûts.

Ces objectifs ne sont pas indépendants et le problème principal consiste à arbitrer entre ces exigences, afin d'arriver à un compromis satisfaisant pour le client et pour l'entreprise.

I.1.2 Le système de production

Un système de production est composé d'un ensemble de ressources qui permet de réaliser l'objectif de la production, c'est-à-dire la transformation de matières premières et/ou de composants en produits finis. On distingue quatre types de ressources : les équipements, les hommes, les matières et les informations techniques, procédurales ou relatives à l'état et à l'utilisation du système productif.

On trouve, dans Giard (88), deux types de typologie des systèmes de production : la première est axée sur le fait qu'une production peut être réalisée soit pour répondre à une

commande, soit pour alimenter un stock et la seconde est liée au mode d'organisation de la production, qui diffère selon l'importance et la variété des flux de produits traités par les systèmes productifs.

Nous présentons, dans la figure I.2, la typologie de systèmes de production de Noodward (Le Moal et Tarondeau, 79) qui inclut les types mentionnés avant.

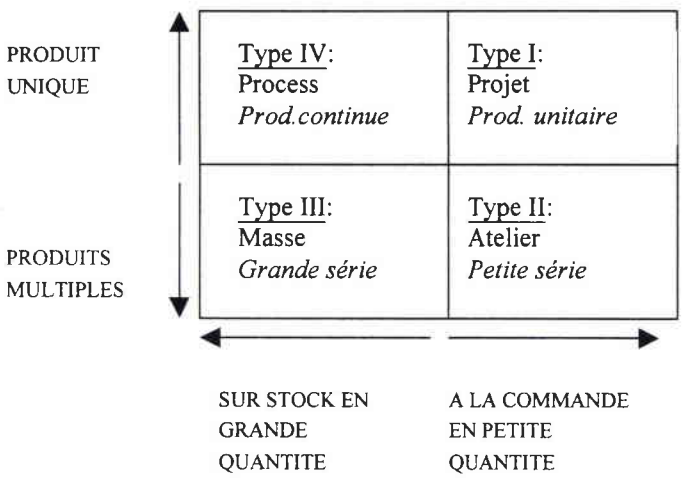

Figure I.2 : Typologie de Systèmes de Production

A cause de la faible durée de vie des produits, de la grande diversification de la demande, qui devient en plus incertaine et exigeante, les entreprises se trouvent obligées d'innover et de renouveler fréquemment leurs gammes de produits. Ainsi, on remarque que les productions de type "masse" et "process" voient leur volume diminuer au profit de productions du type "atelier".

I.1.3 La gestion de production

L'objectif de la gestion de production est d'assurer la fabrication d'une quantité donnée d'un produit donné, dans les délais prévus, avec la qualité demandée, au meilleur coût, en utilisant au mieux les moyens disponibles et en maintenant les en-cours et les stocks au minimum.

Les interactions entre le système de gestion, le système de production et son environnement sont schématisées dans la figure I.3.

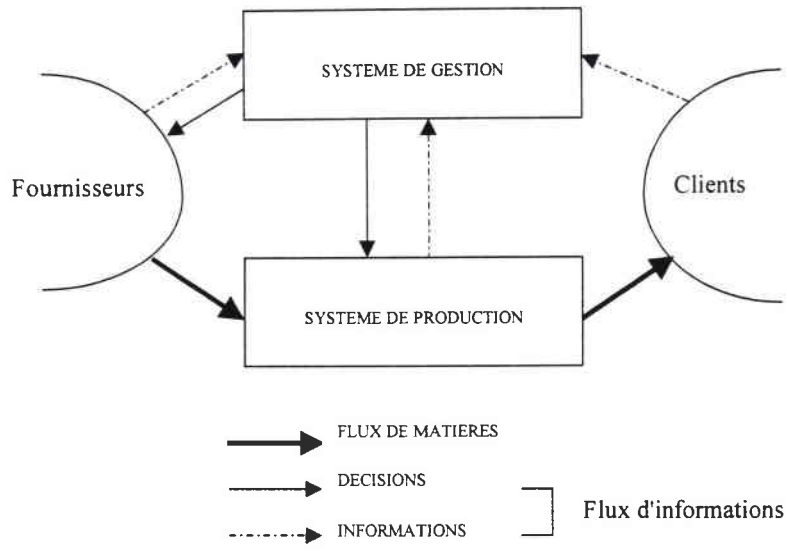

Figure I.3 : Interactions entre système de gestion, système de production et environnement

D'un côté, les fournisseurs alimentent le système de production suivant les décisions du système de gestion. Les matières circulent ensuite dans l'atelier de production entre les ressources et les stocks disponibles. Enfin, les produits finis sont stockés jusqu'à la livraison aux clients. Le système inclut parfois les décisions concernant les transports de matériel, en amont et en aval de la production.

Le système de gestion reçoit les demandes des clients et, en fonction de celles-ci, quantifie la production tout en prenant les contraintes des fournisseurs en compte. De plus, il gère le système de production en tenant compte des aléas par un retour d'informations.

Les décisions prises par le système de gestion sont de plusieurs types et s'appliquent à différents niveaux. De nombreux travaux ont été consacrés à la gestion hiérarchisée des systèmes de production : Antony (65), Bitran et Hax (77), Gershwin (89), Xie (89). On classe habituellement les décisions de gestion en trois catégories :

- les décisions stratégiques qui traduisent la politique générale de l'entreprise. Les décisions prises à ce niveau sont à long terme. Elles concernent les objectifs principaux de l'entreprise au niveau investissements, nouvelles orientations (produits nouveaux, techniques récentes), etc.

- les décisions tactiques qui assurent la gestion du travail et des moyens de production dans le cadre des décisions prises au niveau stratégique. Elles visent à définir ce qu'il faut produire en fonction des clients, des stocks, de l'environnement, etc. Ces décisions consistent généralement à définir un plan de production.

- Les décisions opérationnelles qui régissent en détail la fabrication proprement dite, dans le respect des décisions tactiques. Parmi les décisions opérationnelles concernant la gestion de la production, on trouve, en particulier, celles qui assurent le contrôle des matières (gestion de stock) et celui de la main-d'œuvre et des équipements (ordonnancement).

La figure I.4 représente la hiérarchie des différents niveaux. Toute décision prise à un niveau devient une contrainte pour le niveau inférieur. Si celui-ci ne peut pas satisfaire cette contrainte, il en informe le niveau supérieur qui remet en question sa décision s'il y a lieu.

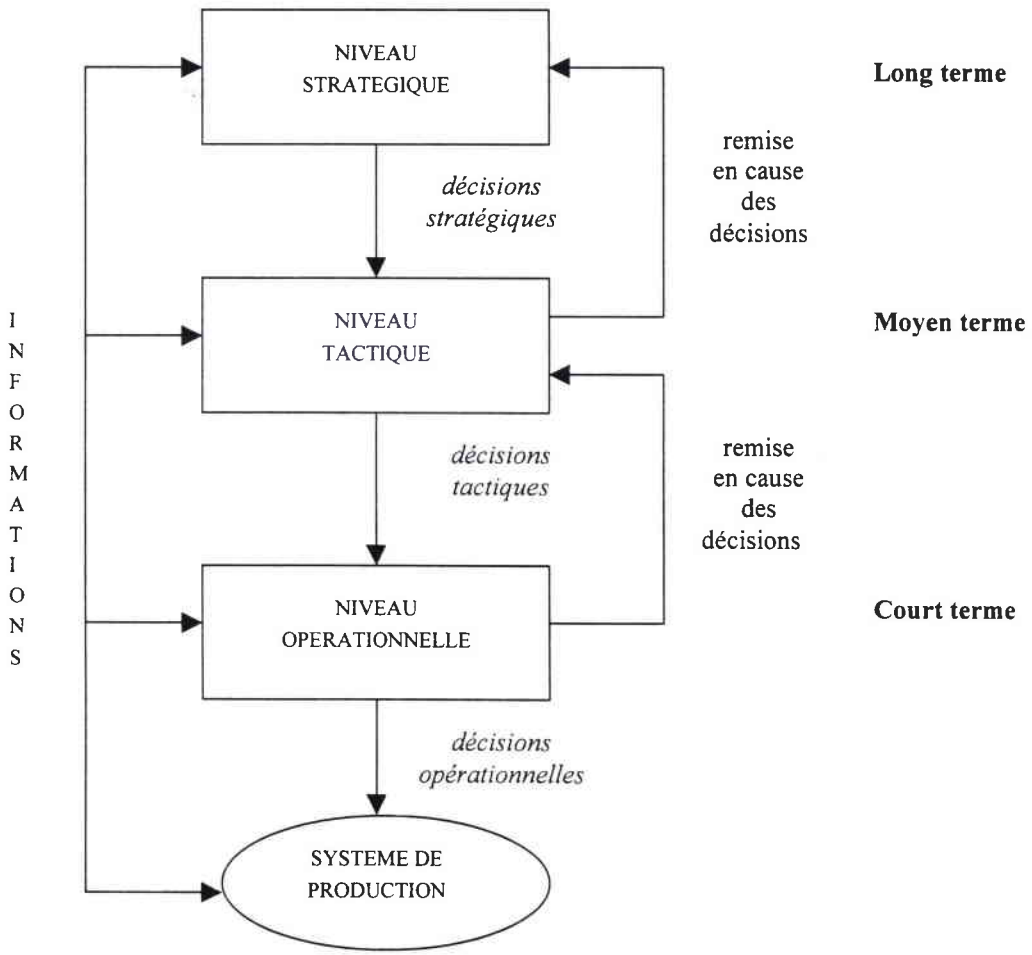

Figure I.4 : Structure décisionnelle à trois niveaux

Ces trois classes de décisions ne diffèrent pas seulement par l'horizon qui les caractérise. Deux éléments supplémentaires doivent être pris en compte : le niveau de compétence hiérarchique de la personne qui prend la décision et celui d'agrégation de la décision, c'est-à-dire le niveau de détail des décisions prises quant aux productions à effectuer et aux moyens à mettre en œuvre.

I.1.4 Ordonnancement de la production

Dans la suite de ce paragraphe, nous nous intéressons plus particulièrement au niveau opérationnel, plus précisément à l'ordonnancement à court terme qui planifie le travail dans un atelier de production.

Un atelier de production est composé de ressources, telles les machines et les ressources humaines. De plus en plus, la flexibilité joue un rôle important dans les entreprises. De ce fait, nous avons maintenant des ateliers flexibles. Les ateliers flexibles constituent un compromis entre la productivité et la flexibilité. Un atelier flexible permet la production économique de petites ou moyennes séries pour suivre une demande fluctuante, sans pour autant devoir gérer d'importants stocks de produits finis.

Les ressources qui constituent un atelier flexible doivent être capables d'effectuer des opérations différentes sur des produits différents. L'atelier flexible est composé de différentes types de ressources comme, par exemple, des machines à commande numérique, des outils, des ressources humaines, des robots et des AGV (*Automated Guided Vehicles*).

Un atelier flexible contient donc un ensemble de ressources distinctes sur lesquelles un ensemble de travaux (*jobs*) doit être exécuté. Chaque travail nécessite un ensemble d'opérations, chacune d'elles devant être exécutée sur une ou plusieurs machines en utilisant des ressources humaines et/ou moyens de transports et/ou outils.

Traditionnellement, on utilise un système de notation pour représenter les problèmes d'ordonnancement. Ce système contient trois champs comme suit :

Type d'atelier/ caractéristiques des opérations/ critère choisi

où :

- Le **type d'atelier** décrit le mode d'organisation et l'environnement du système de production. Dans la plupart des études, les seules ressources prises en compte sont les machines. La table I.1 nous donne quelques exemples classiques d'atelier.

Type	Signification
1	1 machine
Pm	m machines à performances identiques ou machines parallèles
Fm (*Flow Shop)*	m machines (les tâches utilisent toutes les machines et dans le même ordre)
Jm (*Job Shop)*	m machines (les tâches utilisent toutes les machines et dans un ordre différent)
Om (*Open Shop)*	m machines (les gammes sont quelconques)

Table I.1 - Exemples des types d'atelier

Dans cette table, nous avons :

- un **open shop ou atelier à cheminements quelconques** : quand les opérations sont indépendantes ;
- un **flow shop ou atelier à cheminement unique** : quand les opérations des produits suivent le même ordre de production ;
- un **job shop ou atelier à cheminements multiples** : quand les opérations d'un produit suivent un ordre de production, non nécessairement identique pour tous les produits.

- **Les caractéristiques des opérations** les plus usuelles sont :

  - **Les contraintes potentielles** qui peuvent être de deux types :

    - **Les contraintes de succession** entre deux opérations i et j, qui expriment le fait que le début de l'opération i et le début de l'opération j sont décalés dans le temps ;
    - **Les contraintes de localisation temporelle** qui imposent l'exécution d'une opération dans une fenêtre de temps décrite par sa date de disponibilité et sa date échue.

  - **Les contraintes disjonctives** qui imposent la non-réalisation simultanée de deux opérations.

  - **Les contraintes cumulatives** qui sont dues à des ressources non renouvelables telles que l'énergie, les matières premières, etc. Ces

contraintes limitent également le nombre des opérations qu'il est possible d'exécuter simultanément sur une ressource.

- **Les opérations avec ou sans préemption.**

- Le **critère** décrit la ou les mesures de performance à optimiser. Les critères suivants sont importants du point de vue des systèmes de production :

  - la minimisation de la durée totale de l'ordonnancement (*makespan*) ou:

$$\text{Min } \max_j C_j$$

où C_j est la date de fin de la dernière opération du travail j.

  - la minimisation de la somme des temps de séjour des travaux dans le système ou :

$$\text{Min } \sum_j (C_j - r_j)$$

où r_j est la date d'arrivée du travail j dans le système.

  - la minimisation de la somme des retards ou :

$$\text{Min } \sum_j \max(0, C_j - d_j)$$

où d_j est la date échue du travail j dans le système.

  - la minimisation de la somme des avances ou :

$$\text{Min } \sum_j \max(0, d_j - C_j)$$

  - la minimisation du nombre des travaux en retard ou :

$$\text{Min } \sum_j U_j$$

où $U_j = 1$ si $C_j > d_j$ et $U_j = 0$ sinon.

I.2 Représentation des problèmes d'ordonnancement

Représenter un problème d'ordonnancement, c'est utiliser une méthode, par exemple graphique, pour spécifier de manière rigoureuse et sans ambiguïté les données et les contraintes du problème.

Pour un problème d'ordonnancement, on distingue la représentation du problème et la représentation de sa solution. Les outils de représentation de problème d'ordonnancement les plus utilisés sont les graphes potentiels-tâches et les réseaux de Petri. Quant à la représentation de la solution, l'outil le plus usité et le plus populaire est incontestablement le diagramme de Gantt. Dans ce paragraphe, nous présentons le graphe potentiel-tâches et le diagramme de Gantt. Les réseaux de Petri seront présentés ultérieurement.

Pour illustrer les méthodes et les notions, nous considérons un job-shop composé de trois machines M_1, M_2 et M_3. Il y a trois travaux P_1, P_2 et P_3 à réaliser et chaque travail nécessite deux opérations O_{jk}, $\forall\ j \in \{1,2,3\}$, $\forall\ k \in \{1,2\}$. Chaque opération O_{jk} peut être exécutée par une machine M_{jk} avec un temps opératoire t_{jk}. Ces données sont représentées dans la table I.2.

Travail P_j	Opération: M_{jk} , t_{jk}	
P_1	M_1 , 5	M_2 , 5
P_2	M_2 , 10	M_3 , 5
P_3	M_1 , 5	M_3 , 10

Table I.2 Exemple d'un job-shop

I.2.1 Représentation du problème par les graphes potentiels-tâches

Un graphe potentiels-tâches est composé de n+2 nœuds reliés par un ensemble d'arcs où n désigne le nombre d'opérations. Chaque opération à ordonnancer est représentée par un nœud. De plus, nous introduisons deux opérations fictives correspondant, respectivement, au début et à la fin de l'ordonnancement et représentées, respectivement, par le nœud "0" et le nœud "*".

Un graphe potentiels-tâches comporte deux types d'arcs : les arcs monodirectionnels et les arcs bidirectionnels, aussi appelés arcs conjonctifs et arcs disjonctifs. Les arcs monodirectionnels sont valués et représentent les contraintes de succession des opérations. Deux nœuds i et j sont reliés par un arc monodirectionnel (i,j) avec pour poids le temps opératoire de l'opération i si l'opération j ne peut commencer qu'à la fin de l'opération i, c'est-

à-dire que l'opération j suit l'opération i. Il y a également un arc monodirectionnel (0,i) de poids nul reliant le nœud "0" à chaque nœud "i" et un arc monodirectionnel (i,*) de poids nul reliant tous les nœuds "i" au nœud "*". Pour un problème d'ordonnancement, les contraintes de précédence sont donc représentées par ces arcs monodirectionnels.

Les arcs bidirectionnels ne sont pas valués et sont utilisés pour représenter les opérations utilisant la même ressource. Ainsi, deux nœuds i et j sont reliés par un arc bidirectionnel si les opérations correspondantes nécessitent la même ressource ou machine. Cependant, si le graphe potentiels-tâches est utilisé dans une méthode de résolution qui construit de manière progressive l'ordonnancement, ces arcs bidirectionnels se transforment progressivement en arcs monodirectionnels en fonction d'un choix déjà fait. De ce fait, l'arc bidirectionnel (i,j) devient l'arc monodirectionnel (i,j) si l'opération i précède l'opération j ; il devient l'arc monodirectionnel (j,i) si l'opération i suit l'opération j et il reste inchangé si aucun choix n'est fait.

La figure I.5 donne le graphe potentiel-tâches de l'exemple illustratif.

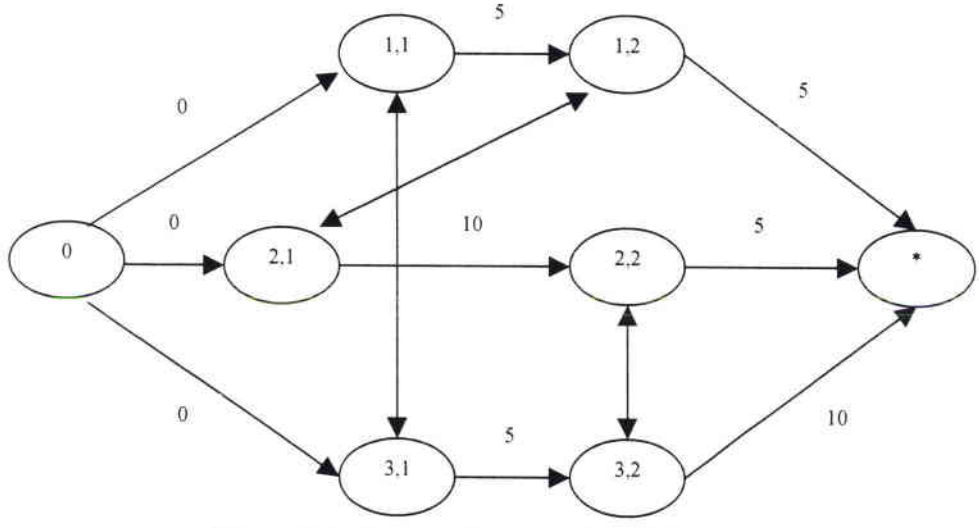

Figure I.5 : Un graphe potentiel-tâches

Cependant, nous ne pouvons pas utiliser les graphes potentiels-tâches pour représenter, d'une manière générale, les relations de conflit de ressources, surtout si les ressources sont de natures différentes (cas de moyens de transport alloués à un produit au long de sa réalisation) ; c'est pourquoi d'autres formes de représentation comme les réseaux de Petri sont de plus en plus utilisées.

I.2.2 Représentation d'une solution du problème d'ordonnancement par diagramme de Gantt

Les diagrammes de Gantt représentent les ordonnancements du point de vue des ressources. Lorsqu'on représente un ordonnancement par un diagramme de Gantt, on représente, en parallèle, l'état d'occupation de l'ensemble de ressources dans le temps. Cette représentation permet d'apprécier la qualité de l'ordonnancement et, surtout, elle permet un suivi simple et efficace de l'exécution de l'ordonnancement.

Pour illustrer cette méthode de représentation, nous considérons l'ordonnancement du job-shop de trois machines et de trois travaux donnés dans la table I.3 ; le diagramme de Gantt correspondant est donné dans la figure I.6.

Opération	$t_{début}$	t_{fin}
$O_{1,1}$	0	5
$O_{1,2}$	5	10
$O_{2,1}$	10	20
$O_{2,2}$	20	25
$O_{3,1}$	5	10
$O_{3,2}$	10	20

Table I.3 Ordonnancement de l'exemple

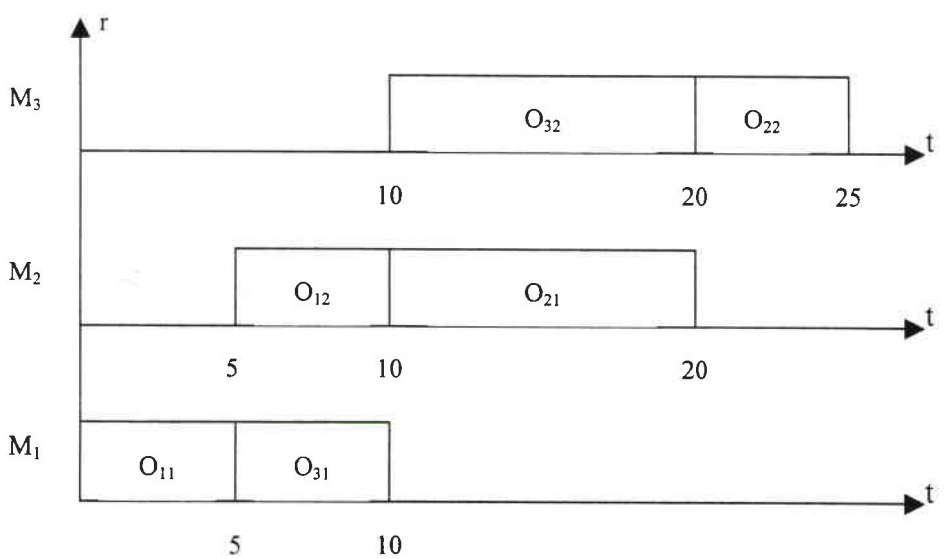

Figure I.6 : Diagramme de Gantt de l'ordonnancement de la table I.3

I.3 Complexité des problèmes d'ordonnancement

Une notion importante, dans la théorie d'ordonnancement, est la notion de complexité qui mesure la difficulté d'un problème. La complexité d'un problème est liée à la complexité ou l'efficacité de l'algorithme qui le résout. D'une manière générale, on définit la complexité d'un problème comme la complexité de l'algorithme le plus efficace qui le résout ; ce dernier n'est pas nécessairement connu et peut être très différent de l'algorithme réellement implanté pour la résolution du problème.

De manière informelle, la complexité d'un algorithme mesure le temps nécessaire pour trouver une solution. On exprime la complexité en fonction de la longueur des données d'entrée. Cette longueur correspond à la taille du problème et représente habituellement le nombre de travaux, le nombre d'opérations et le nombre de ressources, dans le cadre d'un problème d'ordonnancement.

Il est bien évident que la fonction de complexité d'un algorithme dépend de la rapidité de l'ordinateur utilisé. Néanmoins, il est fondamental de différencier les fonctions de complexité polynomiale et les fonctions de complexité exponentielle ou, plus généralement, les fonctions de complexité non polynomiale. On parle alors des algorithmes polynomiaux et des algorithmes non polynomiaux (NP).

Plus formellement, une fonction f(n) est dite d'ordre $O(g(n))$ s'il existe une constante c telle que $|f(n)| \leq c.|g(n)|$ pour tout $n \geq 0$. Un algorithme polynomial a une fonction de complexité $O(p(n))$ où p(n) est un polynôme de degré n et n est la taille du problème. Un algorithme dont la fonction de complexité ne peut être limitée par un polynôme p(n) est dit algorithme non polynomial ou exponentiel. Cette définition inclut des fonctions de complexité telles que $n^{\log n}$, qui normalement, n'est pas considérée comme une fonction exponentielle.

Pour illustrer les différents niveaux de complexité, nous présentons dans la table I.4 quelques fonctions de complexité et leurs temps d'exécution en secondes. Cette table est due à Garey et Johnson (79).

Dans la théorie de la complexité, on parle également des algorithmes pseudo-polynomiaux. Un algorithme est dit pseudo-polynomial si sa fonction de complexité est d'ordre $O(p(n,L))$ où le degré de polynôme varie selon la valeur de certaines données L du problème. Un exemple classique est le problème de Knapsac qui consiste à remplir un contenant avec des objets de taille différente. Ce problème n'admet pas d'algorithme polynomial pour sa résolution mais on peut trouver des algorithmes polynomiaux de complexité $O(p(n))$ où les paramètres du polynôme p(n) sont fonction de la taille du contenant.

Fonction Complexité	Taille du problème					
	10	20	30	40	50	60
n	.00001 secondes	.00002 secondes	.00003 secondes	.00004 secondes	.00005 secondes	.00006 secondes
n^2	.0001 secondes	.0004 secondes	.0009 secondes	.0016 secondes	.0025 secondes	.0036 secondes
n^3	.001 secondes	.008 secondes	.027 secondes	.064 secondes	.125 secondes	.216 secondes
n^5	.1 seconde	3.2 secondes	24.3 secondes	1.7 secondes	5.2 secondes	13.0 secondes
2^n	.001 secondes	1.0 seconde	17.9 secondes	12.7 secondes	35.7 secondes	366 secondes
3^n	.059 secondes	58 minutes	6.5 années	3855 siècles	$2x10^8$ siècles	$1.3x10^{13}$ siècles

Table I.4 - Les temps d'exécution selon le niveau de complexité

Dans l'étude des problèmes d'ordonnancement, un problème est dit polynomial s'il existe un algorithme polynomial pour le résoudre. Sinon, on parle de problèmes NP-difficiles. Un problème est dit NP-difficile au sens fort s'il n'existe pas d'algorithme pseudo-polynomial.

De nombreuses études de complexité (Rinnooy Kan, 76 ; Lenstra et al., 77) ont montré que la plupart des problèmes d'ordonnancement sont NP-difficiles au sens fort. Les seuls problèmes polynomiaux concernent essentiellement des cas particuliers de problèmes à une ou deux machines (Baker, 74 ; Carlier et Chrétienne, 88 ; GOThA, 93) et, par conséquent, n'ont qu'un impact très limité. La NP-complexité des problèmes d'ordonnancement impliquent que le temps nécessaire à leur résolution croît de manière exponentielle avec le nombre de travaux et le nombre de ressources. Il est alors difficile, voire impossible, de trouver des algorithmes exacts pour résoudre des problèmes d'ordonnancement de taille réaliste. C'est pour cette raison que de nombreuses méthodes heuristiques ont été développées pour trouver de bonnes solutions en un temps acceptable.

I.4 Méthodes de résolution des problèmes d'ordonnancement

La NP-complexité des problèmes d'ordonnancement généraux façonne l'étude des problèmes d'ordonnancement. De ce fait, on trouve dans la littérature des méthodes exactes et polynomiales pour des problèmes particuliers et souvent simples, des méthodes générales et exactes pour des problèmes de taille faible, des méthodes heuristiques particulières, taillées

sur mesure, pour certains types de problèmes, et des méthodes heuristiques générales appelées des méta-heuristiques.

Les premières méthodes exactes et polynomiales ont été développées pour résoudre des problèmes d'ordonnancement de projets sans contraintes de ressource. Ces méthodes, dont les plus connues sont la méthode potentiels-tâches et la méthode CPM-PERT, consistent à trouver les chemins critiques dans un graphe valué. Les contraintes de ressources sont ensuite prises en compte à l'aide de la méthode sérielle mais il s'agit alors d'une méthode heuristique. Des méthodes exactes et polynomiales ont également été développées pour les problèmes de flot dans un réseau de transport avec contrainte de capacité. La méthode la plus connue pour les problèmes de flot, est la méthode de Ford-Fulkerson. Nous notons que la méthode de chemin critique et les méthodes pour des problèmes de flot sont largement utilisées dans le développement des méthodes (exactes ou approchées) d'ordonnancement.

La plupart des problèmes d'ordonnancement deviennent NP-difficiles dès qu'il y a plus de deux machines. Pour cette raison, la plupart des méthodes exactes et polynomiales concerne des problèmes d'ordonnancement à une ou deux machines. Comme Carlier et Chrétienne (88) l'ont montré, l'ordonnancement optimal de ces problèmes peut être obtenu à l'aide de la méthode d'échange ou la méthode de liste. La méthode d'échange consiste à améliorer un ordonnancement donné par permutations successives des opérations utilisant la même ressource. La méthode de liste part d'une liste de priorités de travaux à ordonnancer et construit un ordonnancement en plaçant de manière prioritaire la première tâche prête de la liste. On montre à l'aide de ces méthodes que la règle SPT (*Shortest Pocessing Time*) minimise la somme d'en-cours, la règle EDD (*Earliest Due Date*) minimise le retard maximal (L_{max}), le makespan est minimisé, pour le cas d'un flow-shop à deux machines, par l'algorithme de Johnson.

Pour les problèmes d'ordonnancement NP-difficiles, les méthodes les plus utilisées pour obtenir les solutions optimales sont les procédures par séparation et évaluation (ou PSE), la programmation linéaire et la programmation dynamique. Ces méthodes explorent, de manière explicite ou implicite, l'espace des solutions et utilisent des indicateurs de performances pour limiter le nombre des solutions à examiner. Ainsi la programmation linéaire en variables continues et la relaxation Lagrangienne sont souvent utilisées pour fournir rapidement des indicateurs de performance par le biais de relaxation de certaines contraintes telles que la non-préemption des tâches, les contraintes de capacité de ressources.

Compte tenu de la NP-complexité des problèmes d'ordonnancement et de la variété des problèmes qui se posent, de nombreuses méthodes heuristiques ont été développées pour obtenir des solutions proches de l'optimum dans un temps raisonnable. Un grand nombre de méthodes heuristiques sont taillées sur mesure pour des problèmes particuliers et construisent l'ordonnancement selon certaines conditions d'optimalité. Beaucoup d'entre elles utilisent la méthode d'échange ou la méthode de liste mentionnée plus tôt. Parmi les méthodes

heuristiques, nous citons la méthode CDS (Campbell et al., 70) pour l'ordonnancement des flow-shops et la méthode dite de la machine goulot (*shift-bottleneck*) pour la minimisation du makespan des job-shops.

Ces dix dernières années ont vu le développement rapide de méthodes heuristiques génériques pour la résolution des problèmes NP-difficiles. Ces méthodes, appelées méta-heuristiques, utilisent deux notions importantes : le voisinage et la population. La plupart des méta-heuristiques acceptent des solutions moins bonnes que la solution actuelle. La notion de population a été introduite pour capter la structure des solutions optimales. Les méta-heuristiques les plus usitées dans l'ordonnancement sont le recuit simulé, les algorithmes génétiques et la méthode tabou. En plus des méta-heuristiques, la relaxation Lagrangienne peut également être utilisée comme une heuristique générique. Nous allons détailler cette utilisation de la relaxation Lagrangienne ultérieurement.

I.5 Techniques classiques d'ordonnancement

Nous présentons, dans ce paragraphe, les techniques usuelles d'ordonnancement des systèmes de production. Elles comprennent les techniques exactes, les heuristiques dédiées et les méta-heuristiques.

I.5.1 Procédures par Séparation et Evaluation

La Procédure par Séparation et Evaluation (PSE), en anglais Branch and Bound, est la méthode exacte la plus utilisée pour résoudre les problèmes d'ordonnancement des systèmes de production. Dans ce paragraphe, sans perte de généralité, nous considérons le problème de la minimisation d'un critère donné.

Une procédure par séparation et évaluation peut être expliquée par les deux notions suivantes :

- La "séparation" consiste à décomposer successivement un problème en sous-problèmes concernant des sous-ensembles de solutions exclusives et exhaustives par rapport au problème initial. Les solutions de chaque sous-problème comportent une solution partielle connue et commune, et des variables à définir.

- L'évaluation d'un sous-problème permet d'obtenir une Borne Inférieure (BI) de la solution optimale du sous-problème. La méthode d'évaluation la plus utilisée consiste à relaxer certaines contraintes du problème (par exemple la non-préemption des opérations, les contraintes de précédence et les contraintes de

capacité) et à résoudre le problème relaxé à l'aide de techniques telles que la programmation linéaire et la relaxation Lagrangienne. Il est bien évident que l'efficacité d'une PSE dépend fortement de la qualité de ses BI et de leur difficulté de calcul.

Une autre notion utilisée dans les PSE est la Borne Supérieure (BS) de la solution optimale du problème initial. Cette borne est améliorée au cours d'une PSE. Il est bien évident qu'un sous-problème ne contient pas de solution optimale si BI > BS.

Dans une PSE, nous représentons les sous-problèmes par une arborescence dont la racine correspond au problème initial et chaque sommet représente un sous-problème. Les sommets-fils d'un sommet A représente l'ensemble des sous-problèmes du sous-problème A.

Une PSE peut être résumée comme suit :

Etape 1 (Initialisation)
On part de la racine, on calcule sa BI et on calcule une BS initiale. Cette BS initiale est obtenue habituellement à l'aide d'une heuristique et est ensuite améliorée au cours du calcul.

Etape 2 (Séparation)
On choisit un sommet restant S et on crée les sommets fils de S représentant les sous-problèmes correspondants.

Etape 3 (Evaluation)
Pour chaque nouveau sommet, on calcule une BI. Si le nouveau sommet représente une solution complète, on calcule sa valeur de critère et on met à jour la BS.

Etape 4 (Elimination)
On élimine tout sommet tel que BI > BS.

Etape 5 (Critère d'Arrêt)
S'il n'y a plus de sommet restant, on arrête ; sinon, on va à l'étape 2.

Au vu de cet algorithme, il est naturel de s'interroger sur la stratégie de recherche pour le choix du sommet à séparer. Les stratégies les plus connues sont:

- la largeur d'abord qui consiste à explorer d'abord les sommets par niveau d'arborescence,
- la profondeur d'abord qui consiste à explorer d'abord les sommets avec le plus grand niveau de l'arborescence,
- la méthode progressive qui consiste à explorer d'abord le sommet de la plus petite BI, c'est-à-dire le plus prometteur.

Une des premières PSE a été proposée par Dantzig, Fulkerson et Johnson, (Dantzig et al., 54) pour résoudre le problème du Voyageur de Commerce. Depuis, des centaines, voire des milliers de PSE ont été proposées pour des problèmes d'ordonnancement. L'exemple le plus représentatif est la PSE proposée par Carlier et Pinson (Carlier et Pinson, 89) pour minimiser le makespan des job-shops. Cette PSE utilise des BI très serrées et une stratégie de recherche efficace faisant intervenir les propriétés des solutions optimales. Elle a permis la résolution d'un problème de 10 machines et 10 travaux ; ce problème est resté ouvert pendant plus de 20 ans. De cette étude, on constate que l'efficacité d'une PSE dépend essentiellement de la qualité des bornes inférieures et supérieures.

I.5.2 La Programmation Linéaire

Un programme linéaire modélise un problème dont les contraintes sont des inéquations linéaires et la fonction économique, une forme linéaire. Plus précisément, un programme linéaire peut s'écrire sous la forme :

$$\text{Min } C^T x \text{ sous la contrainte } Ax \leq B$$

où C et B sont des vecteurs de colonne et A une matrice. On parle d'un programme linéaire continu si les variables x_i sont des nombres réels, d'un programme linéaire en nombres entiers si les variables x_i sont des entiers et d'un programme linéaire en nombres mixtes si les variables peuvent être des entiers ou des réels.

Les programmes linéaires continus sont des problèmes polynomiaux et sont très bien résolus. La méthode la plus connue est la méthode du Simplexe (Dantzig, 63) qui possède une excellente complexité expérimentale : le nombre d'itérations est de l'ordre du nombre de contraintes.

Par contre, les programmes linéaires en nombres entiers ou mixtes sont NP-difficiles au sens fort. Leur résolution est une direction de recherche majeure de la recherche opérationnelle. Les méthodes les plus connues sont les méthodes de coupe, celle de Gomory (58), par exemple, et les PSE.

Nous remarquons qu'il n'y a pas beaucoup de problèmes d'ordonnancement qui peuvent être modélisés comme des programmes linéaires continus. Par contre, la plupart des problèmes d'ordonnancement peuvent être modélisés par un programme linéaire en nombres entiers ou mixtes. Ainsi, la programmation linéaire peut être utilisée pour la résolution de problèmes d'ordonnancement relaxés (contraintes de précédence, de non-préemption des opérations et de capacité des ressources) et, par conséquent, elle permet d'obtenir les bornes inférieures des problèmes d'ordonnancement.

Nous terminons ce paragraphe par la modélisation du problème de la minimisation de la somme de retard de n tâches sur une machine avec contraintes de disponibilité de tâches. Le programme linéaire en nombres mixtes suivant modélise ce problème :

$$\text{Min} \sum_{j=1}^{n} C_j$$

sous les contraintes suivantes :

$$S_j \geq r_j \tag{1.1}$$

$$C_j \geq S_j + p_j - d_j \, , \; C_j \geq 0 \tag{1.2}$$

$$S_{j'} + p_{j'} \leq Y_{jj'} M + S_j \tag{1.3}$$

$$S_j + p_j \leq (1 - Y_{jj'}) M + S_{j'} \, , \; Y_{jj'} \in \{0,1\}, \forall j, j' \in \{1,2,...,n\} \tag{1.4}$$

où (1.1) est la contrainte de disponibilité de tâches, (1.2) est le coût de retard, (1.3) et (1.4) sont les contraintes de ressource. Ainsi, S_j est la date de début de la tâche j, r_j sa date de disponibilité, p_j son temps opératoire, d_j sa date échue et C_j son coût de retard. $Y_{jj'}$ indique si la tâche j précède la tâche j' et M est un grand nombre avec $M \geq \Sigma_j (p_j + r_j)$ par exemple. Ainsi, nous avons $S_{j'} + p_{j'} \leq S_j$ si j' précède j et $S_j + p_j \leq S_{j'}$ sinon. Si nous relaxons l'intégrité des variables $Y_{jj'}$ et la remplaçons par la contrainte $0 \leq Y_{jj'} \leq 1$, le problème devient un programme linéaire en variables continues et nous obtenons ainsi une borne inférieure.

I.5.3 Programmation Dynamique

La Programmation Dynamique est un principe général applicable à de nombreux problèmes d'optimisation avec contraintes, linéaires ou non linéaires, en variables continues ou discrètes, mais possédant une certaine propriété dite de décomposabilité (Minoux, 83).

Le terme "programmation dynamique" provient du fait que cette méthode a d'abord été appliquée à l'optimisation des systèmes dynamiques, c'est-à-dire des systèmes qui évoluent au cours du temps et dont l'évolution peut être contrôlée par des variables de décision. Mais le principe de la Programmation Dynamique peut être appliqué à une gamme plus grande de problèmes, y compris les problèmes où le temps n'intervient pas, comme les problèmes d'optimisation en nombres entiers.

Le principe de Bellman qui définit la programmation dynamique est :

"Une politique est optimale si, à une période donnée, quelles que soient les décisions précédentes, les décisions qui restent à prendre constituent une politique optimale en tenant compte des résultats des décisions précédentes."

Ce principe est le point central de la théorie de la programmation dynamique et il est appelé principe de l'optimalité.

Lorsque nous appliquons la programmation dynamique à l'ordonnancement, l'idée est de ramener la résolution d'un problème d'ordonnancement qui comporte plusieurs variables à la résolution d'une suite de problèmes d'optimisation plus simples, par exemple à des problèmes avec une seule variable.

Pour mettre en œuvre cette idée, la plupart des approches de programmation dynamique suivent les deux étapes suivantes :

1ère - On place le problème dans une famille de problèmes de même nature.

2ème - On cherche une relation de récurrence reliant entre elles les valeurs optimales de ces différents problèmes.

Cette méthode est généralement coûteuse vu son caractère énumératif mais elle peut être très efficace pour des problèmes particuliers. Par exemple, la programmation dynamique permet d'obtenir des algorithmes polynomiaux par le problème du plus court chemin dans un graphe valué. Le problème initial consiste à trouver le plus court chemin de a à b. La famille de problèmes à considérer est la recherche du plus court chemin d'un sommet quelconque x à b. La relation récurrente reliant ces problèmes est :

$$PCC(x,b) = \min_{y \in \Gamma(x)} \{d(x,y) + PCC(y,b)\}$$

où PCC(x,y) est le plus court chemin de x à y, $\Gamma(x)$ est l'ensemble des successeurs de x et d(x,y) est le poids de l'arc (x,y).

En ordonnancement, la programmation dynamique a souvent été utilisée, pour des problèmes à une machine ou à des machines identiques, pour construire des algorithmes polynomiaux ou pseudo-polynomiaux (Lawler, 77 ; Schrage et Baker, 78 ; Potts et Van Wassenhove, 82 ; Carlier et Chrétienne, 88).

I.5.4 Méthodes heuristiques

Comme nous l'avons dit, la plupart des problèmes d'ordonnancement sont NP-difficiles au sens fort et il est difficile, voire impossible, de trouver des méthodes exactes et

efficaces. Pour cela, on trouve un grand nombre de méthodes heuristiques dans la littérature. Nous présentons, dans ce paragraphe, un échantillon représentatif des méthodes heuristiques dédiées.

I.5.4.1 Méthode par Machine Goulot

Cette méthode, aussi appelée *Shifting Bottleneck*, a été proposée par Adams, Balas et Zawak (Adams et al., 88) pour la minimisation du makespan des job-shops. Elle est considérée comme l'une des méthodes les plus efficaces et sert de méthode de référence pour l'ordonnancement des job-shops.

Cette méthode calcule l'ordonnancement par construction progressive. Elle résout successivement des problèmes d'ordonnancement à une machine. Chaque problème à une machine détermine l'ordre de passage des opérations qui nécessitent la machine considérée, tout en respectant l'ordonnancement des machines déjà ordonnancées. Ce problème peut être prouvé équivalent à la minimisation du makespan de n tâches, sur une machine avec dates de disponibilité et de latence des tâches. Le temps de latence d'une opération correspond au temps séparant la fin de l'opération et la fin du travail correspondant. Ce problème peut être résolu, de manière efficace, à l'aide d'un algorithme mis au point par Carlier (Carlier, 82). Nous donnons, ci-après, quelques détails de la méthode par machine goulot.

La notation utilisée pour décrire cette méthode est la suivante :

M : l'ensemble des machines,
M_0: l'ensemble de machines sur lesquelles la séquence de tâches a été calculée.

A chaque étape de la méthode, on choisit la machine la plus critique, c'est-à-dire la machine goulot, en résolvant un problème à une machine pour toutes machines encore non séquencées (c'est-à-dire dans l'ensemble $M \backslash M_0$), et on ajoute celle-ci à M_0. Ensuite, l'ordonnancement de chaque machine dans M_0 est réoptimisé en utilisant une nouvelle fois l'algorithme de Carlier. Enfin, on répète ceci sur les machines qui sont sur le chemin critique.

L'algorithme, proposé par Adams et al., est :

Etape 0 : $M_0 = \emptyset$

Etape 1 : Identifier la machine goulot k_g parmi les machines $k \in M \backslash M_0$, ainsi que la séquence optimale (sélection S_{kg}) correspondante,
$M_0 \leftarrow M \cup \{ k_g \}$

Etape 2 : Réoptimiser successivement la séquence de chaque machine $k \in M_0$. Si $M = M_0$, stop ; sinon aller à l'étape 1.

I.5.4.2 Méthode par Permutation

La méthode par permutation d'opérations est une méthode qui améliore le makespan d'une solution obtenue pour un problème d'ordonnancement.

Cette méthode consiste à choisir deux opérations successives sur une machine et essayer de les permuter pour diminuer la durée totale de l'ordonnancement. On remarque que ces opérations doivent être sur le chemin critique du graphe potentiel-tâches correspondant sans quoi la permutation n'améliorerait pas la solution. Cette méthode est également très utilisée dans les méta-heuristiques Tabou et Recuit Simulé.

La permutation des opérations peut être faite selon l'algorithme suivant :

Etape 1 : Choix de deux opérations.

Choix de deux opérations successives sur le chemin critique utilisant la même machine.

Etape 2 : Permutation des opérations.

2.1 Permutation de deux opérations sélectionnées.
2.2 Calcul de la durée totale de l'ordonnancement, C_{max}, après la permutation.
2.3 Si le nouveau C_{max} est inférieur à l'ancien, valider la permutation, sinon, l'annuler.

Etape 3 : Condition d'arrêt.

3.1 Si aucune permutation de deux opérations successives du chemin critique n'améliore pas le C_{max}, stop.
3.2 Sinon, retourner à l'étape 1.

Nous remarquons que cette méthode permet de faire converger la solution d'ordonnancement vers un minimum local.

I.5.5 Méta-Heuristiques

Les méthodes appelées méta-heuristiques sont des méthodes heuristiques générales. Nous avons déjà mentionné que les méta-heuristiques utilisent deux notions importantes : le voisinage et la population. Dans ce paragraphe, nous présentons les méta-heuristiques

dans cet ensemble des méta-heuristiques, la relaxation Lagrangienne que nous considérons comme une méthode heuristique de caractère générique.

I.5.5.1 Méthodes de la Relaxation Lagrangienne

La méthode de la Relaxation Lagrangienne est habituellement utilisée pour déterminer des bornes inférieures de qualité (pour les cas de minimisation d'un critère) pour les problèmes combinatoires. Elle consiste à relaxer certaines contraintes et à les injecter dans le critère par le biais de multiplicateurs Lagrangiens. Plus précisément, pour le problème :

$$J = \min_{x \in X} f(x)$$

sous la contrainte :

$$g(x) \leq 0 ,$$

le problème relaxé s'écrit :

$$L(\lambda) = \min_{x \in X} f(x) + \lambda^T g(x) , \text{ pour } \lambda \geq 0$$

Le problème suivant

$$L(\lambda^*) = \max_{\lambda \geq 0} L(\lambda)$$

est le problème dual. Il est facile de montrer que $L(\lambda^*) \leq J$, ce qui prouve que la relaxation Lagrangienne conduit à des bornes inférieures. La différence $J - L(\lambda^*)$ est souvent appelée le saut de dualité. Il été démontré que les bornes de la relaxation Lagrangienne sont meilleures que les bornes obtenues par simple relaxation d'intégrité des variables entiers.

Partant du constat que les bornes de la relaxation Lagrangienne sont généralement serrées pour des problèmes d'ordonnancement réalistes et que les solutions correspondantes sont proches de l'optimum, Luh et Hoitomt (93) ont proposé une autre utilisation de la relaxation Lagrangienne pour l'ordonnancement d'une large gamme de systèmes comprenant les job-shops, les systèmes d'assemblage, les systèmes avec temps de "setup" et capacités de stockage (Luh et Hoitomt, 93 ; Czerwinski et Luh, 94 ; Chang et Liao, 94 ; Chen et al., 94 ; Tomastik et al., 96). La technique proposée part d'une modélisation originale, relaxe certaines contraintes du problème pour obtenir des problèmes relaxés faciles à résoudre, résout le problème dual $L(\lambda^*)$ et construit un ordonnancement admissible à partir de la solution du problème dual. La borne inférieure $L(\lambda^*)$ permet de mesurer la qualité de l'ordonnancement obtenu.

Dans ce qui suit, nous présentons la méthode proposée par Luh et al. pour l'ordonnancement de n travaux dans un job-shop avec M machines. Le problème consiste à minimiser un critère du type $\sum_{j=1}^{n} f_j(C_j)$ où C_j est la date de fin du travail j. Ce critère inclut la somme des retards comme un cas particulier dans lequel $f_j(C_j) = \max \{0, C_j - d_j\}$.

Ce problème peut être formulé comme suit :

$$J = \text{Min} \sum_{j=1}^{n} f_j(C_j)$$

sous les contraintes suivantes :

$$C_{jk} - b_{jk} + 1 = p_{jk}\ ,\ \forall j,\ \forall\ 1 \le k \le N_j \tag{1.5}$$

$$C_{jk} + 1 \le b_{jk+1},\ \forall j,\ \forall\ 1 \le k \le N_j \tag{1.6}$$

$$\sum_{(j,k)\in O(m)} \mathbf{1}\{b_{jk} \le t \le C_{jk}\} \le 1,\ \forall\ 1 \le m \le M,\ \forall\ 1 \le t \le H \tag{1.7}$$

où (1.5) est la contrainte de temps opératoire, (1.6) est la contrainte de précédence, (1.7) est la contrainte de ressource, b_{jk} est la date de début de l'opération (j,k), C_{jk} sa date de fin, p_{jk} son temps opératoire, N_j le nombre d'opérations du travail j où $C_j = C_{jN_j}$ et O(m) est l'ensemble d'opérations utilisant la machine m. Dans ce modèle, nous avons discrétisé le temps en périodes élémentaires et H est le nombre de périodes.

Pour résoudre ce problème, nous relaxons les contraintes de ressource (I.7) et obtenons les problèmes relaxés suivants :

$$L(\lambda) = \text{Min} \sum_{j=1}^{n} f_j(C_j) - \sum_{m,t} \lambda_{mt} \left(1 - \sum_{(j,k)\in O(m)} \mathbf{1}\{b_{jk} \le t \le C_{jk}\}\right)$$

sous les contraintes suivantes :

$$C_{jk} - b_{jk} + 1 = p_{jk}\ ,\ \forall j,\ \forall\ 1 \le k \le N_j$$

$$C_{jk} + 1 \le b_{jk+1},\ \forall j,\ \forall\ 1 \le k \le N_j$$

Ce problème se décompose en sous-problème du niveau travail :

$$L(\lambda) = - \sum_{m,t} \lambda_{mt} + \sum_{j=1}^{n} L_j(\lambda)$$

avec

$$L_j(\lambda) = \text{Min } f_j(C_j) + \sum_{k=1}^{Nj} \sum_{t=bjk}^{Cjk} \lambda_{Mjkt}$$

où M_{jk} désigne la machine pouvant effectuer l'opération (j,k).

Le problème dual est alors :

$$L(\lambda^*) = \max_{\lambda \geq 0} L(\lambda)$$

On montre aisément que les problèmes relaxés du niveau travail $L_j(\lambda)$ peuvent être résolus par une approche de Programmation Dynamique. Le problème dual peut être résolu par une méthode de gradient utilisant l'itération suivante :

$$\lambda^{n+1} = \lambda^n + \varepsilon(n) \nabla_\lambda L$$

où $\varepsilon(n)$ est le pas et $\nabla_{\lambda mt} L = \sum_{(j,k) \in O(m)} \mathbf{1}\{b_{jk} \leq t \leq C_{jk}\} - 1$.

A l'issue de la résolution du problème dual, nous pouvons construire un ordonnancement admissible à l'aide d'une méthode de liste avec l'ordonnancement du problème dual S_{jk} comme degré de priorité.

Pour résumer, la méthode procède comme suit :

Etape 1 : Initialiser les multiplicateurs Lagrangiens λ_{mt} ;

Etape 2 : Résoudre les problèmes relaxés du niveau de travail $L_j(\lambda)$;

Etape 3 : Actualiser les multiplicateurs : $\lambda^{n+1} = \lambda^n + \varepsilon(n) \nabla_\lambda L$;

Etape 4 : Si la convergence de λ^n n'est pas obtenue, aller à l'étape 2 ;

Etape 5 : Construire un ordonnancement admissible $S^{\square}_{jk}$ à l'aide d'une méthode de liste. Soit $J^{\square}$ sa valeur de critère ;

Etape 6 : Evaluer l'ordonnancement obtenu :

$$\frac{J^{\square} - L(\lambda^*)}{J^{\square}} \leq \frac{J^{\square} - J^*}{J^{\square}}$$

où J* est le critère de la solution optimale.

I.5.5.2 Méthodes du recuit simulé

La méthode du recuit simulé et la méthode Tabou, présentées ci-après, peuvent être considérées comme des améliorations de l'optimisation locale. Une méthode d'optimisation locale part d'une solution initiale x_0 et l'améliore en déterminant la solution optimale x' dans un voisinage de x_0. Ce processus est répété en remplaçant x_0 par x' jusqu'à convergence. L'avantage de l'optimisation locale est la simplicité mais elle risque de conduire à un optimum local comme illustré par la figure I.7.

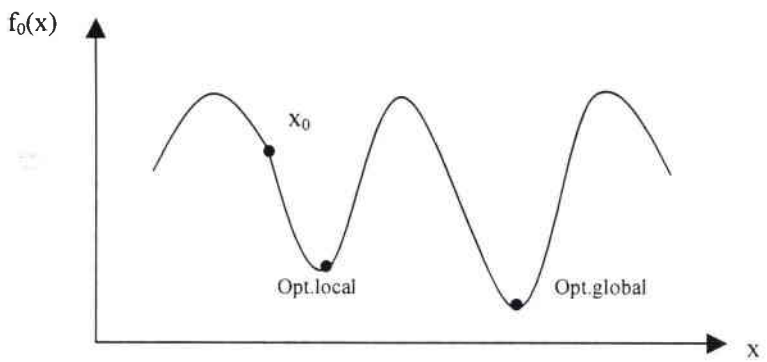

Figure I.7 : Optimum local

Pour éviter le piège de l'un optimum local, la méthode du recuit simulé accepte des solutions moins bonnes mais cette acceptation est contrôlée par un processus inspiré de la méthode du recuit dans la physique statistique (Kirkpatrick et al., 83). Il s'agit de solidifier un matériel en état liquide afin d'atteindre un état d'énergie minimale. Cet état ne peut être atteint qu'en partant d'une haute température et en baissant très lentement la température.

La méthode du recuit simulé utilise la notion de température pour contrôler la dégradation des solutions dans la recherche de l'optimum. L'algorithme ci-dessous est un des schémas les plus utilisés (pour le cas de minimisation):

Etape 1 : Choisir une solution initiale s_o et calculer sa valeur de critère $f(s_o)$;

Etape 2 : Choisir une température initiale T (T>0) ;

Etape 3 : Choisir au hasard une solution voisine s et calculer sa valeur de critère f (s) ;

Etape 4 : Si $f(s) \leq f(s_o)$, $s_o \leftarrow s$; sinon, $s_o \leftarrow s$ avec la probabilité $e^{-(f(s)-f(s0))/T}$;

Etape 5 : $T \leftarrow \alpha T$;

Etape 6 : Si la température finale ε n'est pas atteinte, c'est-à-dire T > ε, aller à l'étape 3 ; sinon, arrêt.

où α est le ratio de température qui définit le processus de contrôle de la température. Il n'existe pas de règle pour choisir les paramètres α, ε et la température initiale. On procède généralement par tâtonnement. Très souvent, $0,87 \leq \alpha \leq 0,99$.

La propriété la plus importante est l'optimalité asymptotique. Il a été prouvé dans Lundy et Mees (86), Aarts et Korst (89), Van Laarhoven et al. (92), que, sous des conditions sur la structure de voisinage, la méthode conduit avec probabilité 1 à un optimum global lorsque $\varepsilon \rightarrow 0$ pour α proche de 1. Cela revient à dire que l'optimum global peut être atteint si la température est baissée lentement, ce qui confirme l'analogie avec la physique statistique. Une condition suffisante sur la structure de voisinage est sa connectivité, c'est-à-dire qu'il existe un chemin de longueur finie allant d'une solution quelconque à un optimum global.

Dans ce paragraphe, nous présentons la méthode du recuit simulé développée par Van Laharhoven et al. (92) pour la minimisation du makespan d'un job-shop. Dans cet article, le graphe potentiels-tâches est utilisé pour représenter le problème. Chaque solution s (c'est-à-dire un ordonnancement) est représentée par l'ordre d'exécution des opérations utilisant la même machine. Une structure de voisinage simple et naturelle aurait été obtenue par permutation des deux opérations v et w quelconques et consécutives sur la même machine. Afin de réduire l'espace de voisinage et améliorer l'efficacité de la méthode, dans Van Laarhoven et al. (92), seules les opérations v et w sur un chemin critique et consécutives sur la même machine sont permutées. Ils ont montré que l'on peut transformer en un nombre fini d'étapes une solution quelconque en un optimum global en respectant la structure de voisinage. Cela garantit l'optimalité asymptotique du recuit simulé. Nous soulignons que le graphe potentiels-tâches est également utilisé pour le calcul de la valeur de critère.

I.5.5.3 Méthode Tabou

Pour ne pas être piégée dans un optimum local, la méthode Tabou utilise la notion de mémoire pour guider la recherche. Cette méthode a été initialement proposée par Glover (Glover, 86) sous le nom de la recherche avec tabous. L'idée de base est de modifier, de manière itérative et locale, une solution, tout en gardant en mémoire les modifications afin de ne pas visiter la même solution deux fois. Pour ce faire, les modifications ou leurs caractéristiques sont stockées dans une liste pour interdire leur utilisation pendant un certain nombre d'itérations, d'où le nom de la recherche avec tabous et celui de la liste de tabous.

Selon Glover (Glover et al., 92) la recherche Tabou est une généralisation des méthodes d'optimisation locale. Elle explore itérativement l'espace X des solutions d'un problème d'optimisation donné, en se déplaçant d'une solution courante s, $s \in X$, à une

nouvelle solution s' située dans le voisinage de s, s' $\in$ N (s), le choix de cette nouvelle solution s' étant déterminé par l'évaluation d'une certaine fonction objective f. Le voisinage N(s) d'une solution s est défini comme l'ensemble des solutions s' $\in$ X pouvant être obtenues suite à l'application d'une transformation locale à s. Par exemple, pour le problème du voyageur de commerce, cette transformation pourrait être une méthode d'exploration locale (ou amélioration itérative) comme la méthode d'échange 2-opt (Lin, 64) et pour l'ordonnancement d'un job-shop, la permutation de deux opérations consécutives sur une machine.

Toutefois, contrairement aux méthodes d'optimisation locale qui s'arrêtent dès qu'il n'y a plus de s' permettant d'améliorer f, s' ici est définie comme étant la meilleure solution parmi les éléments de N(s) sans référence aucune à f(s). Ceci permet à la recherche Tabou de poursuivre la recherche de solutions meilleures même si cela entraîne une dégradation de la fonction objective.

Cette modification du processus d'exploration rend la recherche Tabou insensible aux optima locaux mais elle introduit, du même coup, le risque de reboucler dès que l'on sort de l'un de ces optima et d'y retourner aussitôt. C'est là que la notion de mémoire trouve sa justification. Pour régler ce problème, la recherche Tabou conserve des informations sur le cheminement récent effectué à travers X afin d'interdire certaines transformations de la solution courante qui pourraient ramener la procédure vers des solutions déjà rencontrées. Ces transformations sont alors déclarées tabou, d'où le nom de la méthode. Ceci a pour effet de restreindre l'accès à certaines solutions dans N(s). C'est ce mécanisme d'interdictions ou de tabous qui permet d'orienter l'exploration de X au fur et à mesure qu'elle se déroule. Toutefois, pour conserver à l'approche suffisamment de flexibilité, le caractère tabou de ces transformations ne doit pas être maintenu en permanence. En limitant la durée de vie des tabous, on permet à la méthode de remettre en question ses choix passés (une fois les risques de rebouclage disparus) et ainsi de mener une exploration beaucoup plus large du domaine des solutions.

Plus précisément, une méthode Tabou se résume comme suit :

Etape 1 : Initialiser
- choisir une solution initiale $s_0 \in X$;
- initialiser la liste des tabous
- $k \leftarrow 0$, $s^* \leftarrow s_0$;

Etape 2 : Tant que le critère d'arrêt est non satisfait, faire :
(a) Déterminer s_{k+1}, une solution voisine de s_k, en tenant compte de la liste des tabous ;
(b) Si s_k est meilleur que s^*, alors $s^* \leftarrow s_{k+1}$;
(c) $k \leftarrow k+1$;
(d) mettre à jour les tabous.

Traditionnellement, le critère d'arrêt peut être un nombre donné d'itérations ou un nombre donné d'itérations consécutives sans amélioration. Différentes méthodes peuvent être utilisées dans l'étape 2(a) pour calculer s_{k+1}. La méthode la plus utilisée est l'optimisation locale des solutions non taboues.

La méthode Tabou a été largement appliquée à différents problèmes combinatoires tels que les problèmes de tournées de véhicules (Gendreau et al., 94) et l'ordonnancement des job-shops (Balas et Vazacopoulos, 95). On trouve de nombreuses références dans Soriano et Gendreau (97), Blazewickz et al. (96).

I.5.5.4 Méthodes Génétiques

L'idée de base des algorithmes génétiques vient du processus évolutionnaire des espèces qui se reproduisent sexuellement. Pendant la reproduction sexuelle, les parents créent un nouvel individu, leur descendant différent d'eux. Chaque individu est caractérisé par son empreinte génétique, c'est à dire un ensemble de chromosomes. Les chromosomes du descendant sont le résultat de l'action de deux mécanismes fondamentaux. Le premier est le croisement qui est une combinaison des empreintes génétiques des parents (ou ascendants) pour créer l'empreinte génétique d'un nouvel individu. Le deuxième est la mutation qui est une modification de l'empreinte génétique du descendant. Le descendant sera différent de ses parents ; toutefois, il aura un certain nombre de leurs caractéristiques. Si le descendant a les "bonnes" caractéristiques de ses parents, sa probabilité de survivre est très haute en comparaison aux individus qui ont de mauvaises caractéristiques de leurs parents. Nous pouvons expliquer ces bonnes ou mauvaises caractéristiques en fonction de l'environnement. L'idée est que deux individus bien adaptés au milieu vont donner naissance à des individus eux-mêmes bien adaptés au milieu et, peut-être, mieux adaptés à ce milieu que leurs parents, grâce en particulier aux mutations. A l'inverse, des individus mal adaptés à leur environnement risquent de donner naissance à des individus tellement mal adaptés à leur environnement qu'ils ne résisteront pas à la sélection naturelle.

L'analogie entre le processus évolutionniste et la méthode génétique proposée par Holland (75) se résume comme suit. Chaque individu correspond à une solution admissible du problème d'optimisation. La solution est codée comme un vecteur binaire ou empreinte génétique. L'opérateur de croisement échange les sous-vecteurs binaires des parents pour produire un descendant. La mutation est un opérateur permettant la modification d'un binaire de l'empreinte génétique d'un descendant. La qualité d'un individu (sa chance de survivre) dépend de la valeur de critère de la solution correspondante. Plus précisément, un algorithme génétique se résume comme suit :

1. Générer une population initiale de vecteurs (individus)

2. Tant que le critère d'arrêt est non satisfait, faire :
   (a) Choisir un ensemble de vecteurs-parents dans la population
   (b) Associer de manière aléatoire les parents et appliquer l'opérateur de croisement pour produire les vecteurs-enfants
   (c) Appliquer l'opérateur mutation aux vecteurs-enfants.

Les développements récents permettent d'utiliser des codages généraux et naturels, des opérateurs de croisement et de mutation plus puissants et faciles à utiliser. On trouve en particulier dans Blazewicz et al. (96), Caux et al. (93), de nombreux opérateurs de croisement et de mutation pour l'ordonnancement des job-shops.

I.6 Les Réseaux de Petri et les Systèmes de Production

Les réseaux de Petri (RdP) sont de plus en plus populaires pour la modélisation, la spécification et l'évaluation des performances des systèmes à événements discrets caractérisés par le parallélisme, la synchronisation, les ressources partagées, etc. L'intérêt croissant pour cet outil graphique se justifie par sa puissance de spécification, son utilisation aisée et, surtout l'arsenal mathématique pour l'analyse qualitative et quantitative. Par rapport aux autres outils d'évaluation des performances données dans la figure I.8, les RdP se distinguent par leur capacité à couvrir tout le cycle de conception préliminaire d'un système allant de la spécification, de l'analyse à l'évaluation des performances.

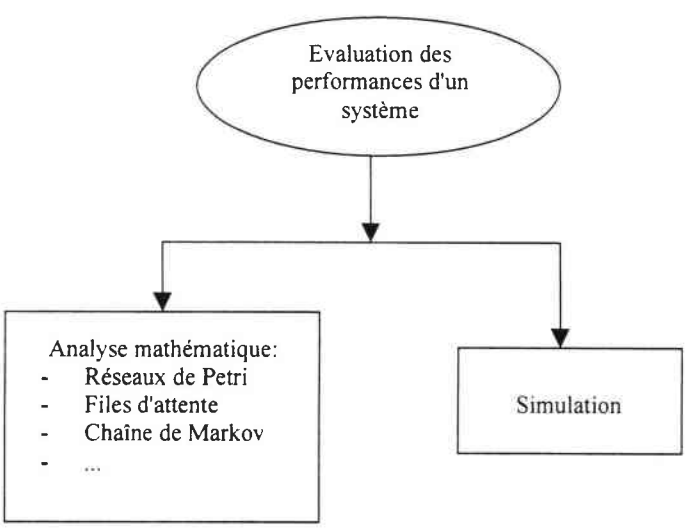

Figure I.8 : Evaluation des performances d'un système

Dans ce paragraphe, nous nous intéressons à l'application des RdP à l'ordonnancement des systèmes de production, plus spécifiquement à la modélisation et à l'optimisation de la production. Dans cet objectif, nous présentons par la suite, les principales notions de RdP et de RdP temporisés et la modélisation de problèmes d'ordonnancement. pour plus de détails, les lecteurs sont renvoyés aux états de l'art sur la théorie des RdP (Murata, 89) et sur leurs applications aux systèmes de production (Di Cesare et al., 92 ; Xie, 95 ; Proth et Xie, 96 ; Camus, 97).

I.6.1 Définitions de base

Un RDP est un graphe biparti composé de deux types de sommets : les places et les transitions. Les places sont représentées par des cercles et les transitions par des barres. Des arcs orientés relient les places aux transitions et les transitions aux places. Un arc ne relie jamais deux sommets de même nature. Chaque place peut contenir un ou plusieurs jetons, représentés par des marques. L'état d'un réseau est défini par le marquage. Un marquage est un vecteur à composantes entières positives ou nulles, dont la dimension est égale au nombre de places ; la n^{me} composante de ce vecteur représente le nombre de jetons qui figurent dans la place n du réseau. La dynamique du système est modélisée par les franchissements des transitions, définis ultérieurement, qui font évoluer le marquage. Un exemple illustratif d'un RDP est donné dans la figure I.9.

Plus formellement un RDP est un 5-uplet $R = (P, T, A, W, M_0)$ où :

- P est un ensemble fini de places, $P = \{p_1, p_2, \ldots, p_n\}$
- T est un ensemble fini de transitions, $T = \{t_1, t_2, \ldots t_m\}$
- A est l'ensemble fini des arcs, $A \subseteq (P \times T) \cup (T \times P)$
- W est la fonction poids attachée aux arcs, $W : A \rightarrow IN$
- M_0 est le marquage initial, $M_0 : P \rightarrow IN$
- $P \cap T = \emptyset$

Un RdP est dit ordinaire si $W(a)=1$, $\forall a \in A$.

I.6.2 Quelques définitions d'un RdP

Dans la suite, nous appelons parfois le graphe sous-jacent d'un RdP, c'est-à-dire $N = (P, T, A, W)$, le RdP non-marqué. Nous utilisons également la notation (N,M_0) pour désigner un réseau de Petri et nous l'appelons un RdP marqué.

Dans la suite nous utilisons les notations suivantes :

$^{\bullet}p$: ensemble des transitions d'entrée de la place p

$p^{\bullet}$: ensemble des transitions de sortie de la place p

$^{\bullet}t$: ensemble des places d'entrée de la transition t

$t^{\bullet}$: ensemble des places de sortie de la transition t

M(p) : le marquage de la place p, c'est-à-dire le nombre de jetons

W(p,t) : le poids de l'arc qui relie p à t

W(t,p) : le poids de l'arc qui relie t à p

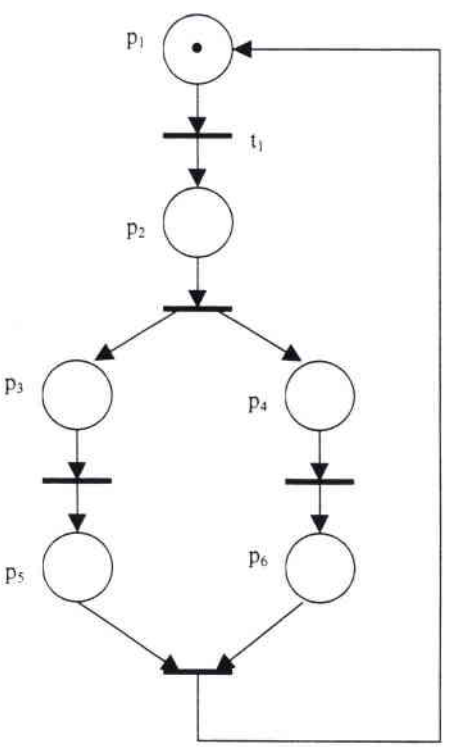

Figure I.9 : Exemple d'un RdP

Règles de validation et de franchissement d'une transition

Une transition t est franchissable si chaque place d'entrée p contient au moins un nombre de jetons égal au poids attaché à l'arc reliant p à t (dans le cas d'un RdP ordinaire ce

poids est égal à 1). On dit aussi que c'est transition tirable. Plus formellement, une transition t est franchissable pour un marquage M si, et seulement si,

$$M(p) \geq W(p,t), \forall\, p \in {}^{\bullet}t.$$

Le franchissement d'une transition franchissable est une opération instantanée et seulement une transition est franchie à la fois. L'opération de franchissement d'une transition t consiste à retirer W(p,t) jetons de toute place d'entrée (de t) $p \in {}^{\bullet}t$ et à ajouter W(t,p) jetons dans toute place de sortie $p \in t^{\bullet}$. Le franchissement d'une transition franchissable à partir d'un marquage M conduit à un nouveau marquage M'.

Le franchissement d'une transition conduit naturellement au franchissement d'une séquence de transitions. Un marquage M est dit atteignable à partir du marquage initial M_0 s'il existe une séquence de transitions franchissable transformant M_0 en M. Nous désignons par $A(N,M_0)$ l'ensemble des marquages atteignables d'un RdP $R=(N,M_0)$.

Une relation fondamentale des RdP est l'équation d'état. Cette relation s'applique à tout marquage atteignable M et est définie comme suit:

$$M = M_0 + C\ \vec{\sigma}$$

où σ est une séquence de transitions franchissable transformant M_0 en M, $\vec{\sigma}$ son vecteur comptage dont chaque élément correspond au nombre d'occurrences d'une transition dans σ ; $C=[C_{pt}]$ est la matrice d'incidence du RdP défini comme suit:

$$C_{pt} = \begin{cases} W(t,p), & \text{si } p \in t^{\bullet} \wedge p \notin {}^{\bullet}t \\ -W(p,t), & \text{si } p \notin t^{\bullet} \wedge p \in {}^{\bullet}t \\ W(t,p) - W(p,t), & \text{si } p \in t^{\bullet} \wedge p \in {}^{\bullet}t \\ 0, & \text{sinon.} \end{cases}$$

Transitions sources et transitions puits

Une transition source est une transition sans place d'entrée ; elle est toujours franchissable ou tirable. Une transition puits est une transition sans place de sortie et elle peut être tirée si elle est franchissable.

Dans la suite, on illustre ces deux types de transition à l'aide de la figure I.10. Dans la figure I.10, t_1 est une transition source et t_5 est une transition puits. Le marquage initial de cet exemple est $M_0 = [1,0,0,0]$.

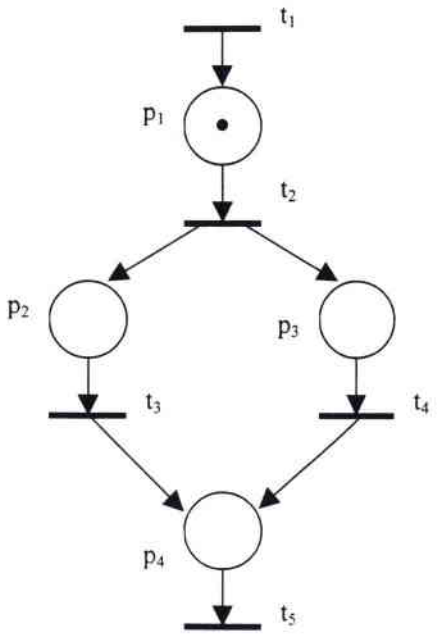

Figure I.10 : Transitions de source et de puits

Circuits élémentaires/boucles/verrous/trappes

Un circuit élémentaire est un chemin qui part d'un sommet (place ou transition) du RdP et y revient sans jamais rencontrer plus d'une fois le même sommet.

Une boucle (p,t) est telle que $p \in \{^\bullet t\}$ et $p \in \{t^\bullet\}$. C'est-à-dire que p est à la fois une place d'entrée et une place de sortie de la transition t. Un RdP est dit pur s'il ne contient aucune boucle.

Dans les RdP ordinaires, on utilise encore deux autres définitions : les verrous et les trappes. Un verrou, aussi appelé siphon, est un ensemble P_s de places où toute transition, qui a une place de sortie dans P_s, a au moins une place d'entrée dans P_s. Une trappe est un ensemble P_t de places où toute transition, qui a une place d'entrée dans P_t, a au moins une place de sortie dans P_t.

Propriétés comportementales des RdP

Soit $R=(N,M_0)$ un RdP marqué:

- R est vivant si, et seulement si, $\forall M \in A(R,M_0)$ il existe une séquence de transitions franchissables à partir de M contenant toutes les transitions ;
- R est réversible si, et seulement si $\forall M \in A(R,M_0)$ il existe une séquence de transitions transformant M en M_0 ;
- R est borné si, et seulement s'il existe un nombre $k > 0$ tel que , $\forall M \in A(R,M_0)$ $M(p) \leq k$.

I.6.3 RdP Temporisés

Il existe dans la littérature, plusieurs modèles de RdP temporisés. L'utilisation de la temporisation et sa localisation dépendent de l'application. Lorsqu'on associe la temporisation aux transitions, on obtient un RdP T-temporisé (Ramchandani, 74). Lorsque la temporisation est associée aux places, les RdP deviennent des RdP P-temporisés (Sifakis, 78). Dans notre cas, c'est-à-dire dans les systèmes de production, nous utiliserons les RdP P-temporisés déterministes, c'est-à-dire que des durées déterministes seront associées aux places.

Dans les RdP P-temporisés, on associe une durée aux places. Les jetons ont un temps de séjour minimal dans chaque place et, pendant ce temps, les jetons ne peuvent pas être utilisés pour le franchissement des transitions. Le franchissement d'une transition est instantané et s'effectue de manière identique que dans les RdP non temporisés.

Nous notons qu'il a été montré (Sifakis, 80) que les RdP P-temporisés et T-temporisés sont équivalents et que l'on passe aisément d'un modèle à un autre.

I.6.4 Les RdP et les Systèmes de Production

Nous avons donné, dans les paragraphes précédents, les notions de base et quelques définitions de RdP et de RdP P-temporisés. Nous montrons, dans ce paragraphe, leur utilisation dans les systèmes de production.

Le système de production à modéliser est un système de production à ressources multiples où chaque travail est composé par un ensemble d'opérations. Chacune de ces opérations est définie par un ensemble de ressources et son temps opératoire.

La modélisation par RdP P-temporisé d'un tel système peut être résumée en deux étapes :

Etape 1: elle consiste à représenter chaque travail par un chemin élémentaire qui représente son processus de production (gamme). Nous illustrons cette étape avec la figure I.11. Le RdP contient N_j+2 places, où N_j est le nombre des opérations du travail j et les deux places supplémentaires correspondent à la place de source et à la place de puits. La place de source correspond aux produits à fabriquer et la place de puits correspond à la réalisation complète du travail. La transition t_{jk} (k=0 à N_j) correspond au début de l'opération O_{jk+1}. Un temps nul est associé aux places de source et puits ; pour les autres places, on associe le temps opératoire de l'opération correspondante.

Etape 2: elle consiste à ajouter les contraintes de ressource au RdP P-temporisé. Chaque ressource sera représentée par une place r_i, i = 1, 2, ..., m contenant initialement un jeton (nous restons dans le cas de ressource unitaire, c'est-à-dire une unité de chaque type de ressource).

La figure I.12 donne le modèle RdP P-temporisé d'un système de production.

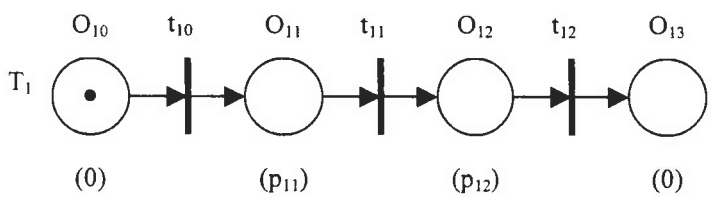

Figure I.11 : RdP d'un travail de deux opérations

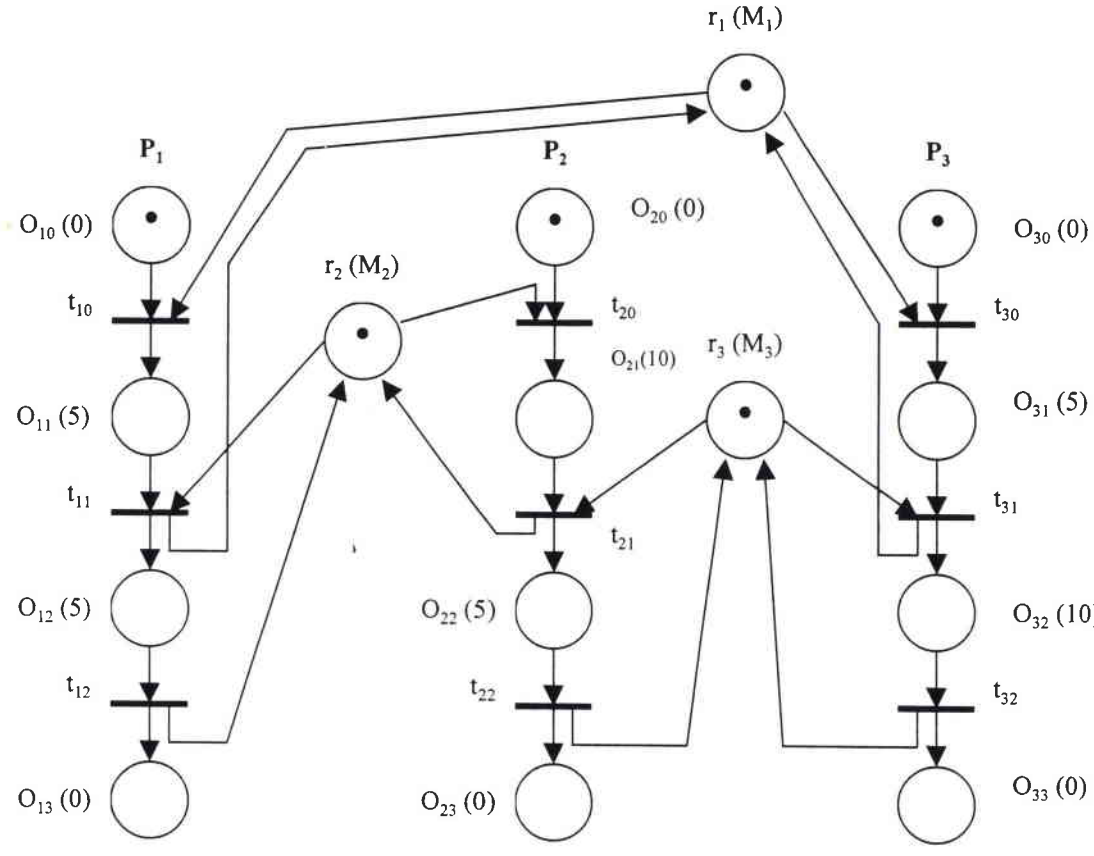

Figure I.12 : RdP P-temporisé du système à ressources multiples

I.7 Ordonnancement de systèmes de production

Nous présentons, dans ce paragraphe, les travaux qui prennent en compte les contraintes importantes d'un point de vue du système de production et qui sont ignorées dans la majorité des travaux sur l'ordonnancement. En particulier, nous nous intéressons à la prise en considération des opérations nécessitant des ressources multiples telles que les moyens de manutention, les robots, les zones de stockage à capacité très limitée, etc. Il s'avère que le problème de blocage devient important pour la coordination de toutes les ressources.

I.7.1 Job-shops hybrides avec opérations à ressources multiples et flexibilité de ressources

Nous présentons, dans ce paragraphe, les travaux de Dauzère-Pérès et al. (Dauzère-Pérès et Paulli, 97 ; Dauzère-Pérès et al., 98) sur l'ordonnancement des job-shops dont l'objectif est la prise en compte de certaines contraintes industrielles telles que la flexibilité des ressources et la nécessité de plusieurs ressources pour une opération.

Le premier article (Dauzère-Pérès et Paulli, 97) considère le problème d'ordonnancement des job-shops hybrides dans lesquels chaque travail nécessite une séquence d'opérations et chaque opération peut être effectuée par un sous-ensemble de machines avec différentes performances. Le problème consiste à affecter les opérations aux machines et à ordonnancer les opérations sur les machines de manière à minimiser le makespan. Les auteurs proposent, dans cet article, une extension des graphes potentiel-tâches pour représenter à la fois l'ordonnancement et l'affectation. Cette représentation a permis de définir de la même manière la réaffectation et le réordonnancement d'une opération. Une méthode tabou fondée sur cette relation de voisinage est ensuite proposée. Les résultats numériques sont excellents.

Le deuxième article (Dauzère-Pérès et al., 98) est une extension de l'article précédent. Plus précisément, on considère un ensemble d'opérations dont la relation de précédence est décrite par un graphe acyclique (connexe ou non). Chaque sommet de ce graphe peut avoir plusieurs prédécesseurs et plusieurs successeurs modélisant une opération assemblage/désassemblage. L'exécution d'une opération nécessite la présence simultanée de plusieurs ressources dont chacune est à sélectionner parmi un ensemble de ressources candidates. Le problème consiste à sélectionner les ressources candidates et à ordonnancer les opérations sur les ressources de manière à minimiser le makespan. A nouveau, une représentation par graphes potentiel-tâches est proposée. Une relation de voisinage des solutions fondée sur la nouvelle représentation est proposée. Cette relation de voisinage permet de relier une solution initiale quelconque à une solution optimale et évite les situations de blocage. Plus précisément, on définit une transformation de base qui concerne une opération i séquencée entre deux opérations u et v sur sa $k^{ième}$ ressource et la séquence entre deux autres opérations s et t sur sa $k'^{ième}$ ressource. Il est bien évident que certaines transformations conduisent à des cycles dans le graphe potentiel-tâches et, par conséquent, donnent des solutions non réalisables. Dauzère-Pérès et Paulli donnent une condition suffisante pour que les transformations conduisent à des solutions réalisables. Cette condition est également une condition nécessaire d'optimalité. Surtout, cette condition définit une structure de voisinage connective, c'est-à-dire qu'on peut transformer une solution quelconque en une solution optimale dans un nombre fini d'étapes. Une méthode tabou a été développée. L'optimalité asymptotique de cette méthode est assurée par la connectivité de la structure de voisinage. De plus, seul un nombre très limité de caractéristiques sont utilisées dans la liste de tabou. Des méthodes simples fondées sur des bornes inférieure et supérieure sont utilisées

pour déterminer les nouvelles solutions dans la recherche avec tabous. Les résultats numériques attestent l'efficacité de la méthode proposée.

I.7.2 Ordonnancement sans blocage d'un atelier automatisé

Ramaswany et Joshi (96) montrent que l'ordonnancement des ateliers de production automatisés diffère de manière significative de l'ordonnancement des job-shops classiques. La présence de blocage est l'un des problèmes critiques dans le pilotage d'un atelier automatisé. En pratique, les situations de blocage sont résolues en temps réel et/ou a posteriori, ce qui exige l'intervention humaine. L'objectif de Ramaswany et Joshi (96) est de montrer l'importance et le bénéfice de la prise en compte de blocage dans l'élaboration de l'ordonnancement.

Plus précisément, Ramaswany et Joshi considèrent un atelier automatisé composé de trois machines sur lesquelles on réalise quatre travaux. Le système pourrait être considéré comme un job-shop s'il n'y n'avait pas les contraintes suivantes : un robot est utilisé pour le transfert des travaux entre les machines ; chaque machine possède un stock tampon de capacité très limitée (de capacité 0, 1 ou 2).

Dans Ramaswany et Joshi (96), des modèles de programmation linéaire en nombre mixte ont été proposés pour ordonnancer le système dans les cas suivants : (i) avec des stocks tampons de capacité illimitée et sans moyen de transport ; (ii) sans stock tampon et sans moyen de transport ; (iii) sans stock tampon et avec prise en compte de moyen de transport ; (iv) avec des stocks tampons de capacité 1 et sans moyen de transport. Dans ces formulations, le robot et les stocks tampons sont modélisés comme des machines. La durée totale de l'ordonnancement est de 472 dans le cas (i), de 512 dans le cas (ii), de 560 dans le cas (iii) et de 502 dans le cas (iv).

De ces résultats, on constate que les ressources considérées comme secondaires dans les job-shops classiques peuvent avoir des impacts significatifs dans le pilotage d'un atelier automatisé. De plus, l'ordonnancement avec stocks tampons de capacité illimitée et sans moyen de transport peut conduire au blocage mortel dans un atelier automatisé. En effet, dans le cas de stocks tampons nuls, une situation de blocage se produit si le robot transfère un produit à une machine occupée puisque le même robot est nécessaire pour décharger de la machine le produit en cours de fabrication.

I.7.3 L'ordonnancement des réseaux de Petri par recherche empirique

En s'inspirant des procédures par séparation et évaluation, Lee et Di Cesare (Lee et Di Cesare, 94-a) ont proposé une méthode pour l'ordonnancement optimal des RdP avec un

marquage initial et un marquage final. Cette méthode s'apparente à l'algorithme A^* (Laurière, 87) connu en Intelligence Artificielle. Elle parcourt l'espace de marquages atteignables et s'arrête dès que le marquage final est atteint. Il est bien évident que cette méthode n'évite pas le problème d'explosion d'états et elle ne permet pas d'obtenir la solution optimale. Toutefois, pour guider la recherche vers une bonne solution, une évaluation f(m) est calculée pour chaque marquage m en cours d'examen. Cette évaluation s'écrit sous la forme suivante :

$$f(m) = g(m) + h(m)$$

où g(m) représente le coût partiel allant du marquage initial m_0 à m et h(m) est une évaluation du coût restant, c'est-à-dire du coût allant de m au marquage final m_f. h(m) est également appelée fonction heuristique et dépend évidement du critère à optimiser et de la structure du RdP.

Plus précisément, la méthode se résume par l'algorithme suivant dans lequel OPEN est la liste de marquages rencontrés mais non encore séparés et CLOSED la liste de marquages déjà séparés.

Etape 1 : Placer le marquage initial dans la liste OPEN

Etape 2 : Si OPEN est vide, alors terminer avec échec

Etape 3 : Retirer le premier marquage m de OPEN et le placer dans la liste CLOSED

Etape 4 : Si m est le marquage final, construire la séquence optimale de transitions du marquage initial au marquage final et terminer avec succès

Etape 5 : Trouver les transitions franchissables depuis le marquage m

Etape 6 : Générer les marquages successeurs m' de m et les pointeurs de m à m' correspondants. Calculer g(m') pout tout m'

Etape 7 : Pour tous les successeurs m' de m faire:

a. Si m' est déjà dans OPEN, alors diriger son pointeur vers le chemin menant au plus petit g(m').
b. Si m' est déjà dans CLOSED, alors diriger son pointeur vers le chemin menant au plus petit g(m'). Si ce pointeur est redirigé, alors déplacer m' dans OPEN.
c. Calculer h(m') et f(m') et placer m' dans OPEN.

Etape 8 : Trier OPEN dans l'ordre croissant de f(m).

Etape 9 : Aller à l'étape 2.

On trouve également, dans la littérature, des variantes de cette méthode en y ajoutant une procédure de *backtracking* contrôlée dans Xiong et Zhou (98) ou une forme quadratique de la fonction heuristique dans Jeng et al. (96).

A priori, cette méthode s'applique à tout problème d'ordonnancement des RdP avec un marquage initial et un marquage final. Parmi les applications, on trouve dans Lee et Di Cesare (94-a) la minimisation du makespan d'un job-shop avec des opérations nécessitant plusieurs ressources et avec flexibilité de ressource ; dans Lee et Di Cesare (94-b), l'ordonnancement des ateliers flexibles avec contraintes des AGV ; dans Jeng et al. (96), l'ordonnancement des ateliers flexibles avec des stocks tampons à capacité limitée, des AGV et des moyens de transport. Bien que la méthode s'applique à des RdP avec possibilité de blocage, les travaux ci-dessus concernent seulement des RdP sans blocage. Toutefois, Xiong et Zhou (98) ont appliqué cette méthode au problème d'ordonnancement d'un atelier flexible dû à Ramaswany et Joshi qui possède de blocages potentiels.

En guise de conclusion, cette méthode est très générale et s'applique à une grande variété des problèmes d'ordonnancement. Néanmoins, les fonctions heuristiques existantes ne concernent que le makespan et leur qualité semble très mauvaise. Bien que la méthode s'arrête dès qu'une solution a été trouvée, elle souffre également de l'explosion combinatoire et son application est intéressante pour de problèmes de petite taille mais peu efficace pour des problèmes de taille raisonnable.

I.7.4 Le lien entre la recherche opérationnelle et les réseaux de Petri

Richard (97) a traité de l'ordonnancement acyclique des systèmes de production avec contraintes de ressource. Pour cela, les Graphes d'Evénements avec Contraintes de Ressources (ou GETR) sont définis. Les GETR sont des graphes d'événements augmentés des places de ressource dont chacune contient initialement un jeton et exprime la contrainte d'une ressource unitaire. Les GETR sont des graphes d'événements augmentés proposés par Chu et Xie (Chu et Xie, 97) dont les propriétés peuvent être vérifiées à l'aide des propriétés structurelles. Ce RdP peut modéliser une grande quantité de systèmes de production avec contraintes de ressources. Afin de spécifier des systèmes de production de grande taille, Richard a proposé une procédure de synthèse pour modéliser systématiquement les travaux, les ressources, les outils, les stocks à capacité limitée, les stocks FIFO et diverses contraintes techniques. La procédure de synthèse conduit à un RdP sans blocage si le système modélisé est du type flow shop hybride. Pour trouver l'ordonnancement des GETR, Richard a proposé une approche utilisant CHIP, un langage de programmation sous contrainte. En utilisant les

GETR, le problème d'ordonnancement est encore un problème NP-difficile. C'est pour cette raison que Richard a également proposé une généralisation de la méthode sérielle, une méthode classique d'ordonnancement des projets, pour construire de bonnes solutions des problèmes de grande taille.

Il a également étudié l'ordonnancement des RdP temporisés séquentiels (ou STPN), une classe de RdP comportant une place d'exclusion mutuelle reliant à chaque transition. Dans cette nouvelle classe de RdP, la place d'exclusion mutuelle impose l'exécution séquentielle des transitions. Richard a démontré la NP-complexité de ce problème d'ordonnancement et il a proposé une méthode de résolution à deux étapes dont la première détermine le nombre de tirs de chaque transition à l'aide de la programmation linéaire et la deuxième étape vérifie l'existence d'une séquence de tirs correspondant à l'aide d'un algorithme de joueurs de jetons (*token players*). Ces deux étapes sont NP-complètes. Un effort est alors fait pour mettre en évidence de nouvelles conditions suffisantes d'accessibilité pour l'équation d'état d'un RdP. A partir de ces conditions d'accessibilité, Richard a démontré que tout programme linéaire en nombre entier peut être associé un STPN équivalent. Ces résultats sont ensuite étendus au cas continu, c'est-à-dire aux RdP continus et aux programmes linéaires avec variables continues.

I.7.5 "Hoist Scheduling" ou ordonnancement de robots de manutention en galvanoplastie

Ce problème d'ordonnancement a initialement été posé pour la gestion des lignes de traitement de surface de métaux. Il est connu sous le nom de *Hoist Scheduling* ou HSP. Il consiste à ordonnancer les mouvements du robot (ou des robots) de manutention pour charger ou décharger des cuves suivant une séquence opératoire donnée pour chaque produit. Les caractéristiques suivantes le différencient des problèmes d'ordonnancement classiques:

- le temps de passage d'un produit dans chaque cuve est borné par une fenêtre de temps, c'est-à-dire qu'il doit rester assez longtemps mais ne doit pas rester trop longtemps dans une cuve ;

- les temps des déplacements accomplis par les robots sont non négligeables et il faut gérer le parcours de chaque robot y compris les trajets à vide ;

- les collisions entre les robots sont interdites.

On trouve un état de l'art sur le problème HSP dans Manier et Baptiste (94). Nous nous contentons de donner quelques travaux représentatifs.

Deux approches sont considérées: l'approche prédictive et l'approche réactive. Dans les travaux sur l'approche prédictive, on se place dans le cas de grandes séries et on s'intéresse à l'ordonnancement périodique. La plupart des travaux modélisent le problème d'ordonnancement comme un problème de programmation linéaire en nombres mixtes. La résolution se fait soit en utilisant des logiciels de programmation linéaire (Philips et Unger, 76 ; Bracker et Chapman, 85), soit à l'aide des procédures par séparation et évaluation (Shapiro et Nuttle, 88 ; Lei et Wang, 89 ; Chen et al., 94).

L'approche réactive utilise des règles heuristiques pour déterminer le lancement des produits et choisir les mouvements des robots. Elle s'applique aux petites et moyennes séries mais pose le problème de respect de contraintes telles que les durées opératoires et la collision. Pour améliorer cet aspect, les RdP P- temporels dans lesquels le temps de séjours de jetons dans chaque place est borné ont été utilisés dans Collart-Dutilleul et Denat (97) pour modéliser et superviser les systèmes du type HSP.

I.7.6 Les systèmes non cycliques ou *on-line* et les RdP

Les systèmes non cycliques ou systèmes *on-line* sont des systèmes qui évoluent dans le temps en réponse aux demandes du monde extérieur.

Wang (95) a étudié ces systèmes et leur modélisation à l'aide des RdP. Il a proposé une approche modulaire qui utilise les RdP et leurs propriétés pour traiter les systèmes de production de grande taille. Une telle approche conduit à une gestion hiérarchisée de la planification à court terme et de l'ordonnancement.

La planification à court terme a comme objectif d'établir la quantité de chaque produit à fabriquer pendant chaque période élémentaire en fonction de la capacité globale du système. L'ordonnancement décide, à l'intérieur de la première période élémentaire, des ressources qui vont exécuter les différentes opérations et le début de chacune.

Wang a utilisé l'approche *horizon glissant* pour la planification à court terme à horizon H. Cette approche fonctionne comme suit : au début de la seconde période élémentaire, on recommence la planification sur l'horizon H en tenant compte des nouvelles données qui sont intervenues depuis le calcul précédent et des produits déjà ordonnancés sur cet intervalle, on calcule l'ordonnancement sur la première période élémentaire et ainsi, successivement.

Le modèle utilisé pour la planification est l'ensemble des modèles des gammes de fabrication des produits. La gamme de fabrication d'un produit, du point de vue de la gestion de production, est la succession des opérations à effectuer avec, pour chacune d'elles, la liste des machines sur lesquelles elle peut être effectuée et le temps nécessaire pour l'exécuter sur chacune des ressources.

Le modèle RdP de la gamme de fabrication d'un produit comporte des transitions sources temporisées à zéro. Les franchissements de ces transitions représentent les lancements en fabrication de nouvelles unités de produits. Ce RdP comporte aussi des transitions puits dont le franchissement représente la fin de fabrication. Les RdP doivent être à sorties contrôlables, c'est-à-dire qu'il est toujours possible de franchir une transition de puits quelconque sans être obliger de franchir d'autres transitions de puits (Wang et Xie, 96 ; Proth et al., 97).

Le modèle de planification est constitué de composantes connexes disjointes, chacune de ces composantes étant un RdP à sortie contrôlable sans place de ressource. Chaque composante connexe est le modèle d'une gamme de fabrication et ne comporte qu'une seule transition puits.

Le modèle d'ordonnancement est obtenu en complétant le modèle pour la planification de la manière suivante :

- pour chaque machine, on introduit une place contenant un jeton. Ces places sont les places de ressources.

- on introduit les arcs qui font de chaque place ressource, la place d'entrée et de sortie de toutes les transitions correspondantes à cette machine.

Du fait que les RdP élémentaires peuvent conduire à des modèles de grande taille, Wang a utilisée l'approche modulaire suivante:

- décomposer le système de fabrication en modules ;

- modéliser ces modules à l'aide des RdP et des places d'interface ;

- vérifier que les propriétés qualitatives de chacun des modules sont les propriétés souhaitées ;

- simplifier chacun de ces modules en préservant les propriétés qualitatives ;

- intégrer les modules simplifiés tout en préservant les propriétés qualitatives.

Le modèle intégré, composé de modules simplifiés, servira à la planification à moyen terme et donnera les quantités à fabriquer par chaque module pendant chaque période élémentaire. On pourra effectuer la planification à court terme et l'ordonnancement au niveau de chaque module en utilisant les algorithmes proposés par Savi (94).

En conclusion, les RdP peuvent être utilisés pour gérer des systèmes non cycliques de taille quelconque. Pour atteindre cet objectif, les systèmes sont décomposés en modules contrôlables et ces derniers peuvent être simplifiés pour obtenir des modèles intégrés agrégés et utilisables au niveau haut. Ainsi, les RdP constituent un moyen d'intégration de la planification et de l'ordonnancement.

I.7.7 Ordonnancement sans attente et sans en-cours

C'est une classe importante de problèmes d'ordonnancement qui se distingue par les contraintes du type sans attente (*no-wait*) ou sans en-cours (*blocking*). Dans un système sans attente, chaque produit doit être traité sans interruption du début jusqu'à la fin. Dans un système sans en-cours, un produit ayant été traité sur une machine reste bloqué sur cette machine jusqu'à ce que la machine suivante devienne libre. Ces situations se produisent souvent dans la sidérurgie où le traitement de lingots à très haute température doit être réalisé de manière continue. On trouve ces caractéristiques également dans les systèmes pétrochimiques, pharmaceutiques, cosmétiques, etc.

Une étude bibliographique détaillée est donnée dans Hall et Sriskandarajah (96). Nous donnons, dans ce paragraphe, quelques résultats importants concernant les flow-shops qui représentent la majorité des systèmes réels.

D'abord, pour les systèmes sans attente ou sans en-cours, l'ordre de passage des produits sur les machines est identique quelle que soit la machine. Alors, l'ordonnancement revient à déterminer une permutation des produits et leur date de lancement. On montre que la minimisation du *makespan* se transforme en un problème du voyageur du commerce. De plus, le problème de minimisation du *makespan* est polynomial pour le cas de deux machines et se résout par l'algorithme de Gilmore et Gomory (Gilmore et Gomory, 64). Malheureusement, le problème devient NP-difficile au sens fort à partir de trois machines dans un système sans attente ou sans en-cours.

Les problèmes d'ordonnancement sans attente et sans en-cours peuvent être considérés comme des cas particuliers de procédés chimiques *multibatchs*. Des nombreux travaux ont été consacrés à l'optimisation de ces systèmes composés des réacteurs et des conteneurs reliés par des conduits de canalisation. La planification et l'ordonnancement de ce type de système nécessitent la prise en compte de "recettes" de fabrication de produits, la disponibilité des matières premières telles que les additifs, les capacités des réacteurs et des conteneurs. Pour l'ordonnancement à court terme, on parle des systèmes ZW (*zero wait*), FIS (*finite intermediate storage*), UIS (*unlimited intermediate storage*) et des systèmes hybrides. Les systèmes ZW correspondent aux systèmes sans attente ou sans en-cours. On trouve une littérature très riche sur ce sujet dans Rippin (93) ; Sahinidis et Grossmann (91) ; Ku et Karim, (90).

I.7.8 Ordonnancement des cellules robotisées

Il s'agit de l'ordonnancement d'une cellule composée de deux, trois ou quatre machines alimentées par un robot. La cellule fabrique de manière répétitive un ensemble de produits. Le problème consiste à déterminer le cycle des mouvements d'un robot et le séquencement des produits afin de minimiser le temps de cycle, c'est-à-dire maximiser la productivité.

Sethi et al. (92) ont trouvé la séquence optimale des mouvements du robot pour le cas de deux machines fabriquant un seul produit. Un algorithme polynomial a été proposé par Hall et al. (97) pour le cas de deux machines fabriquant plusieurs produits. Pour le cas de trois machines, Sethi et al. (92) ont réduit le mouvement du robot à six séquences possibles. Malheureusement, le problème d'ordonnancement des produits pour deux de ces six séquences reste NP-difficile au sens fort (voir Hall et al., 97). Une méthode du type *Branch & Bound* a été proposée par Chen et al. (94) pour les deux cas difficiles. Hall et al. (97) montrent également que le problème d'ordonnancement devient NP-difficile au sens fort, pour toute séquence du robot, pour le cas de plus de trois machines. Finalement, nous notons que Zuberek (95) a utilisé les RdP temporisés pour modéliser et évaluer chaque séquence du robot et l'ordonnancement des produits.

I.8 Détection, prévention, élimination et correction des blocages

La mise en jeu simultanée d'une grande variété de ressources dans les systèmes de production modernes complique significativement leur coordination et pose le problème de blocage. En effet, une situation de blocage se produit si, par exemple, un robot de manutention transporte un produit vers une machine déjà occupée et le même robot est nécessaire pour le déchargement de la machine.

Le phénomène de blocage a été largement étudié pour les systèmes d'exploitation informatiques. Il est reconnu que les conditions suivantes sont nécessaires pour les blocages (Coffman et al., 71) : (1) exclusion mutuelle, (2) non préemption, (3) retenir et attendre et de (4) attente circulaire.

Pour éviter les blocages, il suffit de s'assurer qu'au moins une des quatre conditions est fausse. Dans les systèmes de production, les deux premières conditions sont généralement vraies. Le blocage ne se produit pas dans les systèmes de production traditionnels tels que les job-shops et les flow-shops, car la troisième condition est fausse. En effet, dans ces systèmes, une machine nécessaire pour la fabrication d'un produit est libérée dès la fin de l'opération correspondante et ceci, quelle que soit la disponibilité de la machine suivante. Malheureusement cette condition "retenir et attendre" est présente dans les systèmes

automatisés car la capacité de stockage est souvent très faible et un moyen de manutention est nécessaire pour charger et décharger une machine. Eviter le blocage dépend alors fortement de la dernière condition.

Pour les systèmes de production, les solutions pour faire face au problème de blocage sont la prévention, le contrôle en temps réel (*deadlock avoidance*) et la correction a posteriori (*deadlock recovery*). La prévention consiste à définir a priori pour chaque état du système, une politique d'allocation de ressources afin d'éliminer les blocages. Cette politique est nécessairement statique. Par contre, le contrôle en temps réel est une stratégie dynamique et détermine en temps réel les allocations de ressource admises et les allocations à interdire. La correction a posteriori consiste à définir les stratégies à adopter pour sortir d'une situation de blocage. Nous nous limitons dans ce paragraphe à la prévention et au contrôle en temps réel.

Les outils les plus utilisés dans la littérature sont les RdP et la théorie des graphes. Nous présentons, dans un premier temps, les travaux utilisant les RdP. Viswanadham et al. (90) ont étudié la prévention et le contrôle en temps réel. La prévention s'appuie sur le graphe de marquage et les transitions conduisant au blocage sont interdites. Le contrôle en temps réel se fait par anticipation et admet toute transition qui ne conduit pas à un blocage dans un nombre donné d'étapes (*finite look ahead*). Il est bien évident que cette approche ne garantit pas l'élimination totale de blocage. Banaszak et Krogh (90) considèrent un système avec des travaux de différents types. Chaque travail nécessite une suite d'opérations et, à chaque opération, est associée une ressource. Cette ressource reste occupée jusqu'à la disponibilité d'une ressource nécessaire pour l'opération suivante. Une politique de contrôle en temps réel a été proposée. Cette politique, connue sous le nom de BKDAA, est une extension de l'algorithme de Banker pour les systèmes d'exploitation qui admet un travail dans le système si toutes les ressources nécessaires pour sa réalisation sont disponibles ; ces ressources sont alors réservées même si elles ne sont pas nécessaires immédiatement. BKDAA étend l'algorithme de Banker en décomposant le processus d'un travail en sous-processus, appelés zones. Chaque zone est composée d'une suite d'opérations nécessitant des ressources partagées suivie d'une suite d'opérations utilisant des ressources dédiées. La politique BKDAA admet un jeton dans une nouvelle zone si le nombre de jetons dans la zone ne dépasse pas le nombre de ressources dédiées. Cette politique a ensuite été améliorée par divers auteurs (Hsieh et Chang, 94 ; Xing et al., 96). Ezpeleta et al. (95) ont généralisé le modèle RdP de Banaszak et Krogh en tenant compte de la flexibilité de routage et ils ont proposé une politique de prévention sous-optimale fondée sur les propriétés des siphons. Les siphons ont été également utilisés pour définir une politique de contrôle en temps réel pour le même modèle de RdP (Barkaoui et Ben Abdallah, 94).

Cho et al. (95) ont considérés le modèle de production proposé par Banaszak et Krogh dans lequel chaque travail est réalisé en visitant une suite de ressources. Les ressources sont supposées toutes différentes les unes des autres et il n'y pas de stock tampon entre les ressources. Ils ont proposé une représentation graphique de l'état du système, appelée graphe

d'état, dans laquelle les sommets correspondent aux ressources ; un sommet est étiqueté de (j,k) si la ressource correspondante est utilisée pour l'opération k du travail j, un arc étiqueté (j,k) relie deux sommets r_1 et r_2 si le travail j passe de la ressource r_1 à la ressource r_2 à la fin de l'opération k. Des conditions nécessaires et/ou suffisantes ont été proposées pour détecter les situations de blocage et les situations de blocage éminent qui conduisent au blocage quelle que soit l'évolution ultérieure. A partir du graphe d'état ci-dessus, Fanti et al. (97) ont proposé un autre graphe permettant la détection aisée du blocage de second niveau qui n'est pas un blocage immédiat mais conduit au blocage dans une seule transition. Des politiques de contrôle en temps réel ont été proposées pour éliminer tous les blocages.

Reveliotis et al. (97) ont proposé un modèle de système de production très général, appelé RAS conjonctif, dans lequel chaque travail nécessite une suite d'opérations et le besoin en ressources de chaque opération est décrit par un vecteur qui donne le nombre de ressources de chaque type nécessaire. Compte tenu de la complexité du problème, ils ont étudié les politiques de contrôle en temps réel qui éliminent tous les blocages et qui sont polynomiales.

I.9 Conclusion

De cette étude bibliographique, nous constatons que la plupart des travaux en ordonnancement ignorent les aspects importants des systèmes de production automatisés suivants :

- variété de ressources,
- possibilité de blocage,
- opérations nécessitant simultanément plusieurs ressources.

Les travaux portant sur la détection et l'élimination de blocage ne prennent pas en compte les performances quantitatives du système. La plupart des travaux sur l'ordonnancement des systèmes de production à l'aide des RdP s'appuient sur l'exploration des graphes de marquages et, par conséquent, souffrent de l'explosion combinatoire.

Chapitre II

Job-Shops Multi-Ressources avec Blocage : définition et étude de la complexité

II.1 Introduction

La course à la productivité, la recherche d'une meilleure qualité et de coût le plus bas obligent les entreprises à automatiser, à intégrer et à rendre plus flexibles leurs systèmes de production. Les réponses sont d'abord technologiques grâce à des machines plus performantes utilisant la technologie numérique, des machines flexibles capables de passer rapidement d'une production à une autre, des moyens de transport automatisés tels que les AGV (*Automated Guided Vehicles*), etc. De plus en plus, les entreprises automatisent partiellement, voire entièrement, leurs moyens de production clés. Les moyens de production sont aujourd'hui très riches et variés, dépassent largement la fonction de transformation associée, traditionnellement, aux moyens de production. On trouve, dans un système de production moderne, des machines à commande numérique, une grande variété de moyens de transport automatisés (convoyeurs, AGV, ponts roulants), des robots pour la manutention et/ou l'assemblage, des installations automatisées pour le stockage des produits et/ou des composants ainsi qu'une variété impressionnante d'outils tels que les outils de découpe, les outils d'usinage, les palettes et les outils de fixation. Un autre facteur important est la réduction importante du nombre de chaque moyen de production en raison des coûts souvent très élevés. Le fonctionnement d'un tel système pose le délicat problème de la coordination de l'ensemble des ressources, problème auquel nous nous attaquons. Pour une description détaillée des systèmes flexibles, voir Stecke (85), Kusiak (90), Sawik (90), Solot et Vliet (94).

Comme souligné par Ramaswany et Joshi (96), la planification et le contrôle d'un système automatisé de production diffèrent considérablement des problèmes de l'ordonnancement étudiés dans la littérature traditionnelle que nous avons présentés dans le chapitre I. La plupart des travaux sur l'ordonnancement ne considèrent que les produits et les machines et ignorent souvent la variété des ressources mises en jeu dans un système de production. Dans ces travaux, on suppose souvent que les temps de transport soient négligeables ou bien constants, quel que soit l'ordonnancement. En pratique, la gestion des moyens de manutention est très délicate et une mauvaise gestion peut perturber de manière significative la production. Une autre hypothèse, faite explicitement ou implicitement dans la quasi-totalité des travaux, est l'existence des stocks tampons à capacité illimitée. Cette hypothèse est fausse dans le cas d'un système automatisé. Pire encore, l'ordonnancement obtenu sous cette hypothèse peut conduire au blocage mortel du système, ce qui exigera l'intervention humaine pour débloquer le système.

Pour prendre en compte la variété des ressources et le problème de blocage, l'approche la plus souvent utilisée est hiérarchique. L'affectation des ressources est faite de manière progressive, selon leur importance. Par exemple, Sawick (93) a proposé une approche à deux niveaux qui détermine d'abord un ordonnancement en prenant compte les capacités des machines et modifie ensuite cet ordonnancement pour prendre en compte les moyens de transport. Dans la quasi-totalité des travaux, l'ordonnancement est calculé sans prendre en compte le problème de blocage et on résout le problème de blocage en temps réel en

interdisant le lancement en fabrication des produits pouvant conduire au blocage ou en appliquant des procédures de correction pour sortir des situations de blocage. Les méthodes utilisées pour faire face au blocage sont les méthodes de détection, prévention, contrôle en temps réel et correction que nous avons présentés dans le chapitre I.

Contrairement aux approches mentionnées, nous proposons, dans cette thèse, une approche intégrée d'ordonnancement pour prendre en compte simultanément la variété des ressources et le problème de blocage. Les deux caractéristiques saillantes du modèle que nous proposons sont : (i) les opérations nécessitant simultanément des ressources de différents types que nous appelons opérations à ressources multiples ; (ii) la contrainte "retenir et attendre" pour le passage d'une opération à l'opération suivante du même travail, c'est-à-dire que les ressources nécessaires pour une opération ne sont libérées qu'au début de l'opération suivante. La caractéristique (i) permet la prise en considération de l'ensemble de ressources. La caractéristique (ii) correspond mieux à la réalité industrielle que l'hypothèse utilisée dans les travaux existants qui supposent que les ressources sont libérées à la fin de chaque opération, ceci quel que soit l'état des ressources nécessaires pour l'opération suivante.

Dans la suite de ce chapitre, nous définissons les systèmes du type job-shop Multi-Ressources avec Blocage et formulons le problème d'ordonnancement. Nous donnons ensuite une classification de ces nouveaux systèmes selon le besoin en ressources des opérations. Nous montrons que les job-shops Multi-Ressources avec Blocage fournissent un cadre général pour étudier l'ordonnancement des systèmes industriels car ils incluent comme cas particuliers de nombreux problèmes abordés dans la littérature. Nous terminons le chapitre par une étude détaillée de la complexité du problème d'ordonnancement des job-shops Multi-Ressources avec Blocage en nous appuyant sur les résultats connus pour des cas particuliers et en développant des résultats nouveaux pour les cas non couverts dans la littérature existante.

II.2 Job-shops Multi-Ressources avec Blocage et leur ordonnancement

Dans la théorie de l'ordonnancement des systèmes de production, l'ordonnancement des job-shops est un des problèmes les plus étudiés car d'une part, les job-shops sont couramment rencontrés dans la pratique et d'autre part, leur ordonnancement est un problème de complexité formidable. Formellement, un job-shop est un atelier dans lequel les opérations d'un produit (ou travail) suivent un ordre de production non nécessairement identique pour tous les produits. L'importance des job-shops et leur ordonnancement sont les raisons-d'être de nombreuses méthodes exactes et approchées, proposées dans la littérature (voir l'excellent état de l'art élaboré par Blazewicz et al., 96).

Comme il a été constaté dans la littérature, l'évolution des systèmes de production rend les modèles théoriques, tels que les flow-shops et les job-shops, inapplicables. Comme nous l'avons mentionné à maintes reprises, une des caractéristiques fondamentales des systèmes de production est la mise en jeu d'une grande variété de moyens de production afin d'assurer différentes fonctions d'un système automatisé ou semi-automatisé. Une autre caractéristique due au partage des ressources de production, telles que les stocks tampons à capacité très limitée et les moyens de manutention, est la possibilité de blocage mortel suite à une mauvaise coordination des ressources. Cette caractéristique est nouvelle et n'apparaît pas dans les systèmes classiques tels que les flow-shops et les job-shops.

L'objectif de ce paragraphe est d'introduire une généralisation des job-shops prenant en compte la variété des ressources impliquées dans une production et le phénomène de blocage mortel dû au partage des ressources. Nous appelons les nouveaux systèmes job-shops Multi-Ressources avec Blocage (ou job-shops MRB en abréviation). Dans la suite, nous définissons formellement les ressources, les opérations, les travaux et l'ordonnancement d'un job-shop MRB.

Ressources: De l'état de l'art du chapitre I, nous constatons que la plupart des travaux en ordonnancement des systèmes de production ne prennent en compte que les machines comme ressources. Traditionnellement, les autres ressources, telles que les outils, les moyens de manutention et les zones de stockage, sont considérées comme des ressources secondaires. On espère que l'ordonnancement calculé avec contraintes de machines peut être adapté pour prendre en compte toutes les ressources secondaires sans trop dégrader la solution. Malheureusement, comme constaté dans Ramaswany et Joshi (96), non seulement la solution peut être dégradée de manière significative pour un système automatisé réaliste mais certains ordonnancements peuvent conduire au blocage mortel et, par conséquent, sont non réalisables. Contrairement aux modèles classiques d'ordonnancement, un job-shop MRB prend en compte à la fois les ressources principales et aussi les ressources secondaires. Il est alors naturel d'avoir plusieurs ressources du même type.

Un job-shop MRB est composé d'un ensemble de ressources de différents types. Il peut exister plusieurs unités de ressources du même type. Plus formellement, soit $R = \{1, 2, ..., r, ..., m\}$ l'ensemble de ressources où m est le nombre de types de ressources. Soit H_r le nombre de ressources du type r. Nous soulignons que la possibilité d'avoir plusieurs ressources du même type distingue les job-shops MRB des job-shops classiques tout comme elle distingue les flow-shops hybrides des flow-shops classiques.

Opérations: La notion d'opérations est plus générale dans ce travail que dans la littérature classique de l'ordonnancement. Traditionnellement une opération est une transformation réalisée à l'aide d'une machine. Dans ce travail, nous considérons

comme des opérations toutes les phases de la fabrication d'un produit, y compris les manutentions, l'attente dans un stock tampon, le chargement/déchargement d'une machine, etc.

Pour représenter la variété des ressources nécessaires pour une opération O_{jk}, nous supposons que chaque opération O_{jk} a besoin d'un ensemble de ressources $R_{jk} \subseteq R$ et son temps opératoire est p_{jk} unités de temps. Une opération O_{jk} peut avoir besoin de plusieurs unités de ressources d'un type $r \in R_{jk}$. Soit h_{jkr} le nombre d'unités nécessaires avec la convention $h_{jkr} = 0, \forall r \notin R_{jk}$. On peut alors représenter le besoin en ressources d'une opération O_{jk} par un vecteur $\mathbf{h}_{jk}$.

Nous avons ici une des caractéristiques saillantes d'un job-shop MRB dans lequel une opération peut nécessiter de multiples ressources et de multiples unités de chacune. Nous appelons ce type d'opérations les opérations MRMU (Multi-Ressources et Multi-Unités). Un cas intéressant est le cas où $R_{jk} = \phi$. Il peut représenter la date de disponibilité d'un produit, le délai de transport sans contrainte de ressource de transport, etc. Le cas $p_{jk} = 0$ peut représenter l'attente d'un produit avec des ressources en possession si $R_{jk} \neq \phi$ et libéré de toute ressource si $R_{jk} = \phi$.

Travaux: Dans un job-shop MRB, il y a un ensemble J de travaux à réaliser avec $J = \{J_1, J_2, ..., J_j, ..., J_n\}$, où n est le nombre de travaux. Chaque travail J_j nécessite une séquence d'opérations $\{O_{j1}, O_{j2}, ..., O_{jk}, ..., O_{jNj}\}$, où N_j est le nombre d'opérations du travail J_j. Nous utilisons la notation suivante pour représenter un travail :

$$J_j = \{(R_{j1}, \mathbf{h}_{j1}, p_{j1}), (R_{j2}, \mathbf{h}_{j2}, p_{j2}), ..., (R_{jNj}, \mathbf{h}_{jNj}, p_{jNj})\}$$

où R_{jk} est l'ensemble des types de ressources nécessaires pour l'opération (j,k), c'est-à-dire $R_{jk} = \{r \in R / h_{jkr} > 0\}$.

Une autre caractéristique saillante des job-shops MRB est la propriété "Retenir et Attendre" qui définit les conditions de passage d'une opération à l'opération suivante du même travail.

Propriété "Retenir et Attendre": L'ensemble des ressources $\mathbf{h}_{jk}$ nécessaires à une opération (j,k) sont retenues jusqu'au début de l'opération suivante (j,k+1) et les ressources $\{(h_{jkr} - h_{jk+1r})\ r / r \in R_{jk}, h_{jkr} - h_{jk+1r} > 0\}$ qui ne sont plus nécessaires sont libérées après le début de (j,k+1).

Cette propriété de la succession d'opérations diffère fondamentalement des modèles classiques d'ordonnancement dans lesquels les ressources nécessaires à une opération O_{jk} sont libérées à la fin de O_{jk} quelle que soit la disponibilité des ressources nécessaires pour l'opération suivante O_{jk+1}. Comme nous l'avons mis en évidence dans le chapitre I, la propriété "Retenir et Attendre" est un phénomène courant dans les systèmes automatisés de production, à cause des ressources de manutention et des stocks tampons à capacité très limitée.

Pour résumer, un job-shop MRB est caractérisé par (i) la ressource à multiples unités ; (ii) les opérations MRMU ; (iii) la propriété "Retenir et Attendre" pour la succession des opérations.

Dans la suite de ce paragraphe, nous formulons le problème d'ordonnancement d'un job-shop MRB. Le critère considéré est la minimisation de la durée totale, c'est-à-dire le makespan. Le problème consiste à choisir, pour chaque type de ressource $r \in R$, une séquence π_r d'entrée des opérations nécessitant les ressources r et la date de début S_{jk} pour chaque opération (j,k) afin de minimiser le makespan.

Plus formellement, le problème d'ordonnancement d'un job-shop MRB est définit comme suit :

$$\text{Min } \underset{j}{\text{MAX}} \{C_j\},$$

sous les contraintes suivantes :

$$C_j = S_{jNj} + p_{jNj}\ ,\ \forall 1 \le j \le n \tag{2.1}$$

$$S_{jk} + p_{jk} \le S_{jk+1},\ \forall 1 \le j \le n,\ \forall 1 \le k \le N_j \tag{2.2}$$

$$\sum_{(j,k)} h_{jkr}\, \mathbf{1}\{S_{jk} \le t < S_{jk+1}\} \le H_r,\ \forall t\ ,\ \forall\, r \in R \tag{2.3}$$

$$S_{\pi r[i]} \le S_{\pi r[i+1]}\ ,\ \forall\, r \in R,\ \forall i \tag{2.4}$$

Les séquences d'entrée π_r ne conduisent pas au blocage (2.5)

où les relations (2.1) définissent la date de fin de chaque travail, les relations (2.2) correspondent aux contraintes de précédence des opérations. Dans les relations (2.3), h_{jkr} ressources du type r sont utilisées pour l'opération (j,k) ou sont retenues en attendant le début de l'opération suivante (j,k+1) si $S_{jk} \le t < S_{jk+1}$. Les relations (2.3) correspondent donc aux contraintes de capacité de ressources. Les relations (2.4), dans lesquelles $\pi_r[i]$ désigne la $i^{ème}$ opération de la séquence d'entrée π_r, assurent la cohérence entre les dates de début et les séquences d'entrée des ressources. Les contraintes (2.5) concernent l'absence de blocage et ne

peuvent pas être modélisées explicitement en termes de relations mathématiques. Nous utilisons ultérieurement les RdP pour exprimer ces contraintes.

A première vue, il semblerait que les contraintes (2.1) - (2.4) impliquent les contraintes (2.5). Ceci est vrai pour les systèmes traditionnels mais il est faux pour les job-shops MRB. Pour illustrer cela, considérons un job-shop MRB de deux ressources $\{r_1\ r_2\}$ et de deux travaux $\{J_1\ J_2\}$. J_1 et J_2 nécessitent chacun deux opérations. J_1 nécessite la ressource r_1 pour sa première opération et la ressource r_2 pour sa deuxième opération. J_2 nécessite la ressource r_2 pour sa première opération et la ressource r_1 pour sa deuxième opération. Les temps opératoires sont tous égaux et $p_{jk} = 1$. Ce job-shop MRB peut être considéré comme un job-shop sans en-cours. L'ordonnancement $S_{11} = 0$, $S_{12} = 1$, $S_{21} = 0$ et $S_{22} = 1$ satisfait les contraintes de capacité et de précédence (2.1) - (2.4). Selon cet ordonnancement, les deux travaux J_1 et J_2 commencent simultanément leur première opération. A l'instant $t = 1$, pour pouvoir commencer leur deuxième opération, J_1 a besoin de la ressource r_2 qui est retenue par J_2 et J_2 a besoin de la ressource r_1 qui est retenue par J_1. Nous sommes en face d'une situation de blocage. L'ordonnancement est donc irréalisable. Nous illustrons cette situation dans la figure II.1, en utilisant le diagramme de Gantt.

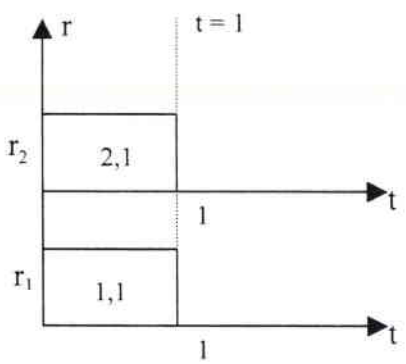

Figure II.1 : Une situation de blocage d'un ordonnancement d'un job-shop MRB

II.3 Classification des job-shops Multi-Ressources avec Blocage

Dans ce paragraphe, nous donnons une classification des job-shops MRB et des opérations selon le nombre de ressources, les besoins en ressources et les temps opératoires.

D'abord, nous distinguons le cas sans ressources identiques et le cas avec ressources identiques, appelés respectivement Systèmes à Ressources Unitaires et Systèmes à Ressources Multiples. Plus précisément,

- un Système à Ressources Unitaires (SRU) est un job-shop MRB dans lequel $H_r = 1$, $\forall\, r \in R$, c'est-à-dire que les ressources sont toutes différentes.

- un Système à Ressources Multiples (SRM) est un job-shop MRB dans lequel $H_r > 1$, pour au moins un $r \in R$, c'est-à-dire que les ressources ne sont pas toutes différentes.

Il est bien évident que, dans un SRU, le besoin en ressources d'une opération (j,k) est entièrement caractérisé par R_{jk}. En effet, dans un SRU, $h_{jkr} = 1$, $\forall\, r \in R_{jk}$. Nous pouvons alors simplifier la représentation du processus de fabrication d'un travail comme suit :

$$J_j = \{(R_{j1}, p_{j1}) , (R_{j2}, p_{j2}) , ..., (R_{jNj}, p_{jNj})\}.$$

Nous donnons maintenant une classification des opérations selon leur besoin en ressources et leur temps opératoire. Plus précisément, une opération est appelée :

- Opération ZRZD (*Zero Resource Zero Delay*) si $R_{jk} = \phi \wedge p_{jk} = 0$.
  Une opération ZRZD correspond aux attentes des travaux dans les modèles d'ordonnancement classiques. Pendant une opération ZRZD, le travail est libéré de toute ressource.

- Opération ZRD (*Zero Resource with Delay*) si $R_{jk} = \phi \wedge p_{jk} > 0$. Une opération ZRD est similaire à une opération ZRZD, sauf qu'elle doit durer au moins p_{jk} unités de temps. Elle peut représenter le délai de transport d'un produit dans le cas où les contraintes de ressources de transport ne sont pas prises en compte.

- Opération SRSU (*Single Resource Single Unit*) si $R_{jk} = \{r\} \wedge h_{jkr} = 1$. La plupart des opérations des modèles d'ordonnancement classiques sont des opérations SRSU.

- Opération SRMU (*Single Resource Multi-Units*) si $R_{jk} = \{r\} \wedge h_{jkr}$ quelconque. Les opérations SRMU permettent la représentation des contraintes de stockage.

- Opération MRSU (*Multi-Resource Single Unit*) si $|R_{jk}|$ quelconque $\wedge\, h_{jkr} = 1$, $\forall r \in R_{jk}$. Les opérations MRSU permettent la prise en compte de la variété de ressources de la production. Elles sont également utilisées dans l'ordonnancement des travaux multi-processeurs dans un environnement de calcul parallèle avec des processeurs dédiés.

- Opération MRMU (*Multi-Resource Multi-Units*) si $|R_{jk}|$ quelconque $\wedge$ h_{jkr} quelconque. C'est le cas le plus général mais le plus difficile à gérer.

où R_{jk} est l'ensemble des types de ressources nécessaires pour l'opération (j,k), h_{jkr} est le nombre de ressources du type r nécessaires pour cette opération et p_{jk} est son temps opératoire. Il est bien évident qu'un système SRU ne possède pas d'opération SRMU et MRMU.

En tenant compte de la succession des opérations, nous introduisons les opérations suivantes :

G-opération (j,k) (G = *get*) avec $\mathbf{h}_{jk-1} \leq \mathbf{h}_{jk}$

R-opération (j,k) (R = *release*) avec $\mathbf{h}_{jk} \geq \mathbf{h}_{jk+1}$.

De nouvelles ressources sont nécessaires pour commencer une G-opération. La fin d'une R-opération libère une partie des ressources. Mais dans les deux types de succession ci-dessus, aucun travail ne retient une ressource dont il n'a plus besoin. Nous notons qu'une opération (j,k) peut être à la fois une G-opération et une R-opération. Par convention, la première opération (j,1) de chaque travail est une G-opération et la dernière opération (j,N_j) est une R-opération.

Les G/R-opérations nous permettent de définir une forme canonique des job-shops MRB.

Un job-shop MRB est dit **système canonique** si pour chaque opération (j,k), (j, k-1) est une R-opération si (j,k) n'est pas une G-opération.

Un système canonique est dit **système simple** si les temps opératoires sont tous positifs, c'est-à-dire $p_{jk} > 0$, $\forall$ (j,k).

Nous remarquons que dans un système canonique ou simple, la succession entre deux opérations consécutives du même travail est caractérisée soit par l'acquisition de nouvelles ressources, soit par la libération de ressources, mais jamais les deux à la fois, c'est-à-dire aucun travail n'attend de nouvelles ressources pour libérer des ressources qu'il retient. Nous soulignons que dans un système canonique ou simple, si (j,k) n'est pas une G-opération, alors (j, k-1) est une R-opération et elle peut également être une G-opération.

Afin d'illustrer la différence entre un job-shop MRB et un système canonique, nous considérons deux exemples :

- le job-shop MRB1 composé de deux travaux :

  $J_1 = \{(r_1\ r_2, 5), (r_2, 3), (r_1\ r_2, 4)\}$
  $J_2 = \{(r_1, 4), (r_2, 5)\}$

- le job-shop MRB2 composé de deux travaux :

  $J_3 = \{(r_0\ r_1, 5), (r_0, 0), (r_0\ r_2, 4)\}$
  $J_4 = \{(r_0\ r_2, 4), (r_0, 0), (r_0\ r_1, 3)\}$

où $h_{jkr} = 1$, $\forall\ r \in R_{jk}$. Le job-shop MRB1 n'est pas un système canonique puisque, dans J_2, $(r_2, 5)$ est une R-opération mais $(r_1, 4)$ n'est pas une R-opération (elle est seulement une G-opération). En effet, le passage de l'opération $(r_1, 4)$ à l'opération $(r_2, 5)$ dans J_2 est caractérisé par l'acquisition de r_2 avant la libération de r_1. Par contre, le job-shop MRB2 est un système canonique car, dans J_3, $(r_0\ r_1, 5)$ est une G/R-opération, $(r_0, 0)$ n'est ni G-opération ni R-opération, $(r_0\ r_2, 4)$ est une G/R-opération, et dans J_4, $(r_0\ r_2, 4)$ est une G/R-opération, $(r_0, 0)$ n'est ni G-opération ni R-opération, $(r_0\ r_1, 3)$ est une G/R-opération.

Nous montrerons que, dans un système simple, les contraintes de capacité et de précédence impliquent l'absence de blocage. Pour cette raison, les systèmes simples jouent un rôle très important dans la construction d'ordonnancement sans blocage. Ceci est encore consolidé par le Théorème 2.1 suivant.

Pour tout job-shop MRB composé d'un ensemble de travaux $\{J_1, J_2, ..., J_j, ..., J_n\}$, nous appelons sa forme canonique un job-shop MRB composé du même ensemble de ressource et de travaux $\{ J_1^c, J_2^c, ..., J_n^c \}$ tels que:

$$J_j^c = \{(R_{j1}^c, h_{j1}^c, p_{j1}^c), (R_{j2}^c, h_{j2}^c, p_{j2}^c), ..., (R_{jN_j^c}^c, h_{jN_j^c}^c, p_{jN_j^c}^c)\}$$

où

$$N_j^c = 2\,N_j - 1$$

$$(R_{j,2k-1}^c, h_{j,2k-1}^c, p_{j,2k-1}^c) = (R_{jk}, h_{jk}, p_{jk})\ ,\ \forall 1 \le k \le N_j$$

$$R_{j,2k}^c = R_{jk} \cup R_{jk+1}\ ,\ \forall 1 \le k \le N_j$$

$$h_{j,2k,r}^c = \max\ \{h_{jkr}, h_{j,k+1,r}\}\ ,\ \forall 1 \le k \le N_j\ ,\ \forall r \in R$$

$$p_{j,2k}^c = 0\ ,\ \forall 1 \le k \le N_j\ .$$

Pour résumer, la forme canonique est obtenue en ajoutant, entre deux opérations consécutives (j,k) et (j,k+1), une opération ZD nécessitant l'ensemble des ressources des opérations (j,k) et (j,k+1).

Théorème 2.1: Pour tout ordonnancement $\{S_{jk},\pi_r\}$ d'un job-shop MRB, soit $\{S^c_{jk}, \pi^c_r\}$ un ordonnancement de sa forme canonique telle que $S^c_{j,2k-1} = S_{jk}$, $S^c_{j,2k} = S_{j,k+1}$ et π^c_r est telle que $(j,(2k-1)) > (j,2k-1) > (j,2k) > (j',2(k'-1)) > (j',2k'-1) > (j',2k')$ dans π^c_r avec $(j,k) = \pi_r[i]$ et $(j',k') = \pi_r[i+1]$. Alors $\{S_{jk}, \pi_r\}$ est réalisable si, et seulement si, $\{S^c_{jk}, \pi^c_r\}$ est réalisable.

La preuve est triviale puisque dans un job-shop MRB, la propriété "Retenir et Attendre" implique que, lors du passage de l'opération (j,k) à (j,k+1), l'ensemble des ressources nécessaires à ces deux opérations sont disponibles. Cette phase de transition, où l'ensemble des ressources nécessaires aux opérations (j,k) et (j,k+1) sont allouées au travail J_j, correspond exactement à l'opération (j,2k) de la forme canonique. L'ordonnancement $\{S^c_{jk}, \pi^c_r\}$ respecte l'ordonnancement $\{S_{jk},\pi_r\}$ et la propriété "Retenir et Attendre".

II.4 Modélisation des systèmes industriels

Dans ce paragraphe, nous montrons que les job-shops MRB fournissent un cadre général pour étudier les problèmes d'ordonnancement avec une grande variété de ressources et de contraintes industrielles. Pour ce faire, nous modélisons, par des job-shops MRB, différents modèles de la littérature :

- Beaucoup de caractéristiques de travaux habituellement décrites littéralement peuvent être représentées de manière explicite à l'aide des opérations ZD ou ZR. Par exemple,

  - Date de disponibilité d'un travail r_j :

    $$\{(\phi, \mathbf{r_j}), (R_{j1}, p_{j1}), ...\}$$

  - Durée de latence d'un travail l_j :

    $$\{(R_{j1}, p_{j1}), ..., (R_{jNj}, p_{jNj}), (\phi, \mathbf{l_j})\}$$

  - Délai de communication ou de transport sans contrainte de moyen de transport d_j :

$$\{(R_{j1}, p_{j1}), (\phi, \mathbf{d}_j), (R_{j2}, p_{j2})\}.$$

- Les job-shops avec un ensemble de machines où la gamme de production de chaque travail est la suivante :

$$\{(M_{j1}, p_{j1}), (M_{j2}, p_{j2}), ..., (M_{jNj}, p_{jNj})\}$$

Dans un job-shop, les seules ressources sont les machines et on suppose implicitement que les capacités des stocks tampons sont illimitées. Il suffit alors d'insérer une opération ZRZD entre deux opérations consécutives. La gamme de chaque travail, dans le job-shop MRB équivalent, est la suivante (les vecteurs $\mathbf{h}_{jk}$ ne sont pas donnés car $h_{jkr} = 1, \forall r \in R_{jk}$) :

$$\{(M_{j1}, p_{j1}), (\phi,0), (M_{j2}, p_{j2}), (\phi,0), ...\}$$

où les opérations ZRZD sont ajoutées après chaque opération du job-shop pour représenter la succession des opérations.

- Les systèmes flexibles de production (ou job-shops avec des AGV comme moyens de transport) où la gamme de production est la suivante:

$$\{(AGV, M_{j1}), (AGV, M_{j2}), ..., (AGV, M_{jNj})\}$$

Dans ce cas, l'ensemble des ressources utilisées pour le job-shop est composé d'un ensemble de machines et des AGV. Un AGV est nécessaire tout au long de la réalisation d'un travail. L'AGV est occupé à partir du début de la première opération et n'est libéré qu'à la fin de la dernière opération. Avec le job-shop MRB, leur gamme de production est :

$$\{(AGV,M_{j1},p_{j1}), (AGV,0), (AGV,M_{j2},p_{j2}), (AGV,0), ..., (AGV,M_{jNj},p_{jNj})\}$$

- Les job-shops sans en-cours (ou job-shops *with blocking*) sont des job-shops sans stocks tampons intermédiaires (voir Hall et Sriskandarajah, 96), où chaque travail peut être représenté comme suit :

$$\{(M_{j1}, p_{j1}), (M_{j2}, p_{j2}), ..., (M_{jNj}, p_{jNj})\}$$

dans le cadre d'un job-shop MRB.

- Les flow-shops hybrides sont similaires aux flow-shops mais avec des machines identiques (Vignier et al., 97). La représentation des travaux est

similaire à celle des job-shops; par contre, le nombre de ressources $H_r > 1$ pour au moins un type de machine r.

- Les lignes de production sont composées des machines séparées par des stocks tampons de capacité limitée (Dallery et al., 90, Dallery et Gershwin, 92). Dans une ligne de production, les machines sont agencées dans un ordre donné et des stocks tampons séparent les machines et régulent la fluctuation de la production. Soient M_1, M_2, ..., M_n les machines et S_1, S_2, ..., S_{n-1} les stocks tampons de capacité C_1, C_2, ..., C_{n-1} respectivement. Le processus de fabrication d'un produit en terme de job-shop MRB est :

$$\{(M_1 , p_{j1}), (S_1,0), (M_2 , p_{j2}), (S_2,0), ..., (M_n , p_{jn})\}$$

avec $H_{M_j} = 1$ et $H_{S_j} = C_j$. C'est un système à ressources multiples. Un cas particulier est une ligne sans stock tampon dont chaque travail est représenté comme suit:

$$\{(M_1 , p_{j1}), (M_2 , p_{j2}), ..., (M_n , p_{jn})\}$$

- Les cellules robotisées (Sethi et al., 92, Hall et al., 97) sont composées d'un ensemble de machines alimentées par un robot R. Il n'existe pas de zones de stockage. Le robot est nécessaire pour toute opération de manutention. Chaque travail peut être représenté comme suit :

$$\{(R, \Delta), (M_{j1} , p_{j1}), (R, \Delta), (M_{j2} , p_{j2}), (R, \Delta), ...\}$$

où Δ est le temps de charge/décharge du robot.

- Les systèmes multi-processeurs (Drozdowski, 96) ne sont pas des systèmes de production. Un système multi-processeur peut être considéré comme un job-shop mais chaque opération nécessite plusieurs ressources. Le processus d'un travail peut être présenté sous la forme suivante :

$$\{(R_{j1} , p_{j1}), (\phi,0), (R_{j2} , p_{j2}), (\phi,0), ...\}$$

II.5 Etude de la complexité

Dans ce paragraphe, nous présentons une étude de la complexité du problème d'ordonnancement d'un job-shop MRB. Différents cas particuliers, selon le besoin en ressources, sont considérés. Nous donnerons les idées de base pour prouver le niveau

complexité de chacun et les références pour obtenir une preuve plus complète pour les cas étudiés dans la littérature.

II.5.1 Notation utilisée

Nous présentons la notation en deux parties : la première se réfère aux ressources et la deuxième se réfère aux travaux.

Ressources

m	:	le nombre de types de ressources
$R = \{1, 2, ..., r, ..., m\}$	:	l'ensemble des types de ressources
H_r	:	le nombre de ressources du type r
$H = [H_1, H_2, ..., H_r, ..., H_m]^T$	:	le vecteur du nombre de ressources

Travaux

n	:	le nombre de travaux
$J = \{J_1, J_2, ..., J_j, ..., J_n\}$	:	l'ensemble des travaux
O_{jk} ou (j,k)	:	la $k^{ème}$ opération du travail j
$\{O_{j1}, O_{j2}, ..., O_{jk}, ..., O_{jNj}\}$:		la séquence d'opérations du travail j
N_j	:	le nombre d'opérations du travail j
h_{jkr}	:	le nombre de ressources du type r nécessaires pour l'opération (j,k). Evidement, $h_{jkr} \leq H_r$
$\mathbf{h}_{jk} = [h_{jk1}, h_{jk2}, ..., h_{jkr}, ..., h_{jkm}]$	:	le vecteur de besoins en ressources de l'opération (j,k)
$R_{jk} = \{r \in R / h_{jkr} > 0\}$	:	l'ensemble des types de ressources nécessaires pour l'opération (j,k)
p_{jk}	:	le temps opératoire de l'opération (j,k)

S_{jk} : la date de début de l'opération (j,k)

C_{jk} : la date de fin de l'opération (j,k)

II.5.2 Complexité du problème d'ordonnancement

Dans ce paragraphe, nous effectuons une étude détaillée du problème de la minimisation du makespan d'un job-shop MRB. Nous considérons les cas suivants : (i) deux cas triviaux : un seul travail ou une seule opération par travail; (ii) cas avec opération ZR ; (iii) cas des systèmes à ressources unitaires ; (iv) cas des systèmes à ressources multiples.

La table II.1 est une synthèse des résultats de ce paragraphe et les résultats en gras sont nouveaux.

CAS TRIVIAUX	
n=1	trivial
$m = 1/H_1 = 1/N_j = 1$	trivial
CAS AVEC OPERATIONS ZR	
$m = 1/H_1 = 1/N_j \leq 2$	**polynomial**
$m = 1/H_1 = 1/N_j \leq 3$	NP-difficile au sens fort
SYSTEMES A RESSOURCES UNITAIRES ET SANS OP. ZR	
$m = 2/N_j$ = 1/sans op. ZR	**polynomial**
$m \geq 3/N_j = 1/ p_{jk} = 1$:	polynomial
$m=3/N_j = 1$	NP-difficile au sens fort
$p_{jk} = 1/N_j = 1$	NP-difficile au sens fort
$m = 2/ p_{jk}$ = 1/avec op. ZR	NP-difficile au sens fort
$m=2/N_j = 3/O_{j2} = (\phi,0)$	polynomial
$m=2/N_j \leq 2$/sans op. ZR	**NP-difficile au sens fort**
$m=2/N_j \leq 2/\lvert R_{jk} \rvert$ = 1/sans op. ZR	**NP-difficile au sens fort**
SYSTEMES A RESSOURCES MULTIPLES	
$m = 1/N_j = 1/p_{jk} = 1$	NP-difficile au sens fort
$m = 1/N_j = 1/H \in \{2,3\}$	pseudo-polynomial
$m = 1/N_j = 1/H = 5$	NP-difficile au sens fort
$m = 1/N_j = 1/H = 4/h_{jkr} > 2$	**polynomial**
$m = 1/N_j = 1/H = 4/h_{jkr} > 1$	**pseudo-polynomial**
$m = 1/N_j = 1/H = 4/ h_{jkr}$ quelconque	open
m = 1/H = 2/avec op. ZR	NP-difficile au sens fort
$m = 1/H = 2/N_j \leq 2$/sans op. ZR	**NP-difficile au sens fort**

En gras : les résultats nouveaux.

Table II.1 : Synthèse des résultats du paragraphe II.5.2

II.5.2.1 Cas triviaux

Deux cas triviaux, n=1 ou m=1/H=1/N_j=1, sont considérés. Dans le cas n=1, il y a un seul travail et il suffit d'ordonnancer les opérations les unes après les autres, selon la gamme de production. Pour le cas m=1/H=1/N_j=1, chaque travail nécessite une seule opération. Nous n'avons pas besoin de considérer les travaux avec les opérations ZR. Par conséquent, le problème se réduit au problème classique de la minimisation de C_{max} sur une machine, c'est-à-dire 1/ /C_{max}. Tout ordonnancement actif, c'est-à-dire sans temps mortel, est optimal.

II.5.2.2 Cas avec opérations ZR

Théorème 2.2 : La minimisation de C_{max} est polynomiale pour le cas m =1/H_1=1/$N_j \leq 2$.

Preuve:
Dans ce cas, il y a quatre types de travaux comme suit :

- Classe A : $J_i = (\phi,p_j)$. Dans cette classe les travaux ont des opérations du type ZR.

- Classe B : $J_i = (r,p_j)$. Dans cette classe les travaux ont des opérations qui nécessitent la ressource unique r.

- Classe C : $J_i = \{(\phi,r_j), (r,p_j)\}$. Chaque travail J_j de cette classe commence par une opération ZRD suivie d'une opération sur la ressource unique r.

- Classe D : $J_i = (r,p_j)\ (\phi,l_j)$. Chaque travail J_j de cette classe commence par une opération sur la ressource unique r suivie d'une opération ZRD.

Les travaux qui appartiennent à la classe A ne seront pas considérés pour l'ordonnancement car la ressource n'est pas utilisée. Les autres travaux peuvent être considérés comme des tâches sur la ressource unique r avec date de disposition pour le cas C et temps de latence pour le cas D.

L'ordonnancement optimal π^* est obtenu comme suit :

- les travaux de la classe D sont ordonnancés d'abord et dans l'ordre décroissant de l_j ;

- les travaux de la classe B sont ordonnancés ensuite et dans un ordre quelconque ;

- les travaux de la classe C sont alors ordonnancés dans l'ordre croissant de r_j.

Dans la suite, nous montrons l'optimalité de cette solution en utilisant la méthode par permutation de deux travaux (Carlier et Chrétienne, 88). Pour cela, considérons un ordonnancement quelconque $\pi^0 \neq \pi^*$. Il existe alors deux travaux consécutifs i et j sur la ressource unique r qui violent les conditions de π^*. Nous considérons π^N l'ordonnancement dérivé de π^0 en permutant i et j sans modifier les dates de début des autres travaux. Nous montrons, dans la suite, que l'ordonnancement est admissible et $C^N_{max} < C^0_{max}$, ce qui conclut la preuve.

Puisque les makespan s'écrivent comme suit :

$$C^0_{max} = \max \{ \max_{l \in J-\{i,j\}} C^0_l , C^0_i , C^0_j \},$$

$$C^N_{max} = \max \{ \max_{l \in J-\{i,j\}} C^N_l , C^N_i , C^N_j \},$$

il suffit de montrer que π^N est admissible et

$$\max \{ C^0_i , C^0_j \} \geq \max \{ C^N_i , C^N_j \}$$

où C^0_i et C^0_j correspondent aux valeurs de la durée totale du travail i et du travail j avant la permutation entre ces deux travaux i et j, respectivement. Et, C^N_i et C^N_j correspondent aux valeurs de la durée totale de i et j après la permutation.

Nous considérons les cinq combinaisons des deux travaux comme suit :

(1) $T_j \in$ Classe D $\wedge$ $T_i \in$ Classe B

(2) $T_j \in$ Classe D $\wedge$ $T_i \in$ Classe C

(3) $T_j \in$ Classe D $\wedge$ $T_i \in$ Classe D $\wedge$ $l_j > l_i$

(4) $T_j \in$ Classe B $\wedge$ $T_i \in$ Classe C

(5) $T_j \in$ Classe C $\wedge$ $T_i \in$ Classe C $\wedge$ $r_i > r_j$

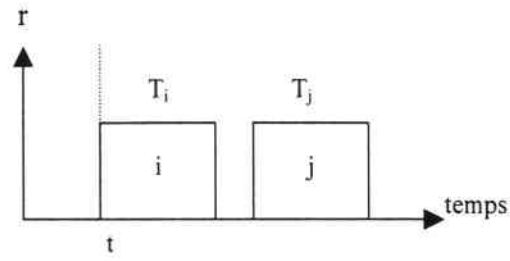

Figure II.2 : Ordonnancement initial des deux travaux i et j

Dans la suite, nous analysons chaque possibilité de permutation entre deux travaux :

(1) Dans la figure II.3, nous avons l'illustration des deux ordonnancements: (a) avant et (b) après la permutation entre les travaux avec $T_i \in$ Classe B et $T_j \in$ Classe D ou $T_i \in B \wedge T_j \in D$. Pour les ordonnancements π^0 et π^N, nous avons :

$C_i^0 = t + p_i$ $\qquad$ $C_j^0 \geq t + p_i + p_j + l_j$

$C_i^N = t + p_i + p_j$ $\qquad$ $C_j^N = t + p_j + l_j$

Il est clair qu'il n'est pas nécessaire de modifier les dates de début des autres travaux, ce qui implique que π^N est admissible. De plus,

$$\max \{ C_i^N, C_j^N \} = \max \{ t + p_j + l_j, t + p_i + p_j \}$$
$$\leq t + p_i + p_j + l_j \leq \max \{ C_i^0, C_j^0 \}.$$

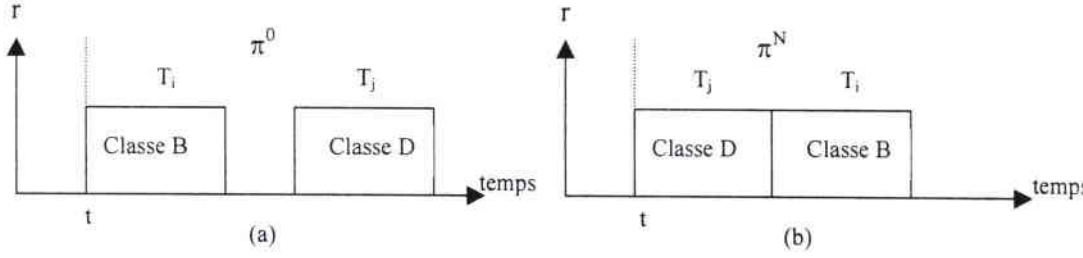

Figure II.3 : Ordonnancement des deux travaux : T_i et T_j (cas 1)

(2) La figure II.4 illustre les deux ordonnancements: (a) avant et (b) après la permutation entre les travaux avec $T_i \in$ Classe C et $T_j \in$ Classe D ou $T_i \in C \wedge T_j \in D$.

La preuve de ce cas est analogue à la preuve du cas (1) puisque T_i est déjà disponible à l'instant t.

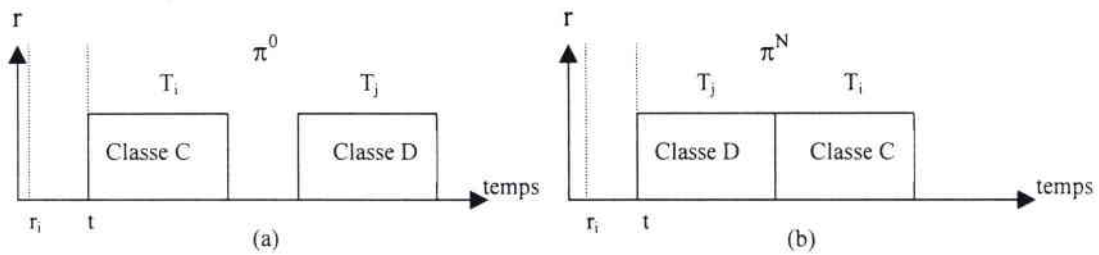

Figure II.4 : Ordonnancement des deux travaux : T_i et T_j (cas 2)

(3) Dans cette combinaison, $T_i \in$ Classe D et $T_j \in$ Classe D avec $l_j > l_i$. Pour les ordonnancements π^0 et π^N, nous avons :

$$C_i^0 = t + p_i + l_i \qquad C_j^0 \geq t + p_i + p_j + l_j$$

$$C_i^N = t + p_i + p_j + l_i \qquad C_j^N = t + p_j + l_j$$

Il est clair qu'il n'est pas nécessaire de modifier les dates de début des autres travaux, ce qui implique que π^N est admissible. De plus,

$$\max \{ C_i^N, C_j^N \} = \max \{ t + p_i + p_j + l_i, t + p_j + l_j \}$$
$$\leq t + p_i + p_j + l_j \leq \max \{ C_i^0, C_j^0 \}.$$

(4) Dans cette combinaison, $T_i \in$ Classe C et $T_j \in$ Classe B. La preuve est similaire aux cas précédents.

(5) Dans cette combinaison, $T_i \in$ Classe C et $T_j \in$ Classe C avec $r_i > r_j$. La preuve est similaire aux cas précédents puisque l'ordonnancement π^N est admissible selon la figure II.5. *Q.E.D.*

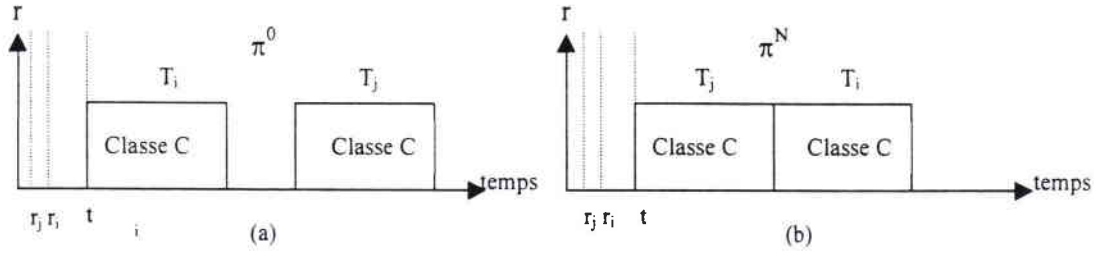

Figure II.5 : Ordonnancement des deux travaux : T_i et T_j (cas 5)

Théorème 2.3 : La minimisation de C_{max} est NP-difficile au sens fort, dans le cas $m=1/H_1=1/N_j \leq 3$

Ce cas est NP-difficile au sens fort parce qu'il inclut comme cas particulier le problème NP-difficile au sens fort $1/r_j,l_j/C_{max}$ (Garey et Johnson, 78) où chaque tâche peut être représentée comme (ϕ,r_j) (M,p_j) (ϕ,l_j). Nous donnons la preuve pour la complétude.

Preuve :

La preuve est faite par réduction du problème NP-difficile au sens fort de 3-Partition. Pour l'étude de la complexité, on présente souvent les problèmes sous la forme de "instance/question". Dans la suite, nous présentons le problème avec ce type de présentation :

Problème 3-Partition

Instance : Un entier b et un ensemble $N = \{a_1, ..., a_{3n}\}$ de 3n entiers positifs tels que $b/4 < a_j < b/2$ et $\sum_{j=1}^{3n} a_j = nb$.

Question : Existe-t-il une partition de N, N_1, ..., N_n, telle que $\sum_{j \in N_i} a_j = b$, $\forall i = 1, ..., n$?

Pour chaque instance du problème 3-Partition, nous associons une instance du problème $1/r_j,l_j/C_{max}$ de 4n tâches :

$J = \{t_1, t_2, ..., t_n\} \cup N$

$$p_j = \begin{cases} a_j, \forall j \in N \\ 1, \quad \forall j = t_i \end{cases}$$

$$r_j = \begin{cases} 0, & \forall j \in N \\ ib+i-1, & \forall j = t_i \end{cases}$$

$$l_j = \begin{cases} 0, & \forall j \in N \\ (n-i)(b+1), & \forall j = t_i \end{cases}$$

Nous pouvons maintenant poser une autre question comme suit :

Question : Existe-t-il un ordonnancement tel que $C_{max} = n(b+1)$?

Considérons un ordonnancement tel que $C_{max} = n(b+1)$. Puisque que $C_j \geq r_j + p_j + l_j \geq n(b+1)$, $\forall j = t_i$, alors nous avons la date de début $S_j = r_j$ pour tout $j = t_i$. De la figure II.6, on constate que le problème d'ordonnancement a une solution telle que $C_{max} = n(b+1)$ si, et seulement si, le problème 3-Partition a une solution. Par conséquent, le problème $1/r_j,l_j/C_{max}$ est NP-difficile au sens fort.

Q.E.D.

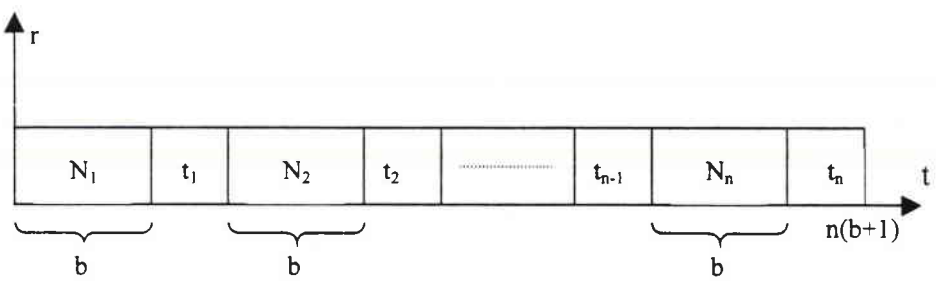

Figure II.6 : Un ordonnancement du problème $1/r_j,l_j/C_{max}$

II.5.2.3 Systèmes à Ressources Unitaires et sans opérations ZR

Dans ce paragraphe, nous considérons les systèmes à ressources unitaires, c'est-à-dire $H_r = 1$, $\forall r \in R$. Nous nous intéressons essentiellement au cas sans opérations ZR.

Considérons d'abord le cas m = 1. Comme le cas $m=1/H=1/N_j=1$, nous avons aussi une seule ressource. La possibilité pour un travail d'avoir un nombre quelconque d'opérations rend le cas m = 1 différent du cas $m=1/H=1/N_j=1$. Mais la solution reste triviale à cause de la propriété "Retenir et Attendre". Il suffit, parce que nous avons une seule ressource, d'ordonnancer les opérations dans la gamme de production de chaque travail et les travaux dans un ordre quelconque.

Théorème 2.4 : La minimisation de C_{max} est polynomiale pour le cas m=2/N_j = 1/sans opérations ZR.

Preuve:

Le système possède deux ressources différentes $\{r_1, r_2\}$ et chaque travail nécessite une opération. Il y a trois types de travaux : les travaux bi-ressources $A = \{(r_1\ r_2, p_j)\}$ ainsi que les travaux mono-ressources $B = \{(r_1, p_j)\}$ et $C = \{(r_2, p_j)\}$. L'ordonnancement, qui consiste à exécuter d'abord les travaux bi-ressources et ensuite les travaux mono-ressources, est optimal car les travaux bi-ressources terminent à l'instant $\sum_{j\in A} p_j$, les travaux mono-ressources sur r_1 terminent à l'instant $\sum_{j\in A\cup B} p_j$ et les travaux mono-ressources sur r_2 terminent à l'instant $\sum_{j\in A\cup C} p_j$. Par conséquent, l'ordonnancement est optimal.

Q.E.D.

Considérons ensuite le cas $m \geq 3/N_j = 1$.

Théorème 2.5 (Hoogeveen et al., 94) : La minimisation de C_{max} est polynomiale pour le cas $m \geq 3/N_j = 1/p_{jk} = 1$.

Preuve:

Puisque $p_{jk} = 1$ et $N_j = 1$, on peut regrouper les travaux selon leur besoin en ressource h_{jkr}, $\forall r \in R$. Il y a donc, au plus, $2^M - 1$ types de travaux. Soient M le nombre de types de travaux et b_j le nombre de travaux du type j, $\forall\ 1 \leq j \leq M$. Dans chaque période, une combinaison de travaux est exécutée. Puisque $H_r = 1$, chaque combinaison c contient, au plus, un travail d'un type donné. Soit $a_{cj} \in \{0,1\}$ le nombre de travaux du type j dans c. Par conséquent, il y a, au plus, $2^M - 1$ combinaisons différentes. Soit k le nombre de combinaisons. Le problème de la minimisation de C_{max} consiste à déterminer le nombre de périodes x_c de chaque combinaison c pour

$$\text{Min} \sum_{c=1}^{k} x_c$$

sous la contrainte :

$$\sum_{c=1}^{k} a_{cj}\, x_j = b_j,\ \forall\ 1 \leq j \leq M$$

Il a été démontré, dans Lenstra (83), qu'un programme linéaire en nombres entiers avec un nombre donné de variables est polynomial. Pour m donné, le nombre de variables k est donné et le problème de la minimisation de C_{max} est donc polynomial.

Q.E.D.

Théorème 2.6 (Hoogeveen et al., 94) : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $m = 3/N_j = 1$.

Preuve:

Le théorème 2.3 de Hoogeveen et al. (94), montre la NP-difficulté au sens fort en utilisant une réduction du problème NP-difficile de 3-Partition.

Q.E.D.

Si le nombre de ressources m fait partie de l'énoncé du problème (ou instance), il a été démontré dans Hoogeveen et al. (94), théorème 2.5, que décider l'existence d'un ordonnancement de longueur d'au plus 3 est NP-difficile au sens fort. La preuve est fondée sur une transformation du problème NP-difficile au sens fort de la 3-Colorabilité d'un graphe (Garey et Johnson, 79). Par conséquent :

Théorème 2.7 (Hoogeveen et al., 94) : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $p_{jk}=1/N_j = 1$.

Considérons maintenant le cas m=2 où il existe deux types de ressources.

Théorème 2.8 (Brucker et Kramer, 95) : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $m=2/p_{jk}=1/$ avec opérations ZR.

En effet, ce cas contient le problème d'ordonnancement de tâches multiprocesseurs suivant comme cas particulier :

$$J_j = \{(R_{j1},1), (\phi,0), (R_{j2},1), (\phi,0), ..., (R_{jNj},1)\}$$

avec $R_{jk} \subseteq R=\{r_1, r_2\}$. Il a été démontré dans Brucker et Kramer (95), théorème 6, que le problème est NP-difficile au sens fort. La preuve s'établit en deux étapes : (1) suppression des tâches bi-ressources pour obtenir un problème de tâches mono-ressources avec contraintes de précédence, (2) utiliser le théorème 2.7 de Hoogeveen et al. (94), pour prouver la NP-difficulté.

Théorème 2.9 (Brucker et Kramer, 95) : La minimisation de C_{max} est polynomiale pour le cas $m=2/N_j = 3/O_{j2} = (\phi,0)$.

Dans ce cas, nous avons un problème classique d'ordonnancement de tâches multi-processeurs. Selon le théorème 7.a de Brucker et Kramer (95), la minimisation de C_{max} est polynomiale. L'ordonnancement est calculé comme suit : d'abord si la première (respectivement la dernière) opération d'un travail est une tâche bi-processeur, elle peut être déplacée au début (respectivement à la fin) de l'ordonnancement sans modifier le C_{max} ; l'ordonnancement des autres opérations nécessitant une seule ressource est alors équivalent au problème d'ordonnancement de job-shop à 2-machines $J2/n_i \leq 2/C_{max}$. Ce dernier peut être résolu en temps O(n log n) par l'algorithme de Jackson (Carlier et Chrétienne, 88).

Théorème 2.10 : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $m=2/N_j \leq 2$/sans opérations ZR.

Preuve:

La preuve de la NP-difficulté au sens fort est obtenue par réduction du problème 3-Partition. Pour chaque instance du problème 3-Partition, nous associons le problème d'ordonnancement suivant :

$$J = N \cup \{t_1, t_2, ..., t_n\}$$
$$J_j = \{(r_1, a_j)\}, \forall j \in N$$
$$J_j = \{(r_1 r_2, 1), (r_2, b)\}, \forall j = t_i$$

Question: Existe-t-il un ordonnancement tel que $C_{max} = n(b+1)$?

Considérons un ordonnancement tel que $C_{max} = n(b+1)$. Il est bien évident que $C_{max} = n(b+1)$ implique que les travaux $t_1, t_2, ..., t_n$ sont exécutés sans interruption, à partir du début de l'ordonnancement. Ceci laisse n intervalles de disponibilité sur r_1 de longueur b chacun pour ordonnancer les travaux $j \in N$. Ce qui implique que le problème de 3-Partition a une solution si, et seulement si, le problème d'ordonnancement possède un ordonnancement avec $C_{max} = n(b+1)$. Nous illustrons cet ordonnancement avec $C_{max} = n(b+1)$ dans la figure II.7.

Q.E.D.

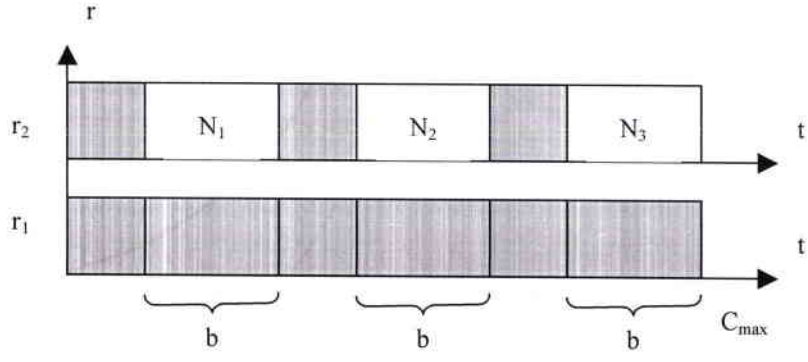

Figure II.7 : Ordonnancement d'un SRU

La NP-difficulté persiste même si chaque opération demande une seule ressource.

Théorème 2.11 : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $m=2/N_j \leq 2$ / $|R_{jk}| = 1$/sans opérations ZR.

Preuve:

Contrairement au cas précédent, ici chaque opération nécessite une ressource. La preuve de la NP-difficulté est établie par réduction du problème de 3-Partition. A chaque instance du problème 3-Partition, est associée l'instance suivante du problème d'ordonnancement :

$$J = N \cup \{t_1, t_2, ..., t_n, t_{n+1}, t_{n+2}\}$$
$$J_j = \{(r_1, a_j)\}, \forall j \in N$$
$$J_j = \{(r_1, 1), (r_2, b+1)\}, \forall j \in \{t_1, t_2, ..., t_n, t_{n+1}\}$$
$$J_j = (r_1, b+1), \text{ avec } j = t_{n+2}$$

Question: Existe-t-il un ordonnancement tel que $C_{max} = (n+1)(b+1)+1$?

Considérons un ordonnancement avec $C_{max} = (n+1)(b+1)+1$. Compte tenu des contraintes de précédence, un travail t_j avec $1 \leq j \leq n+1$ doit commencer sa première opération sur r_1 à l'instant 0 ; puis, les opérations sur r_2 des travaux $t_1, t_2, ..., t_{n+1}$ doivent être exécutées sans interruption. Puisque chaque travail t_j avec $1 \leq j \leq n+1$ ne peut libérer la ressource r_1 qu'au début de sa deuxième opération, les opérations sur r_1 de ces travaux sont réparties dans le temps, comme dans la figure II.9, afin d'éviter d'occuper la ressource r_1, comme dans la figure II.8.

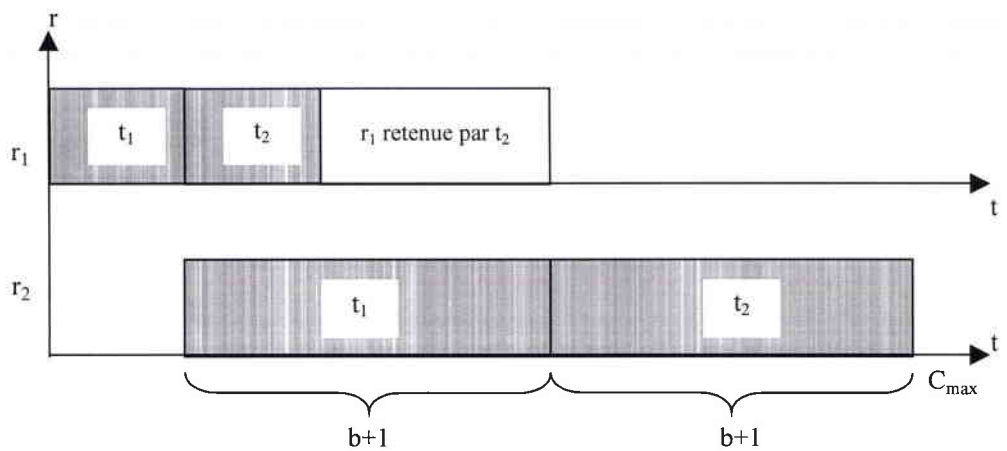

Figure II.8 : Ordonnancement du cas $m=2/N_j \leq 2/|R_{jk}| = 1$

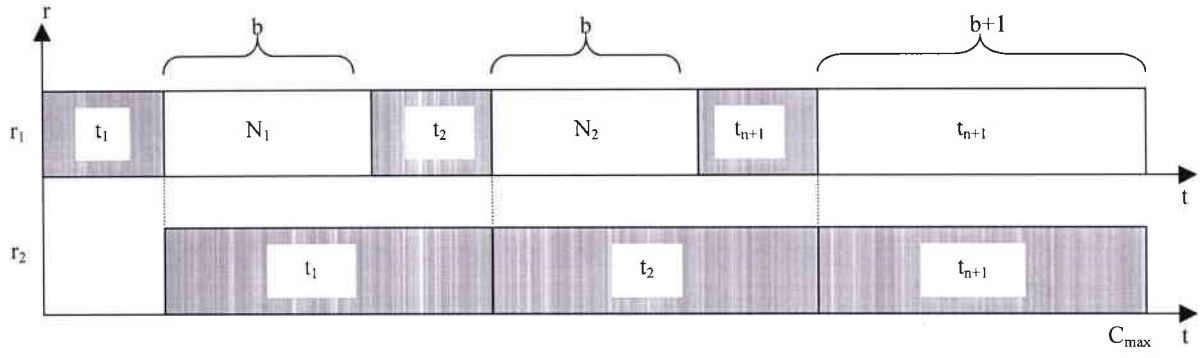

Figure II.9 : Ordonnancement du cas m=2/$N_j \leq 2$/ | R_{jk} | = 1 avec la répartition des opérations sur r_1

Il reste, sur la ressource r_1, n intervalles de longueur b chacun et un intervalle de longueur b+1. Le dernier intervalle est consacré au travail t_{n+2} afin d'avoir C_{max} = (n+1)(b+1)+1. Il est alors évident que le problème d'ordonnancement a une solution si, et seulement si, le problème 3-Partition a une solution.

Q.E.D.

II.5.2.4 Systèmes à Ressources Multiples

Nous considérons, dans ce paragraphe, les systèmes à ressources multiples, c'est-à-dire $\exists r \in R / H_r > 1$.

D'abord, le cas d'un seul type de ressource et d'une seule opération par travail, c'est-à-dire m=1/H > 1/ N_j = 1, a été largement étudié dans Du et Leung (89), dans le cadre de l'ordonnancement des tâches multi-processeurs. Nous donnons, ci-dessous, un résumé des résultats principaux.

Théorème 2.12 (Du et Leung, 89) : La minimisation de C_{max} est NP-difficile au sens fort pour le cas m=1/N_j=1 /p_{jk} = 1.

Ce cas est équivalent au problème de Bin-Packing si on associe les périodes de temps aux conteneurs, H à la capacité de chaque conteneur, les travaux aux objets à remplir. Il est alors clair que le problème est NP-difficile au sens fort (Garey et Johnson, 78).

Théorème 2.13 (Du et Leung, 89) : La minimisation de C_{max} est pseudo-polynomiale pour le cas m=1/N_j=1 /H $\in$ {2,3}.

Il a été démontré, dans Du et Leung (89), que tout ordonnancement se transforme en forme canonique, illustrée par la figure II.10 pour le cas H = 2 et par la figure II.11 pour le cas H = 3.

Il est clair qu'un ordonnancement canonique est entièrement caractérisé par les nombres y, x_1, x_2, ..., x_5. La Programmation Dynamique est alors utilisée pour résoudre les problèmes F(i,x) avec x = (x_1,x_2) pour le cas H = 2 et x = (x_1,x_2, ...,x_5) pour le cas H = 3. F(i,x) consiste à trouver un ordonnancement admissible de caractéristiques x pour les i premiers travaux.

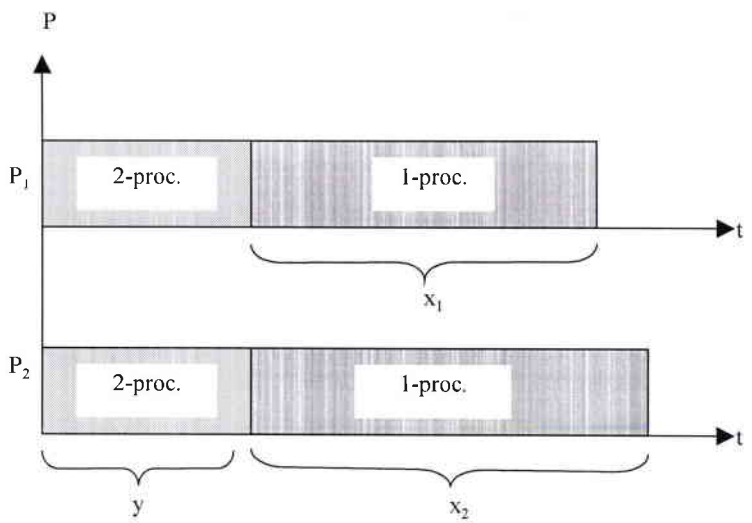

Figure II.10 : Ordonnancement canonique pour H = 2

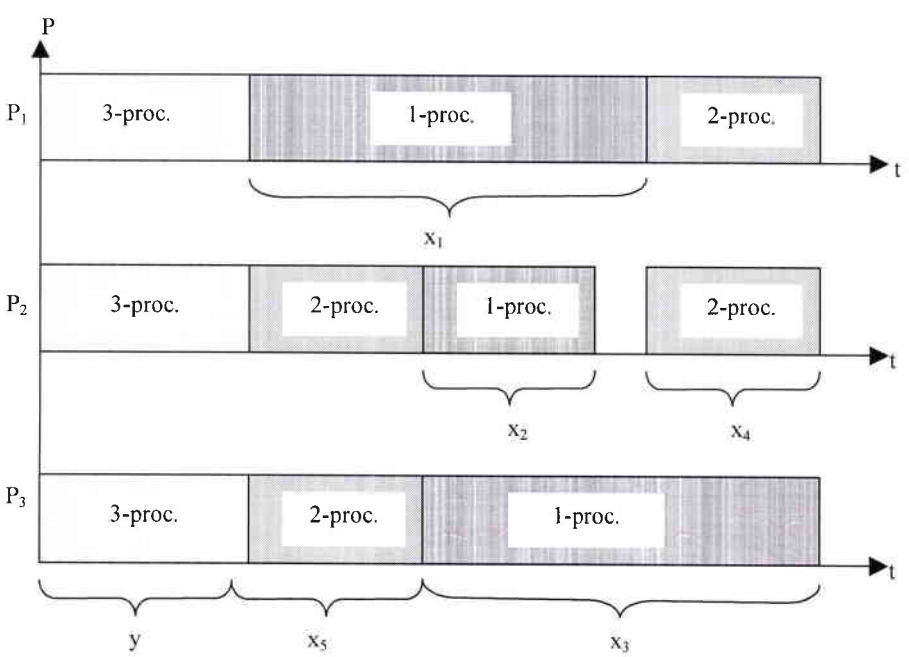

Figure II.11 : Ordonnancement canonique pour H = 3

La complexité du cas $m=1/N_j=1/H=4$ est, à ce jour, inconnue.

Théorème 2.14 (Du et Leung, 89) : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $m=1/N_j=1$ /H= 5.

La NP-difficulté de ce cas a été prouvée dans Du et Leung (89), théorème 2, en utilisant une transformation du problème 3-Partition et les formes canoniques présentées ci-dessus.

Théorème 2.15 : Pour le cas $m=1/N_j=1/H=4$, la minimisation de C_{max} est polynomiale si $h_{jkr}>2$ et pseudo-polynomiale si $h_{jkr}>1$.

Preuve:

Si $h_{jkr} > 2$, alors chaque opération nécessite 3 ou 4 ressources. Il est alors impossible d'exécuter deux opérations en parallèle. Tout ordonnancement est donc optimal. Le problème est donc polynomial.

Si $h_{jkr} > 1$, les opérations nécessitant 3 ou 4 ressources ne peuvent être exécutées en parallèle avec une autre opération. Par contre, les opérations nécessitant 2 ressources peuvent être exécutées en parallèle. Le problème se transforme alors en un problème de la minimisation de C_{max} sur deux machines identiques, c'est-à-dire P2/ /C_{max}. Ce problème est pseudo-polynomial, ce qui prouve que le cas $h_{jkr} > 1$ est pseudo-polynomial.

Q.E.D.

Si h_{jkr} est quelconque, le problème reste ouvert.

Si chaque travail peut avoir besoin de plusieurs opérations, Du et Leung (89) ont prouvé la NP-difficulté pour le cas avec opérations ZR.

Théorème 2.16 (Du et Leung, 89) : La minimisation de C_{max} est NP-difficile au sens fort pour le cas m=1/H=2 /avec opérations ZR.

Ce cas inclut, comme cas particulier, le problème d'ordonnancement de travaux avec tâches multiprocesseurs. Il y a une opération ZRZD entre deux opérations du même travail. Le nouveau problème est NP-difficile selon le théorème 1 de Du et Leung (89). La preuve est fondée sur une transformation du problème 3-Partition en une instance du problème d'ordonnancement :

$$T = N \cup \{t_1, t_2, ..., t_n\} \cup \{s_1, s_2, ..., s_{n-1}\}$$
$$J_j = (r_1, a_j), \forall j \in N$$
$$J_j = (r, b), \quad \forall j = t_1, t_2, ..., t_n$$

$$J_j = (2r, 1), \forall j = s_1, s_2, ..., s_{n-1}$$

dont les tâches $t_1, t_2, ..., t_n$ et $s_1, s_2, ..., s_{n-1}$ sont sujets à la contrainte de précédence $t_1 < s_1 < t_2 < s_2 < ... t_{n-1} < s_{n-1} < t_n$. Il n'y a pas d'autres contraintes de précédence.

Question: Existe-t-il un ordonnancement tel que $C_{max} = nb + n - 1$?

Pour avoir un tel ordonnancement, les tâches t_i et s_i doivent être exécutées sans interruption à partir de l'instant 0. Ceci laisse n intervalles de longueur b chacun avec une seule unité de ressource disponible pour les tâches N. Par conséquent, le problème d'ordonnancement a une solution si, et seulement si, le problème 3-Partition a une solution.

Pour le cas sans opérations ZR, nous proposons le résultat suivant :

Théorème 2.17 : La minimisation de C_{max} est NP-difficile au sens fort pour le cas $m=1/H=2/N_j \leq 2$.

Preuve:

La preuve se fait par réduction du problème 3-Partition. A chaque instance du problème 3-Partition, nous associons l'instance suivante du problème d'ordonnancement :

$$J = N \cup \{1, 2, ..., n\}$$
$$J_j = (r, a_j), \forall j \in N$$
$$J_j = \{(2r, 1), (r, b)\}, \forall j = 1, 2, ..., n$$

Question: Existe-t-il un ordonnancement tel que $C_{max} = n(b+1)$?

Afin d'obtenir un tel ordonnancement, les travaux 1, 2, ..., n doivent être réalisés sans interruption à partir du début de l'ordonnancement. Ceci laisse n intervalles de longueur b chacun et avec une unité de ressource disponible. Par conséquent, le problème d'ordonnancement a une solution si, et seulement si, le problème 3-Partition a une solution. Nous illustrons ce cas dans la figure II.12.

Q.E.D.

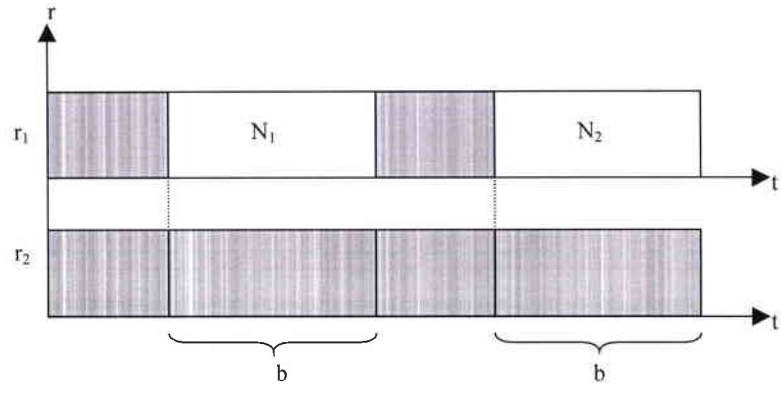

Figure II.12 : Ordonnancement du cas $m=1/H=2/N_j \leq 2$

II.6 Conclusion

Dans ce chapitre, nous avons introduit les job-shops Multi-Ressources avec Blocage qui permettent la prise en compte de la variété de ressources impliquées dans une production. Le problème d'ordonnancement, dont le critère est la minimisation du makespan, est formulé. Sa complexité est étudiée de manière détaillée.

Au vu des résultats sur la complexité du problème d'ordonnancement des job-shops MRB, on constate que le problème devient très vite extrêmement difficile car il est déjà NP-difficile à partir de deux ressources et de deux opérations par travail, dans le cas d'un SRU ou d'un SRM. C'est la raison pour laquelle, nous consacrons la suite de cette thèse au développement des méthodes approchées.

Chapitre III

Ordonnancement des Systèmes à Ressources Unitaires

III.1 Introduction

Dans ce chapitre, nous nous limitons aux Systèmes à Ressources Unitaires, c'est-à-dire les job-shops MRB, dans lesquels les ressources sont toutes différentes. Comme nous l'avons montré dans le paragraphe II.4, les Systèmes à Ressources Unitaires avec opérations ZRZD incluent, comme cas particuliers, les job-shops classiques, les Systèmes à Ressources Unitaires sans opérations ZRZD incluent comme cas particuliers les job-shops sans en-cours ou job-shops *with blocking* et les cellules robotisées et, les job-shops MRB avec opérations ZD incluent comme cas particuliers les systèmes flexibles avec des AGV dédiés. Il est bien évident que les SRU ne permettent pas la prise en compte des capacités de stockage ni des AGV en nombre quelconque. Néanmoins, ils représentent une classe importante des job-shops MRB.

Ce chapitre est organisé de la manière suivante : du paragraphe III.2 au paragraphe III.4, nous nous limitons à l'ordonnancement des opérations avec délai, c'est-à-dire $p_{jk} > 0$; le paragraphe III.2 présente une formulation simplifiée du problème d'ordonnancement pour les SRU ; nous montrons surtout que les contraintes de précédence et les contraintes de capacité de ressources impliquent l'absence de blocage dans le cas d'un système simple ; ce résultat simplifie l'ordonnancement des systèmes simples et sert de point de départ pour les méthodes proposées dans le paragraphe III.4. Dans le paragraphe III.3, nous nous intéressons à la modélisation des SRU à l'aide des RdP, à la représentation de l'ordonnancement et à la vérification de blocage pour un ordonnancement donné ou pour un état du système donné. Le paragraphe III.4 propose plusieurs heuristiques pour l'ordonnancement des SRU généraux ; les RdP seront utilisés pour la vérification de blocage. Dans ce même paragraphe, nous proposons des méthodes heuristiques pour l'ordonnancement des SRU généraux en passant par les systèmes simples. Le paragraphe III.5 est consacré aux extensions ; deux extensions sont considérées : (i) la prise en compte des opérations ZD et (ii) l'optimisation des critères réguliers autres que le makespan. Le paragraphe III.6 présente des résultats numériques et le paragraphe III.7 présente une conclusion.

III.2 Reformulation du problème d'ordonnancement des opérations avec délai

Les opérations zéro délai compliquent de manière significative l'ordonnancement des job-shops MRB et leur prise en compte est délicate. C'est pourquoi, nous nous limitons aux opérations avec délai dans les paragraphes III.2-III.4. La relaxation de cette contrainte sera considérée dans le paragraphe III.5.

Hypothèse 3.1 : Les temps opératoires sont positifs, c'est-à-dire $p_{jk} > 0$.

Le problème d'ordonnancement consiste à choisir les dates de début S_{jk} des opérations (j,k) et une séquence π_r d'entrée des opérations nécessitant la ressource r afin de minimiser la durée totale de l'ordonnancement C_{max} et d'éviter les blocages. Plus précisément,

$$\text{Min } \underset{j}{\text{MAX}} \{C_j\},$$

sous les contraintes suivantes :

$$C_j = S_{jNj} + p_{jNj}\ ,\ \forall 1 \le j \le n \tag{3.1}$$

$$S_{jk} + p_{jk} \le S_{j,k+1},\ \forall 1 \le j \le n,\ \forall 1 \le k \le N_j \tag{3.2}$$

$$\sum_{(j,k)/r \in R_{jk}} \mathbf{1}\{S_{jk} \le t < S_{j,k+1}\} \le 1,\ \forall t\ ,\ \forall r \in R \tag{3.3}$$

$$S_{\pi_r[i]} \le S_{\pi_r[i+1]}\ ,\ \forall r \in R,\ \forall i \tag{3.4}$$

Les séquences d'entrée π_r ne conduisent pas au blocage (3.5)

où les contraintes (3.1) définissent la date de fin de chaque travail, les contraintes (3.2) sont les contraintes de précédence des opérations de chaque travail, les contraintes (3.3) assurent le respect de la capacité de ressources, les contraintes (3.4) assurent la cohérence entre S_{jk} et π_r, et les contraintes (3.5) ne peuvent pas se mettre formellement sous forme mathématique.

Comme nous l'avons montré dans le paragraphe II.2, les contraintes (3.1)-(3.4) n'impliquent pas l'absence de blocage. La vérification de blocage est nécessaire pour tout ordonnancement d'un SRU général satisfaisant les contraintes de précédence et de capacité de ressources. Une exception notable concerne les systèmes simples pour lesquels nous montrons, dans la suite de ce paragraphe que tout ordonnancement vérifiant les contraintes de précédence et de capacité de ressources est sans blocage.

D'abord, dans le cadre d'un SRU, une opération (j,k) est une G-opération (G = *get*) si $R_{j,k-1} \subseteq R_{jk}$ et elle est une R-opération (R = *release*) si $R_{jk} \supseteq R_{jk+1}$. Un SRU est un système simple si $p_{jk} > 0$ et si pour chaque opération (j,k), (j, k-1) est une R-opération si (j,k) n'est pas une G-opération.

Théorème 3.1

Pour un système simple, un ordonnancement $\{S_{jk}\}$ satisfaisant les contraintes de précédence (3.2) et les contraintes de capacité de ressources (3.3) est sans blocage et donc admissible. Les séquences d'entrée π_r, qui respectent les dates de début et qui, pour les opérations (j,k) avec la

même date de début, donnent priorité aux opérations dont (j,k-1) est une R-opération, sont sans blocage.

Preuve par contradiction :

Supposons qu'il existe un instant t tel que toute opération (j,k) / $S_{jk} < t$ peut commencer à la date S_{jk} tout en respectant la propriété "Retenir et Attendre" et qu'il existe une opération (i,l) / $S_{il} = t$ qui ne peut pas démarrer à l'instant t sans violer la propriété "Retenir et Attendre". Deux cas sont possibles :

- Cas (i) : l'opération (i,l-1) est une R-opération, $R_{i,l-1} \supseteq R_{il}$. Puisque les capacités de ressources sont respectées, aucune opération autre que (i,l) n'a besoin des ressources dans R_{il} à l'instant t. Puisque $S_{il-1} < t$, l'opération (i,l-1) peut commencer à la date $S_{i,l-1}$. Par conséquent, toutes les ressources dans R_{il} sont disponibles à l'instant t et l'opération (i,l) peut commencer à la date S_{il}, ce qui contredit l'hypothèse de la preuve.

- Cas (ii) : l'opération (i,l) est une G-opération, $R_{i,l-1} \subseteq R_{il}$. Puisque (i,l) ne peut pas commencer à l'instant t et que les contraintes de capacité de ressources sont vérifiées, il existe une ressource $r \in R_{il}$ et une opération (j,k) où $j \neq i$ telles que $r \in R_{jk}$ et $S_{jk+1} = t$, et que la ressource r ne puisse être libérée au début de la période t (voir figure III.1).

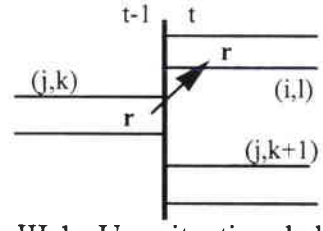

Figure III.1 : Une situation de blocage

Compte tenu de la contrainte de capacité de ressources ($H_r = 1$) et de la propriété "Retenir et Attendre", $r \notin R_{jk+1}$. Par conséquent, l'opération (j,k+1) n'est pas une G-opération, ce qui implique que l'opération (j,k) est une R-opération. Puisque $S_{jk+1} = t$, selon le résultat du cas (i), l'opération (j,k+1) peut commencer à la date t et la ressource r peut alors être libérée au début de la période t. Cela prouve, par l'absurde, que toutes les nouvelles ressources nécessaires pour l'opération (i,l) sont libres ou peuvent être libérées au début de la période t. Alors l'opération (i,l) peut commencer comme prévu à l'instant t. Cela contredit l'hypothèse de la preuve et conclut la preuve.

Q.E.D.

De cette preuve, nous constatons que d'autres séquences π_r sans blocage peuvent être obtenues en respectant simplement les dates de début S_{jk} et la propriété "Retenir et Attendre".

De ce résultat, l'ordonnancement d'un système simple se réduit à déterminer les dates de début des opérations S_{jk} afin de :

$$\text{Min } \underset{j}{\text{MAX}} \{C_j\},$$

sous les contraintes (3.1), (3.2) et (3.3).

III.3 Représentation à l'aide de RdP et vérification de blocage

Comme nous l'avons constaté précédemment, en général, les contraintes de précédence et des capacités de ressources n'impliquent pas l'absence de blocage et la contrainte d'absence de blocage ne s'exprime pas aisément par des relations mathématiques. C'est pour cela, que dans ce paragraphe, nous abordons les points suivants: (i) la représentation par des RdP des SRU, (ii) la détection de blocage pour un état du système donné, (iii) la représentation des ordonnancements, (iv) la vérification de faisabilité des ordonnancements et (v) la construction d'un ordonnancement au plus tôt connaissant la séquence d'entrée de chaque ressource π_r. Ces résultats seront utilisés, dans les paragraphes suivants, pour l'ordonnancement des SRU généraux.

III.3.1. Modélisation des SRU à l'aide des RdP

Dans ce paragraphe, nous proposons une méthode systématique pour la modélisation des SRU. La méthode se décompose en deux étapes.

La première étape concerne la représentation du processus de fabrication de chaque travail J_j. Le modèle RdP est un chemin élémentaire (voir la figure III.2). Il contient N_j+2 places dont une place de source représente la disponibilité du travail J_j, une place de puits représente la fin du travail, et chacune des N_j places restantes représente une opération. Chaque transition t_{jk}, $\forall 0 \leq k \leq N_j$, correspond au début de l'opération $O_{j,k+1}$. Le RdP est P-temporisé. Un temps nul est associé aux places de source et de puits. Le temps associé à chacune des places restantes est égal au temps opératoire de l'opération correspondante. Les franchissements des transitions sont instantanés. Nous appelons parfois les places de ces chemins élémentaires des places-opérations.

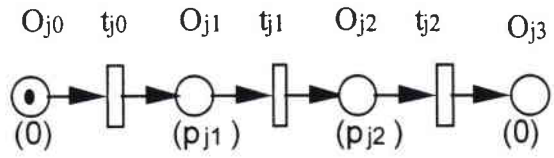

Figure III.2 : Le modèle RdP d'un travail de deux opérations

La deuxième étape complète le modèle RdP en prenant en compte les ressources nécessaires pour chaque opération. Elle représente chaque ressource $r_i \in R$ par une place r_i avec initialement un jeton correspondant à la disponibilité de la ressource. Ces nouvelles places sont appelées places de ressources.

Un arc relie une place de ressource r_i à une transition t_{jk} si $r_i \notin R_{jk}$ et $r_i \in R_{j,k+1}$, c'est-à-dire que la ressource r_i est nécessaire pour l'opération (j, k+1). Un arc relie une transition t_{jk} à une place de ressource r_i si $r_i \in R_{jk}$ et $r_i \notin R_{j,k+1}$, c'est-à-dire que la ressource r_i n'est pas nécessaire pour l'opération (j, k+1). La propriété "Retenir et Attendre" est respectée car les ressources dans R_{jk} qui ne sont plus nécessaires ne sont libérées qu'au franchissement de la transition t_{jk}, c'est-à-dire au début de l'opération (j, k+1). La figure III.3 est le modèle RdP final d'un SRU composé de deux travaux $J_1 = \{(r_1, 5), (r_2, 4)\}$ et $J_2 = \{(r_2, 3), (r_1, 4)\}$.

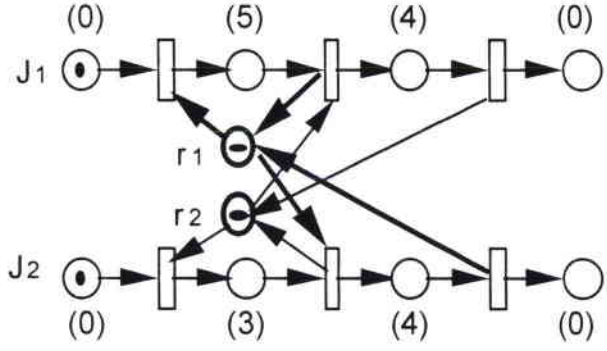

Figure III.3 : Le modèle RdP d'un SRU

III.3.2 La vérification de blocage pour un état donné

Etant donné le modèle RdP d'un SRU, nous pouvons vérifier à l'aide des techniques classiques (voir Murata, 89, Chu et Xie, 97) l'existence de blocage. Cependant, à cause de la propriété "Retenir et Attendre", la plupart des SRU sont sujets au blocage et notre objectif est de trouver l'ordonnancement optimal sans blocage. Dans cette perspective, nous considérons la détection de blocage dans un ordonnancement respectant les contraintes de précédence et

de capacité de ressources. L'objectif est de nous assurer de l'absence des travaux ne pouvant plus avancer à cause de la propriété "Retenir et Attendre".

En terme du modèle RdP, l'ordonnancement respectant les contraintes de précédence et de capacité de ressources se traduit par les dates de franchissement des transitions S_t, $\forall t \in T$ où T est l'ensemble des transitions. Le problème de la vérification de blocage consiste à déterminer, pour tout instant $\tau/S_t = \tau$ pour au moins une $t \in T$, une séquence de franchissements des transitions t telles que $S_t = \tau$. Si une telle séquence n'existe pas, nous disons que l'ordonnancement $\{S_t\}$ est avec blocage à l'instant τ.

Selon la construction du modèle RdP, chaque état d'un SRU correspond à un marquage M de son modèle RdP. M(r) = 0 pour une place de ressource r si, et seulement si, la ressource est occupée et M(p) = 1 pour une place-opération p si, et seulement si, l'opération correspondante est en cours. Le résultat ci-dessous donne une caractérisation de blocage à l'aide de la notion de siphons. Rappelons qu'un siphon est un ensemble de places Q tel que chaque transition d'entrée d'une place dans Q est également une transition de sortie d'autres places dans Q, c'est-à-dire ${}^\bullet Q \subseteq Q^\bullet$.

Théorème 3.2 : Etant donné un ordonnancement $\{S_t, \forall t \in T\}$ satisfaisant les contraintes de précédence et de capacité de ressources, le système est avec blocage à l'instant τ si et seulement si le RdP $(N, M_{\tau-})$ possède un siphon vide avec des places de ressource où $M_{\tau-}$ est le marquage du RdP immédiatement avant l'instant τ.

Preuve :

($\Leftarrow$) Soit U le siphon non marqué par $M_{\tau-}$ et soit W l'ensemble des places de ressources dans U, c'est-à-dire $W = U \cap R$. Selon la construction du modèle RdP, à chaque place de ressources $r \in W$ correspond une opération $O_{j(r),k(r)}$ telle que $M_{\tau-}(O_{j(r),k(r)}) = 1$ et $r \in R_{j(r),k(r)}$. Soit $V = \{j(r) / r \in W\}$ l'ensemble des travaux retenant les ressources dans W.

Selon la propriété des siphons, l'ensemble de places U reste non marqué quelle que soit l'évolution du système. Alors, les travaux dans V ne seront jamais terminés car, sinon, la fin d'un travail j(r), $r \in V$, libérerait éventuellement la ressource r et marquerait la place $r \in U$. Par conséquent, l'ordonnancement $\{S_t\}$ n'est pas réalisable.

($\Rightarrow$) Soit A l'ensemble des transitions à franchir à l'instant τ, c'est-à-dire $A = \{t \in T / S_t = \tau\}$. D'abord, nous remarquons que pour chaque transition $t \in A$, si elle a une place d'entrée r qui est une place de ressource, c'est-à-dire $r \in {}^\bullet t \cap R$, alors r n'est pas une place d'entrée d'une autre transition $t' \neq t \in A$ à cause de la contrainte de capacité de la ressource r et de l'hypothèse $p_{jk} > 0$. Cela implique que les transitions dans A ne sont pas en conflit.

Alors, un jeton arrivant dans une place de ressource r telle que $r \in {}^\bullet t$ et $t \in A$ reste dans r jusqu'au franchissement de t.

Puisque le système est avec blocage à l'instant τ, il existe une sous séquence σ des transitions dans A transformant $M_{\tau-}$ en marquage M, telle que $|\sigma| \subset A$, $|\sigma| \neq A$ et aucune transition dans $A-|\sigma|$ n'est franchissable à M.

Pour chaque transition $t \in A-|\sigma|$, puisqu'elle n'est pas franchissable à M, il existe au moins une place de ressource r telle que $r \in {}^\bullet t$ et $M(r) = 0$. Pour une telle place quelconque, $\forall r \in {}^\bullet t \cap R / M(r) = 0$, puisque l'ordonnancement $\{S_t\}$ vérifie les contraintes de précédence et de capacités de ressources, r est une place de sortie d'une transition t' telle que $S_{t'} = \tau$ et $t' \in A-|\sigma|$.

Soit $B = \bigcup_{t \in A-|\sigma|} \{r \in {}^\bullet t \cap R / M(r) = 0\}$ l'ensemble des places de ressources non marquées par M, à l'entrée des transitions dans $A-|\sigma|$. Nous avons: (i) chaque transition $t \in A-|\sigma|$ a au moins une place d'entrée dans B, (ii) chaque place $r \in B$ a une seule transition de sortie dans $A-|\sigma|$ et une seule transition d'entrée dans $A-|\sigma|$ parce que les transitions dans A ne sont pas en conflit et qu'il y a une seule ressource r. Les transitions dans $A-|\sigma|$ sont alors impliquées dans une relation "Retenir et Attendre" illustrée par la figure III.4.

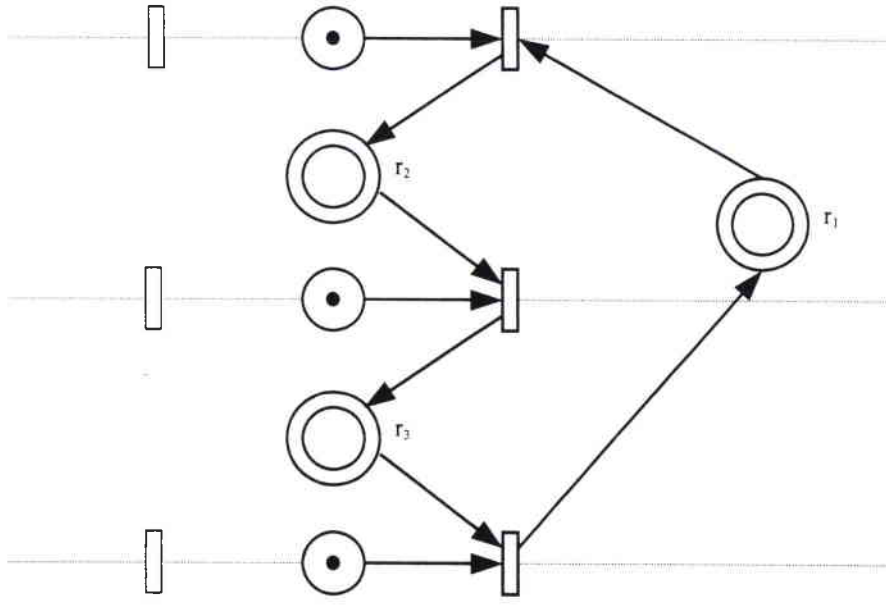

Figure III.4 : Une relation "Retenir et Attendre" où les places de ressources sont représentées par des double cercles

Puisque les transitions dans A ne sont pas en conflit, les places de ressources dans B sont vides également dans $M_{\tau-}$. Nous allons maintenant construire un siphon vide dans $M_{\tau-}$.

Selon la définition, chaque place de ressource $r \in B$ est partagée entre différentes suites P_{r_1}, P_{r_2}, ... de places-opérations. Une seule suite P est marquée. Sans perte de généralité, supposons que la suite P_{r_1} soit marquée par $M_{\tau-}$, c'est-à-dire $M_{\tau-}(P_{r_1}) = 1$ et $M_{\tau-}(P_{r_j}) = 0$, $\forall j \geq 2$ (voir figure III.5). Il est clair que la transition de sortie t de P_{r_1} est telle que $t \in A - |\sigma|$. Par conséquent, t est la transition d'entrée d'une autre place $r' \in B$. Il est alors clair que l'ensemble de places

$$Q = B \cup \bigcup_{r \in B} \{ P_{r_2} \cup P_{r_3} \cup ... \}$$

forme un siphon en tenant compte du fait que chaque place-opération a une transition d'entrée et une transition de sortie. De plus, $M_{\tau-}(Q) = 0$ et $Q \cap R \neq 0$.

Q.E.D.

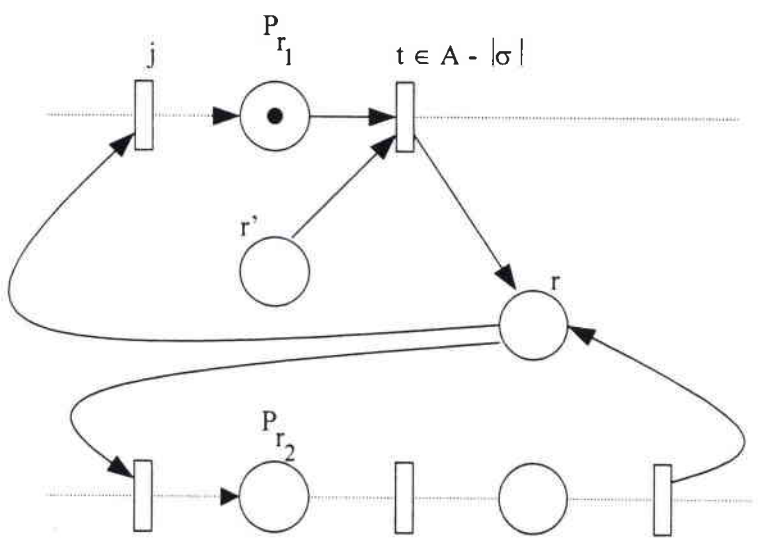

Figure III.5 : Une représentation RdP d'une ressource partagée

Remarque 3.1 : De la preuve du théorème 3.2, pour chaque instant τ, si $(N, M_{\tau-})$ ne possède pas de siphon vide, la séquence des transitions à franchir à l'instant τ peut être obtenue dans un ordre quelconque, à condition de respecter les règles de franchissement. Si $(N, M_{\tau-})$ possède un siphon vide, on a une situation de blocage quel que soit l'ordre de franchissements. L'algorithme suivant peut être utilisé pour vérifier les conditions du théorème 3.2.

Algorithme 3.1 (Vérification de l'existence d'un siphon vide dans un RdP (N,M))

1. Supprimer les places de sources et de puits.
2. Supprimer les places marquées par M.
3. Supprimer les transitions de source et leurs places de sorties.
4. Si de nouvelles transitions de source existent, alors répéter l'étape 3.
5. Si toutes les places et transitions sont supprimées, alors aucun siphon n'est vide dans M ; sinon, les places restantes forment un siphon non marqué par M.

III.3.3 Faisabilité d'un ordonnancement donné

Dans ce paragraphe, nous étudions la vérification de la faisabilité d'un ordonnancement donné. Il est évident que la vérification dépend de la représentation de l'ordonnancement. Nous considérons ici deux possibilités : (i) seules les séquences d'entrée dans les ressources π_r sont données; (ii) seules les dates de début S_{jk} sont données.

III.3.3.1 Cas où les séquences d'entrée dans les ressources sont connues

Lorsque les séquences d'entrée dans les ressources π_r sont connues, nous pouvons modifier le modèle RdP du système pour prendre en compte les séquences d'entrée. Soit π_r = O(r,1), O(r,2), ..., O(r,N(r)) la séquence d'entrée de la ressource r où N(r) désigne le nombre d'opérations nécessitant r. Le nouveau RdP est obtenu en remplaçant chaque place de ressource par N(r) places p(r,1), p(r,2), ..., p(r,N(r)) reliées, selon la séquence π_r, aux transitions d'entrée/sortie des places O(r,1), O(r,2), ..., O(r,N(r)). La place p(r,1) contient initialement un jeton et les autres places sont initialement vides. La figure III.6 illustre ce cas avec $\pi_{r_1} = O_{11}O_{22}$ et $\pi_{r_2} = O_{21}O_{12}$.

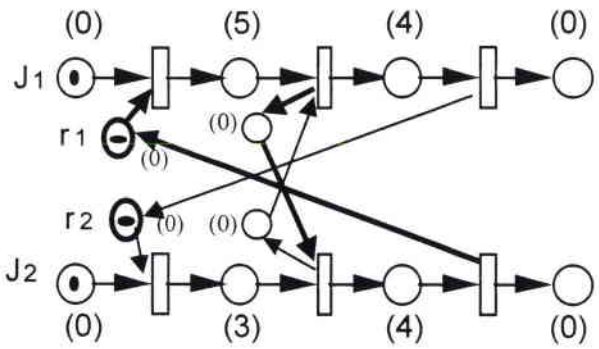

Figure III.6 : Un modèle RdP avec séquences d'entrée

Il est clair que le modèle modifié devient un graphe d'événements si nous supprimons les places de source et de puits. Par conséquent,

Théorème 3.3 : Etant données les séquences d'entrées dans les ressources, le système est sans blocage si, et seulement si, chaque circuit élémentaire du RdP modifié contient au moins un jeton. Si le système est sans blocage, alors les dates de début au plus tôt de chaque opération est égale à la date de franchissement de la transition correspondante. Les dates de franchissement S_t peuvent être déterminées par la relation suivante :

$$S_t = \max_{t'/M_0(t',t)=0 \wedge t' \in {}^{\bullet\bullet}t} \{S_{t'} + p(t',t)\}$$

où ${}^{\bullet\bullet}t$ est l'ensemble des transitions reliées à t via une place, $M_0(t',t)$ et $p(t',t)$ sont respectivement le marquage initial et le temps associé à la place qui relie t' à t.

Preuve :

Un ordonnancement est sans blocage si son modèle RdP modifié, en supprimant les places de source et de puits, contient une séquence franchissable qui contient chaque transition une fois. Selon les propriétés des graphes d'événements (Murata, 89), une telle séquence existe si, et seulement si, chaque circuit élémentaire contient au moins un jeton. Ceci prouve la première partie. La deuxième partie peut être prouvée comme suit : chaque transition t peut être franchie si chaque place d'entrée est marquée et que le jeton est disponible. Pour chaque place d'entrée, telle que $M_0(t',t) = 1$, le jeton demandé est disponible à la date zéro. Pour chaque place d'entrée (t',t) telle que $M_0(t',t) = 0$, le jeton demandé arrive à la date $S_{t'}$ et il sera disponible à la date $S_{t'} + p(t',t)$. Nous avons donc le résultat suivant :

$$S_t = \max_{t'/M_0(t',t)=0 \wedge t' \in {}^{\bullet\bullet}t} \{S_{t'} + p(t',t)\}.$$

Q.E.D.

Remarque 3.2 : L'ordonnancement dans la figure III.4 n'est pas sans blocage parce que le RdP modifié contient un circuit élémentaire non marqué. Le blocage se produit à la date 5, après la fin des opérations O_{11} et O_{21}.

Remarque 3.3 : Le théorème 3.3 peut non seulement être utilisé pour la vérification de blocage mais il peut également être utilisé pour construire un ordonnancement au plus tôt si on connaît les séquences d'entrée dans les ressources.

III.3.3.2 Cas où seules les dates de début sont connues

Puisque les dates de début S_{jk} sont connues, il est alors aisé de vérifier les contraintes de précédence (3.2) et les contraintes de capacité de ressources (3.3). Quant à l'absence de blocage, nous pouvons utiliser la méthode présentée dans le paragraphe III.3 pour nous assurer de l'absence de blocage pour tout instant τ tel que $S_{jk} = \tau$, pour certaine opération (j,k).

Nous pouvons également utiliser les méthodes du paragraphe III.3.3.1 pour la vérification de blocage. Comme les temps opératoires sont tous positifs, c'est-à-dire que $p_{jk} > 0$, sous les contraintes de précédence et de capacité de ressources, il existe une seule séquence d'entrée pour chaque ressource π_r respectant les dates de début car les exécutions de deux opérations partageant une ressource r sont nécessairement décalées dans le temps. Alors, la séquence d'entrée pour chaque ressource π_r est telle que $S_{O(r,1)} \leq S_{O(r,2)} \leq \ldots$ Le RdP modifié peut être construit comme dans le paragraphe III.3.3.1. Il est clair que la date de franchissement de la transition t, S_t, est égale à la date de début de l'opération (j,k), S_{jk}, telle que t est une transition d'entrée de la place-opération O_{jk}.

Théorème 3.4 : Etant données les dates de début des opérations, le système est sans blocage si, et seulement si, chaque circuit du RdP modifié contient au moins un jeton et si les dates de franchissement S_t respectent les contraintes suivantes :

$$S_t \geq \max_{t'/M_0(t',t)=0 \wedge t' \in {}^{\bullet\bullet}t} \{S_{t'} + p(t',t)\}.$$

Preuve :

($\Rightarrow$) Trivial puisque, selon la preuve du théorème 3.3, le RdP modifié ne contient pas de circuit élémentaire non marqué et que chaque transition t n'est pas franchissable avant la date :

$$\max_{t'/M_0(t',t)=0 \wedge t' \in {}^{\bullet\bullet}t} \{S_{t'} + p(t',t)\}.$$

(⇐) L'ordonnancement est sans blocage car le RdP modifié ne contient pas un circuit élémentaire non marqué. De plus,

$$S_t \geq \max_{t'/M_0(t',t)=0 \wedge t' \in {}^{\bullet\bullet}t} \{S_{t'} + p(t',t)\},$$

ce qui implique les contraintes de précédence et de capacité de ressources. Donc, nous avons un ordonnancement admissible.

Q.E.D.

Soit un système composé de deux travaux (figure III.3) et soit l'ordonnancement suivant : $S_{11} = 0$, $S_{12} = 5$, $S_{21} = 0$, $S_{22} = 5$. La séquence de dates de début des opérations est $S_{11} \leq S_{21} \leq S_{22} \leq S_{21}$ et la séquence d'entrée sur chaque ressource est O_{11} O_{22} sur r_1 et O_{21} O_{12} sur r_2. Le RdP modifié est le même que celui donné dans la figure III.6. Il est aisé de vérifier que les conditions du théorème 3.4 ne sont pas vérifiées et que donc l'ordonnancement n'est pas sans blocage.

III.4 Méthodes de résolution

L'objectif de ce paragraphe est de présenter des méthodes heuristiques permettant la construction des ordonnancements sans blocage pour les Systèmes à Ressources Unitaires (SRU). Vu les résultats de la complexité du chapitre II, nous savons que l'ordonnancement d'un SRU est un problème fortement combinatoire. Ce caractère combinatoire et notre souci de développer des méthodes pour des problèmes de taille réaliste nous interdisent les méthodes exactes et les méthodes par recherche dans l'espace des marquages du type A* que nous avons présentées dans le chapitre I. D'un autre côté, la propriété "Retenir et Attendre" rend les job-shops MRB sensibles au blocage et nous interdit l'utilisation des méthodes heuristiques classiques telles que les méthodes par permutation des opérations sur les chemins critiques. C'est la raison pour laquelle, nous proposons dans ce paragraphe de nouvelles méthodes heuristiques en prenant en compte le caractère combinatoire du problème et le problème du blocage.

III.4.1 Présentation des méthodes heuristiques

Les méthodes heuristiques que nous proposons sont toutes des méthodes gloutonnes . Elles ordonnancent les travaux les uns après les autres selon un ordre donné. Une fois ceux-ci ordonnancés, les dates de début S_{jk} d'un travail ne seront pas modifiées dans la suite de la construction d'un ordonnancement. Deux méthodes sont proposées dans ce paragraphe.

Sans perte de généralité, nous supposons que l'ordre d'ordonnancement des travaux est $J_1, J_2, \ldots, J_n$.

III.4.1.1 Heuristique 1

La première méthode, appelée Heuristique 1 (Damasceno et Xie, 98-a), est une méthode conservatrice. Lors de l'ordonnancement d'un travail J_j, elle ordonnance ses opérations, les unes après les autres, suivant le processus de fabrication. Pour l'ordonnancement, d'une opération O_{jk}, nous déterminons, pour chaque ressource r, la date u_r à partir de laquelle la ressource est toujours libre. Il est clair que l'opération O_{jk} peut commencer sans risque de blocage à la date

$$S_{jk} = \text{Max } \{S_{j,k-1} + p_{j,k-1}, \text{Max } \{u_r / r \in R_{jk}\}\}.$$

Les indicateurs u_r sont alors actualisés et la méthode peut continuer de la même manière. Plus formellement.

Algorithme 3.2 (Heuristique 1 (ordonnancement selon l'ordre $J_1, J_2, ..., J_n$))

1. Initialisation
   $u_r \leftarrow 0, \forall r \in R.$

2. Ordonnancement dans l'ordre des travaux et des opérations :
   Pour j = 1 à n faire
      Pour k = 1 à N_j faire
         2.1. $S_{jk} = \text{Max } \{S_{j,k-1} + p_{j,k-1}, \text{Max } \{u_r / r \in R_{jk}\}\}$
         2.2. $u_r \leftarrow S_{jk} + p_{jk}, \forall r \in R_{jk}$

3. Le makespan $C_{max} \leftarrow S_{n,N_n} + p_{n,N_n}$.

L'application de l'Heuristique 1 au job-shop MRB de la figure III.3 donne l'ordonnancement suivant :

$u_{r_1} = 0$	$u_{r_2} = 0$	$S_{11} = 0$
$u_{r_1} = 5$	$u_{r_2} = 0$	$S_{12} = 5$
$u_{r_1} = 5$	$u_{r_2} = 9$	$S_{21} = 9$
$u_{r_1} = 12$	$u_{r_2} = 9$	$S_{22} = 12$
$C_{max} = 16$		

Table III.1 : Ordonnancement d'un job-shop MRB par l'heuristique 1

Le diagramme de Gantt, de la figure III.7 donne une meilleure illustration de l'ordonnancement. La propriété principale de l'Heuristique 1 est clairement illustrée par l'ordonnancement de l'opération O_{21}. Bien que la ressource r_2 soit libre dans l'intervalle de temps [0,5], la méthode n'essaie pas d'ordonnancer O_{21} dans cette fenêtre de temps. De ce point de vue, l'Heuristique 1 est une méthode conservatrice. Bien sûr, la tentative conduirait à un blocage dans cet exemple mais une méthode moins conservatrice pourrait améliorer l'ordonnancement dans les cas généraux. Ceci est le point de départ de l'Heuristique 2 que nous présenterons ultérieurement.

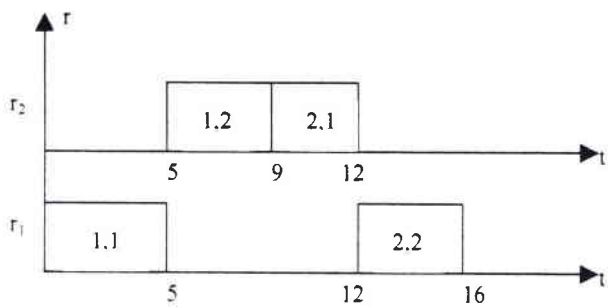

Figure III.7 : Le diagramme de Gantt d'un ordonnancement construit par l'Heuristique 1

Théorème 3.5 : L'ordonnancement obtenu à l'aide de l'Heuristique 1 est sans blocage.

Preuve :

Pour preuve, nous utilisons le modèle RdP modifié prenant en compte les séquences d'entrée dans les ressources. Selon l'Heuristique 1, les contraintes de précédence et de capacité de ressources sont évidemment respectées. De plus, les séquences d'entrée dans les ressources respectent l'ordre des travaux et l'ordre des opérations , c'est-à-dire $j < j'$ ou $j = j'$ et $k < k'$ si $\pi_r[i] = (j, k)$ et $\pi_r[i+1] = (j', k')$.

Par conséquent, la séquence d'entrée π_r dans chaque ressource r passe des travaux de l'indice inférieur aux travaux de l'indice supérieur et pour chaque travail, des opérations de l'indice inférieur aux opérations de l'indice supérieur. Ceci est vrai pour toute place vide du RdP modifié. Pour résumer, il n'existe pas de circuit élémentaire non marqué dans le RdP modifié et l'ordonnancement est donc sans blocage selon le théorème 3.3.

Q.E.D.

Remarque 3.4 : Il est aisé de vérifier que l'Heuristique 1 s'applique à l'ordonnancement de tout Système à Multiples Unités (SMU) avec ou sans opérations ZD.

III.4.1.2 Heuristique 2

Comme dans l'Heuristique 1, l'Heuristique 2 (Damasceno et Xie, 98-a et 98-b) ordonnance les travaux les uns après les autres et, une fois ceux-ci ordonnancés, les dates de début d'un travail ne seront pas modifiées. La différence entre les deux méthodes réside dans l'ordonnancement de chaque travail. L'Heuristique 2 ordonnance de manière optimale le travail en cours de considération et cherche à commencer les opérations au plus tôt, tout en évitant les blocages. Plus précisément,

Algorithme 3.3 (Heuristique 2)

1. Choisir un ordre des travaux. Soit $J_1, J_2, ..., J_n$ l'ordre, sans perte de généralité.

2. Pour chaque travail J_j, résoudre le problème à un travail pour J_j en tenant compte des ordonnancements des travaux $J_1, J_2, ..., J_{j-1}$.

Le problème à un travail est non-trivial et sera abordé dans le paragraphe III.4.2.

III.4.1.3 Optimisation de l'ordre des travaux

Dans les deux méthodes proposées ci-dessus, nous supposons qu'un ordre des travaux soit donné. Il est clair que la qualité de l'ordonnancement obtenu dépend de l'ordre utilisé. Dans ce paragraphe, nous proposons une méthode recuit simulé pour déterminer l'ordre afin de minimiser le makespan. L'idée de base est de vérifier tous les ordres possibles et de choisir l'ordre avec $Min\ C_{max}$. Plus précisément,

Algorithme 3.4 (Optimisation de l'ordre des travaux)

1. Choisir un ordre initial S_0 (par exemple $J_1, J_2, ..., J_n$) et calculer sa valeur de critère $f(S_0)$ à l'aide de l'Heuristique 1 ou de l'Heuristique 2.

2. Choisir une température initiale $T_0 > 0$ et faire $T \leftarrow T_0$.

3. Générer au hasard un ordre voisin S_1 (par permutation des deux travaux quelconques) et calculer sa valeur de critère $f(S_1)$ à l'aide de l'Heuristique 1 ou de l'Heuristique 2.

4. Si $f(S_1) \leq f(S_0)$, faire $S_0 \leftarrow S_1$. Sinon, faire $S_0 \leftarrow S_1$ avec la probabilité $\exp(-\Delta_f / T)$ où $\Delta_f = f(S_1) - f(S_0)$.

5. Faire $T = T\,r$

6. Si $T > T_f$, où T_f est la température finale, aller à l'étape 3. Sinon, arrêt.

III.4.2 Problèmes à un travail

Dans ce paragraphe, nous nous plaçons dans le cadre de l'Heuristique 2 et considérons l'ordonnancement de chaque travail.

Dans la suite de ce paragraphe, nous formulons d'abord le problème à un travail et présentons une méthode par recherche exhaustive ainsi qu'une méthode Programmation Dynamique, pour les SRU généraux et pour les systèmes simples.

III.4.2.1 Formulation du problème à un travail

Lors de l'ordonnancement d'un travail J_j, certains travaux sont déjà considérés et les dates de début de leurs opérations sont fixées. Soit

$$F = \{J_1, J_2, ..., J_{j-1}\}$$

l'ensemble des travaux déjà ordonnancés. Ordonnancer J_j consiste à déterminer les dates de début de ses opérations S_{jk} afin de minimiser sa date de fin C_j tout en respectant les ordonnancements des travaux dans F et en évitant les blocages.

Plus formellement, le problème à un travail pour J_j s'exprime de la manière suivante (nous rappelons qu'il s'agit d'un modèle à temps discret de H périodes élémentaires) :

(SP) :

$$\text{Min } C_j,$$

sous les contraintes suivantes :

$$C_j = S_{j,Nj+1} \tag{3.6}$$

$$S_{jk} + p_{jk} \leq S_{j,k+1},\ \forall 1 \leq k \leq N_j \tag{3.7}$$

$$\sum_{(i,l)/\, i \in F, R_{il} \cap R_{jk} \neq \phi} \mathbf{1}\{S_{il} \leq t < S_{i,l+1}\} = 0,\ \forall 1 \leq k \leq N_j\ \forall\ S_{jk} \leq t < S_{j,k+1} \qquad (3.8)$$

Il n'y a pas de blocage aux instants τ- avec $\tau \in \{S_{j1}, ..., S_{jNj}\}$ (3.9)

Dans cette formulation, nous avons étendu la notation des dates de début S_{jk} au cas $k = N_j + 1$ où N_j est le nombre des opérations. Ainsi, la relation (3.6) définit la date de fin du travail J_j. Les contraintes (3.7) expriment les contraintes de précédence. Il est clair que $S_{j,Nj+1} = S_{jNj} + p_{jNj}$. Les relations (3.8) correspondent aux contraintes de capacité de ressources. En effet, pendant l'opération (j,k), c'est-à-dire $S_{jk} \leq t < S_{jk+1}$, aucune opération (i,l) des travaux i dans F partageant les ressources pour (j,k) ne peut être exécutée, c'est-à-dire que les intervalles $[S_{il}, S_{i,l+1})$ et $[S_{jk}, S_{j,k+1})$ sont disjoints. La relation (3.9) concerne l'absence de blocage. Nous notons que l'ordonnancement des travaux dans F est sans blocage et respecte les contraintes de précédence et de capacité de ressources.

Pour simplifier la formulation, nous introduisons la notation suivante :

$$I_{kt} = \sum_{(i,l)/\, i \in F, R_{il} \cap R_{jk} \neq \phi} \mathbf{1}\{S_{il} \leq t < S_{il+1}\},\ \forall 1 \leq k \leq N_j,\ \forall t \in \{1, 2, ..., H\} \qquad (3.10)$$

Selon la définition, I_{kt} désigne le nombre de ressources nécessaires pour l'opération (j,k) qui sont retenues dans la période t par les travaux déjà ordonnancés. Notons que les nombres I_{kt} sont donnés pour le problème à un travail (SP) et ils ne dépendent pas de l'ordonnancement du travail J_j.

Grâce à cette nouvelle notation, le problème à un travail peut être formulé comme suit :

(SP) :

$$\text{Min } C_j,$$

sous les contraintes suivantes :

$$C_j = S_{j,Nj+1} \qquad (3.11)$$

$$S_{jk} + p_{jk} \leq S_{j,k+1},\ \forall 1 \leq k \leq N_j \qquad (3.12)$$

$$I_{kt} = 0,\ \forall 1 \leq k \leq N_j\ \forall\ S_{jk} \leq t < S_{j,k+1} \qquad (3.13)$$

Il n'y a pas de blocage aux instants τ- avec $\tau \in \{S_{j1}, ..., S_{jNj}\}$ (3.14)

III.4.2.2 Une méthode par recherche exhaustive

L'idée centrale de cette méthode (Damasceno et Xie, 98-c) est d'ordonnancer les opérations de J_j, les unes après les autres, selon le processus de fabrication, tout en acceptant le *backtracking*. L'objectif est de trouver pour chaque opération (j,k) une fenêtre de temps $W_k = (a_k, b_k)$ telle que (i) chaque opération peut être exécutée dans sa fenêtre de temps, (ii) les contraintes de précédence sont vérifiées et (iii) l'ordonnancement est sans blocage.

Les restrictions (i) - (ii) exigent l'existence d'un ordonnancement $\{S_{jk}\}$ tel que :

$$a_k \le S_{jk} \le b_k - p_{jk}, \forall k = 1, 2, ..., N_j \tag{3.15}$$

$$S_{jk} \ge S_{j,k-1} + p_{j,k-1}, \forall\, k = 2, ..., N_j \tag{3.16}$$

$$S_{jk} \le b_{k-1}, \forall\, k = 2, ..., N_j \tag{3.17}$$

Les contraintes (3.15) sont dues à la restriction (i), les contraintes (3.16) aux contraintes de précédence et les contraintes (3.17) à la propriété "Retenir et Attendre".

Théorème 3.6 : Il existe un ordonnancement $\{S_{jk}\}$ vérifiant (3.15) – (3.17) si, et seulement si, $L_k \le U_k$, $\forall k = 1, \ldots, N_j$ où $L_1 = a_1$, $U_1 = b_1 - p_{j1}$, $L_k = \max \{L_{k-1} + p_{j,k-1}, a_k\}$, $U_k = \text{Min } \{b_k - p_{jk} . b_{k-1}\}$, $\forall k = 2, \ldots, N_j$. De plus, si les conditions sont vraies, $\{L_k\}$ est un ordonnancement admissible et, pour tout ordonnancement $\{S_{jk}\}$ admissible, $S_{jk} \ge L_k$, $\forall k = 1, \ldots, N_j$.

Preuve :

($\Rightarrow$) Soit $\{S_{jk}\}$ un ordonnancement vérifiant (3.15) – (3.17). Alors, $S_{jk} \le \text{Min } \{b_k - p_{jk}, b_{k-1}\} = U_k$, $\forall k = 2, \ldots, N_j$ et $S_{j1} \le b_1 - p_{j1} = U_1$. Nous montrons par récurrence que $S_{jk} \ge L_k$. Ceci est vrai pour $k = 1$ car $S_{j1} \ge a_1 = L_1$. Supposons que $S_{jk} \ge L_k$. Des contraintes (3.15) – (3.16), on a $S_{j,k+1} \ge \max \{S_{jk} + p_{jk}, a_{k+1}\}$. Puisque $S_{jk} \ge L_k$, $S_{j,k+1} \ge \max \{L_k + p_{jk}, a_{k+1}\} = L_{k+1}$. Nous avons par récurrence, $S_{jk} \ge L_k$, $\forall k = 1, \ldots, N_j$. Pour conclure, $L_k \le S_{jk} \le U_k$, $\forall k = 1, \ldots, N_j$.

($\Leftarrow$) Considérons l'ordonnancement $S_{jk} = L_k$, $\forall k = 1, \ldots, N_j$. De la définition de L_k,

$$S_{jk} = L_k \ge a_k, \forall k = 1, \ldots, N_j, \tag{3.18}$$

Puisque $L_k \le U_k$, nous avons $S_{jk} \le \text{Min } \{b_k - p_{jk}, b_{k-1}\}$, $\forall k = 2, \ldots, N_j$ et $S_{j1} \le b_1 - p_{j1}$, c'est-à-dire :

$$S_{jk} \le b_k - p_{jk}, \forall k = 1, \ldots, N_j, \tag{3.19}$$

$$S_{jk} \leq b_{k-1}, \forall k = 2, \ldots, N_j, \tag{3.20}$$

Finalement, $S_{jk} = L_k = \max \{L_{k-1} + p_{j,k-1}, a_k\} \geq L_{k-1} + p_{j,k-1}, \forall k = 2, \ldots, N_j$. D'où,

$$S_{jk} \geq S_{j,k-1} + p_{j,k-1}, \forall k = 2, \ldots, N_j. \tag{3.21}$$

Les relations (3.18) – (3.21) correspondent aux contraintes (3.15) – (3.17).

Q.E.D.

Les restrictions (i) – (ii) s'assurent que les contraintes de précédence et de capacité des ressources sont respectées. Il reste à vérifier l'absence de blocage.

Théorème 3.7 : Sous les conditions du théorème 3.6, il existe un ordonnancement sans blocage si, et seulement si, $a_k < b_{k-1}$ ou M_k, $\forall k = 2, \ldots, N_j$, ne sont pas des états de blocage où M_k est l'état du système à l'instant a_k- en supposant que l'opération (j,k-1) est en cours à l'instant a_k-.

Preuve :

Notons que le théorème 3.6 garantit l'existence d'un ordonnancement respectant les contraintes de précédence et de capacité des ressources. Pour un tel ordonnancement S_{jk}, nous montrons le théorème par récurrence avec $N_j = 1, 2, \ldots$.

D'abord, il est trivial que les blocages ne se produisent pas si le nouveau travail N_j nécessite une seule opération et le théorème est vrai pour $N_j = 1$.

Supposons que le théorème soit vrai avec $N_j = n$. Nous montrons qu'il est vrai pour $N_j = n + 1$. Selon l'hypothèse de récurrence, il n'y a pas de blocage avant le début de l'opération (j, n+1), c'est-à-dire avant $S_{j,n+1}$. Des contraintes (3.15) – (3.17), $a_{n+1} \leq S_{j,n+1} \leq b_n$. Nous remarquons que les ressources dans R_{jn} ne sont pas retenues dans $[a_n, b_n]$ par les travaux déjà ordonnancés et les ressources dans $R_{j,n+1}$ ne sont pas retenues dans $[a_{n+1}, b_{n+1}]$ par les travaux déjà ordonnancés. Par conséquent, toutes les ressources dans $R_{jn} \cup R_{j,n+1}$ sont libérées à l'instant $S_{j,n+1}$ si $a_{n+1} < b_n$. Dans ce cas, l'opération (j, n+1) peut commencer à la date $S_{j,n+1}$ et le théorème est prouvé.

Considérons le cas où $a_{n+1} = b_n$. Alors $S_{j,n+1} = b_n$. Selon l'hypothèse de la récurrence, il existe un ordonnancement $\{S_{jk}\}$ pour les travaux $\{J_1, J_2, \ldots, J_j\}$ avec $N_j = n + 1$ tel que (a) les contraintes de précédence et de capacité des ressources sont vérifiées, (b) il n'y a pas de blocage avant l'instant b_n. Alors il existe un ordonnancement sans blocage si, et seulement si, l'opération (j, n+1) et l'ensemble des opérations (i, l), telles que $S_{il} = b_n$, peuvent commencer à cette date. Selon le théorème 3.2, (j, n+1) et l'ensemble d'opérations (i, l), telles que $S_{il} = b_n$,

peuvent commencer à la date b_n si et seulement si l'état du système $M_{\tau-}$ avec $\tau = b_n = a_{n+1}$ n'est pas un état de blocage, c'est-à-dire s'il n'y a pas de siphon vide. Ceci conclut la preuve.

Q.E.D.

Compte tenu des résultats antérieurs, nous pouvons calculer l'ordonnancement optimal d'un travail selon l'algorithme suivant :

Algorithme 3.5

1. Initialiser : $X_k \leftarrow 0$ et $k \leftarrow 1$.

2. Déterminer la fenêtre de temps maximale $W_k = (a_k, b_k)$ telle que $a_k \geq X_k$, $b_k - a_k \geq p_{jk}$, $a_k \leq b_{k-1}$ et $I_{kt}(t) = 0$, $\forall t \in W_k$. Par convention, $b_0 = \infty$.

3. Si W_k n'existe pas, alors $X_{k-1} \leftarrow b_{k-1}$, $k \leftarrow k-1$ et aller à l'étape 2.

4. Vérifier l'absence de blocage :
   Si $k > 1$, faire :
   - déterminer l'état du système M à la date a_k-,
   - vérifier l'absence de blocage de M à l'aide de l'algorithme 3.1. S'il y a blocage, alors $X_{k-1} \leftarrow b_{k-1}$, $k \leftarrow k-1$ et aller à l'étape 2.

5. Vérifier l'existence d'un ordonnancement admissible pour le travail j $\{(R_{j1}, p_{j1}), ..., (R_{jk}, p_{jk})\}$ avec les contraintes de fenêtre de temps W_k.

6. Si l'ordonnancement n'est pas admissible, $X_k \leftarrow b_k$ et aller à l'étape 2.

7. Si $k < N$, alors $X_{k+1} \leftarrow a_k + p_k$, $k \leftarrow k+1$ et aller à l'étape 2. Sinon, arrêt.

Théorème 3.8 : L'ordonnancement obtenu en utilisant l'algorithme 3.5 pour $J_1, ..., J_n$ est sans blocage.

Preuve :
L'ordonnancement obtenu est sans blocage en vertu du théorème 3.7.

Q.E.D.

Remarque 3.5 : Dans le cas d'un système simple, l'ordonnancement obtenu vérifiant les contraintes de précédence et de capacité de ressources est toujours sans blocage (voir le théorème 3.1). Par conséquent, l'étape 4 n'est pas nécessaire pour ordonnancer un système simple.

III.4.2.3 Une méthode Programmation Dynamique

Dans ce paragraphe, nous présentons une méthode Programmation Dynamique (Damasceno et Xie, 99) pour résoudre le problème à un travail (SP). Rappelons la formulation donnée dans III.4.2.1 :

(SP) :

$$C_j^* = \text{Min } C_j,$$

sous les contraintes suivantes :

$$C_j = S_{j.Nj+1},$$

$$S_{jk} + p_{jk} \leq S_{jk+1}, \forall 1 \leq k \leq N_j$$

$$I_{kt} = 0, \forall 1 \leq k \leq N_j \; \forall \; S_{jk} \leq t < S_{jk+1}$$

Il n'y pas de blocage aux instants τ- avec $\tau \in \{S_{j1}, ..., S_{jNj}\}$.

Afin de prendre en compte le blocage, nous définissons l'indicateur suivant :

$$B(k.t) = \begin{cases} 0, & \text{si le système est avec blocage à la fin de la période t–1 sachant que l'opération (j,k) est en cours dans t–1, c'est–à–dire } S_{jk} < t \text{ et } S_{j,k+1} \geq t. \\ 1, & \text{sinon.} \end{cases}$$

B(k,t) indique la possibilité de passer de l'opération (j,k) à l'opération (j, k+1) sans risque de blocage au début de la période t. Nous remarquons que chaque indicateur B(k,t) ne dépend pas de l'ordonnancement réellement choisi pour le nouveau travail et B(k,t) dépend seulement des travaux déjà ordonnancés. Concrètement, nous utilisons l'algorithme 3.1 pour vérifier le blocage ayant comme marquage, pour chaque travail déjà ordonnancé $i \in F$, $M(O_{il}) = 1$ pour l'opération (i,l) en cours d'exécution dans la période t-1 et $M(O_{jk}) = 1$ pour le nouveau travail J_j. S'il n'y a pas de siphon vide, B(k,t) = 1 sinon B(k,t) = 0. Il est aisé de montrer que $B(N_j, t) = 1$, $\forall t$ et B(k,t) = 0 si une des ressources dans R_{jk} est occupée par les travaux dans F dans t-1.

Avec cette nouvelle notation, le problème (SP) s'écrit comme suit :

(SP) :

$$C_j^* = \text{Min } C_j,$$

sous les contraintes suivantes :

$$C_j = S_{j.Nj+1}\ , \tag{3.22}$$

$$S_{jk} + p_{jk} \le S_{jk+1},\ \forall 1 \le k \le N_j, \tag{3.23}$$

$$I_{kt} = 0,\ \forall 1 \le k \le N_j\ \forall\ S_{jk} \le t < S_{jk+1}, \tag{3.24}$$

$$B(k, S_{jk+1}) = 1,\ \forall 1 \le k \le N_j. \tag{3.25}$$

Au lieu de résoudre directement le problème (SP), nous considérons la famille de problèmes de décision suivante :

P(k,τ) :

$$S_{jl} + p_{jl} \le S_{j.l+1},\ \forall 1 \le l \le k, \tag{3.26}$$

$$I_{lt} = 0,\ \forall l = 1, ..., k,\ S_{jl} \le t < S_{j.l+1}, \tag{3.27}$$

$$B(l, S_{j.l+1}) = 1,\ \forall l = 1, ..., k, \tag{3.28}$$

$$S_{j.k+1} = \tau. \tag{3.29}$$

Nous introduisons également l'indicateur suivant :

$$F(k,\tau) = \begin{cases} 1, \text{si } P(k,\tau) \text{ a une solution,} \\ 0, \text{sinon.} \end{cases}$$

Concrètement, le problème P(k,τ) vérifie l'existence d'un ordonnancement sans blocage pour les k premières opérations, tel que l'opération (j, k+1) puisse commencer au début de la période τ.

Relation entre les problèmes de décision et le problème à un travail

Avant d'aborder la résolution des problèmes de décision, nous montrons que les solutions des problèmes de décision permettent de déterminer une solution optimale du problème à un travail (SP).

Théorème 3.9 : $C_j^* = \text{Min } \{t \in \{1, ..., H\} / F(N_j, t) = 1\}$

Preuve :

Soit $t^* = \text{Min } \{t / F(N_j, t) = 1\}$. Nous remarquons que chaque solution $\{S_{jl}\}$ du problème $P(N_j, t^*)$ est un ordonnancement admissible du problème (SP). Donc, $t^* = S_{j,Nj+1} \geq C_j^*$. D'un autre côté, toute solution optimale $\{S_{jl}\}$ du problème (SP) est une solution du problème $P(N_j, C_j^*)$ avec $C_j^* = S_{j,Nj+1}$. Alors, $F(N_j, C_j^*) = 1$. D'où, $C_j^* \geq t^*$ et, par conséquent, $C_j^* = t^*$.

Q.E.D.

Considérons l'ordonnancement suivant :

$$S_{jk}^* = \max \{t / F(k-1, t) = 1 \wedge t \leq S_{j,k+1}^* - p_{jk}\}$$

où $S_{j,N_j+1}^* = C_j^*$.

Théorème 3.10 : $\{S_{jk}^*\}$ est une solution du problème (SP). De plus, $S_{jk}^* \geq S_{jk}$, $\forall k = 1, ..., N_j$, pour toute solution admissible du problème (SP) avec $S_{j,Nj+1} = C_j^*$.

Preuve :

Nous montrons d'abord par induction que S_{jk}^* existe. Il est clair que S_{j,N_j+1}^* existe. Supposons que $S_{j,k+1}^*$ existe. Alors, $F(k, S_{j,k+1}^*) = 1$, c'est-à-dire le problème $P(k, S_{j,k+1}^*)$ a une solution. Soit $\{S_{jl}, \forall 1 \leq l \leq k+1\}$ cette solution. Nous avons :

$$S_{jk} \leq S_{j,k+1}^* - p_{jk} \text{ et } F(k, S_{jk}) = 1,$$

ce qui prouve l'existence de S_{jk}^*. Par induction, tous les S_{jk}^* existent.

Pour prouver l'admissibilité de $\{S_{jk}^*\}$, il faut vérifier les contraintes de précédence, les contraintes de capacité des ressources et l'absence de blocage. Les contraintes de précédence sont vraies puisque $S_{jk}^* \leq S_{j,k+1}^* - p_{jk}$, $\forall 1 \leq k \leq N_j$. Les contraintes de capacité sont aussi respectées si $I_{kt} = 0$, $\forall 1 \leq k \leq N_j$ et $S_{jk}^* \leq t < S_{j,k+1}^*$. Pour nous assurer de cela, considérons une solution $\{S_{jl}, \forall 1 \leq l \leq k+1\}$ du problème $P(k, S_{j,k+1}^*)$. Une telle solution existe car $F(k, S_{j,k+1}^*) = 1$. De (3.26), $S_{jk}^* \geq S_{jk}$. Puisque $\{S_{jl}\}$ est une solution de $P(k, S_{j,k+1}^*)$, $I_{kt} = 0$, $\forall S_{jk} \leq t$

$< S^{*}_{j,k+1}$, ce qui implique $I_{kt} = 0$, $S^{*}_{jk} \leq t < S^{*}_{j,k+1}$. L'absence de blocage est assurée par la contrainte (3.28) avec $l = k$ et $\tau = S^{*}_{jk}$ qui implique $B(k, S^{*}_{j,k+1}) = 1$, $\forall 1 \leq k \leq N_j$.

Nous démontrons maintenant la dernière partie du théorème par l'absurde. Supposons qu'il existe une solution $\{S_{jk}\}$ du problème (SP) telle que $S_{j,Nj+1} = C^{*}_{j}$ et $\exists$ k, $S_{jk} > S^{*}_{jk}$. Soit v le plus grand indice k tel que $S_{jk} > S^{*}_{jk}$. Puisque $S_{j,Nj+1} = S^{*}_{j,N_j+1} = C^{*}_{j}$, $v < N_j + 1$ et $S_{jv} > S^{*}_{jv}$ et $S_{j,v+1} \leq S^{*}_{j,v+1}$. Notons que $\{S_{jl}, \forall 1 \leq l \leq v\}$ est une solution de $P(v-1, S_{jv})$, c'est-à-dire $F(v-1, S_{jv}) = 1$. De plus, $S_{jv} \leq S_{j,v+1} - p_{jv} \leq S^{*}_{j,v+1} - p_{jv}$. Alors,

$$F(v-1, S_{jv}) = 1 \text{ et } S_{jv} \leq S^{*}_{j,v+1} - p_{jv},$$

ce qui implique que $S^{*}_{j,v+1} \geq S_{jv}$, ce qui contredit l'hypothèse $S_{jv} > S^{*}_{jv}$ et conclut la preuve par l'absurde. *Q.E.D.*

Nous considérons également l'ordonnancement suivant :

$$S^{\circ}_{jk} = \text{Min}\{t \leq S^{\circ}_{j,k+1} - p_{jk} / F(k-1, t) = 1 \wedge I_{k\tau} = 0, \forall t \leq \tau < S^{\circ}_{j,k+1}\}$$

avec $S^{\circ}_{j,k+1} = C^{*}_{j}$.

Théorème 3.11 : $\{S^{\circ}_{jk}\}$ est une solution admissible du problème (SP). De plus, $S^{\circ}_{jk} \leq S_{jk}$, $\forall k = 1, ..., N_j$, pour toute solution admissible $\{S_{jk}\}$ du problème (SP).

La preuve est similaire à celle du théorème 3.10 et est ignorée. Nous notons que l'ordonnancement $\{S^{\circ}_{jk}\}$ est un ordonnancement "calé à gauche" et $\{S^{*}_{jk}\}$ un ordonnancement "calé à droite".

Résolution des problèmes de décision

Dans cette section, nous mettons en évidence une récurrence pour la détermination des $F(k, \tau)$. Pour cela, nous considérons les problèmes $P(k+1, \tau)$, c'est-à-dire :

$P(k+1,\tau)$:

$$S_{jl} + p_{jl} \leq S_{j,l+1}, \forall 1 \leq l \leq k+1$$

$$I_{lt} = 0, \forall\ 1 \leq l \leq k+1, \forall\ S_{jl} \leq t < S_{j,l+1}$$

$$B(l, S_{j,l+1}) = 1 \ , \ \forall 1 \le l \le k+1$$

$$S_{j,k+2} = \tau$$

qui peut être mis sous la forme suivante :

$$S_{jl} + p_{jl} \le S_{j,l+1}, \ \forall 1 \le l \le k \tag{3.30}$$

$$I_{lt} = 0, \ \forall \ 1 \le l \le k, \ \forall \ S_{jl} \le t \le S_{jl+1} \tag{3.31}$$

$$B(l, S_{j,l+1}) = 1 \ , \ \forall 1 \le l \le k \tag{3.32}$$

$$S_{jk+1} \le \tau - p_{jk+1} \tag{3.33}$$

$$I_{k+1,t} = 0, \ \forall \ S_{jk+1} \le t < \tau \tag{3.34}$$

$$B(k+1, \tau) = 1. \tag{3.35}$$

De la définition P(k, τ), il existe une solution $\{S_{jl}, \ \forall 1 \le l < k+1\}$ vérifiant (3.30) - (3.32) si, et seulement si, $F(k, S_{j,k+1}) = 1$. Nous avons donc :

$$F(k+1,\tau) = \begin{cases} 1, \text{if } \exists\, t \le \tau - p_{j,k-1} / F(k,t) = 1 \wedge B(k+1,\tau) = 1 \wedge \sum_{i=t}^{\tau-1} I_{k-1,i} = 0 \\ 0, \text{sinon} \end{cases}$$

ou de manière équivalente,

$$F(k+1,\tau) = \bigvee_{t \le \tau - p_{jk}} F(k,t) \wedge B(k+1,\tau) \wedge 1\{I_{k+1,s}, \forall t \le s < \tau\} \tag{3.36}$$

où "$\wedge$" désigne l'opérateur logique *ET* et "$\vee$" l'opérateur logique *OU*. Par convention, $F(0, \tau) = 1, \ \forall 1 \le \tau \le H$.

La méthode Programmation Dynamique pour le problème à un travail peut être résumée comme suit :

Algorithme 3.6 (Programmation Dynamique pour le problème à un travail)

1. Calculer les indicateurs I_{kt} sur la disponibilité des ressources pour l'opération (j,k) en utilisant la relation (3.10).

2. Calculer les indicateurs de blocage $B(k, \tau)$ en utilisant l'algorithme 3.1.

3. Solutions des problèmes de décision
   - 3.1. Initialisation : $k = 0$, $F(k, \tau) = 1$, $\forall 1 \leq \tau \leq H$.
   - 3.2. Pour $k = 0$ à $N_j - 1$,
     Pour $\tau = 1$ à H
     Calculer $F(k+1, \tau)$ en utilisant l'équation (3.36).

4. Calculer $\bar{C}_j^*$ à l'aide du théorème 3.9.

5. Calculer l'ordonnancement sans blocage S_{jk} en utilisant le théorème 3.10 ou 3.11.

Nous remarquons que la résolution des problèmes de décision est due à la méthode Programmation Dynamique en avant. Dans le calcul de $F(k+1, \tau)$ à l'aide de l'équation (3.36), il y a beaucoup de calcul redondant. Cela peut être amélioré par des astuces algorithmiques. Finalement, lorsque cet algorithme est appliqué à un système simple, en vertu du théorème 3.1, les indicateurs $B(k, \tau)$ ne sont pas nécessaires et, en conséquence, l'équation 3.36 peut être simplifiée.

III.4.3 Une méthode heuristique fondée sur les systèmes simples

Au vu des résultats du paragraphe III.4.2, les méthodes d'ordonnancement sont plus simples et plus efficaces pour les systèmes simples car, grâce au théorème 3.1, nous n'avons pas besoin de vérifier explicitement l'existence de blocage.

Du paragraphe II.3, nous savons que (i) tout Système à Ressource Unitaire (SRU) peut être transformé en un système canonique en insérant des opérations ZD et (ii) un système simple est un système canonique dont les temps opératoires sont positifs. Ceci nous conduit naturellement à une méthode qui construit l'ordonnancement sans blocage d'un SRU général en passant par un système simple.

Plus précisément, considérons un SRU composé d'un ensemble de travaux $\{J_1, J_2, ..., J_j, ..., J_n\}$ avec $J_j = \{(R_{j1}, p_{j1}), (R_{j2}, p_{j2}), ..., (R_{jNj}, p_{jNj})\}$ et $p_{jk} > 0$. Sa forme canonique est un SRU composé des travaux $\{J_1^c, J_2^c, ..., J_n^c\}$ tels que

$$J_j^c = \{(R_{j1}^c, p_{j1}^c), (R_{j2}^c, p_{j2}^c), ..., (R_{jN_j^c}^c, p_{jN_j^c}^c)\}$$

où

$$N_j^c = 2N_j - 1$$

$$(R_{j,2k-1}^c, p_{j,2k-1}^c) = (R_{jk}, p_{jk}), \qquad \forall 1 \le k \le N_j$$

$$(R_{j,2k}^c, p_{j,2k}^c) = (R_{jk} \cup R_{j,k+1}, 0), \qquad \forall 1 \le k \le N_j.$$

Nous définissons également un système simple, que nous appelons forme canonique modifiée du SRU, en remplaçant les opérations ZD de la forme canonique par des opérations à délai unitaire. Plus précisément, la forme canonique modifiée est un système composé des travaux $\{J_1^m, J_2^m, ..., J_n^m\}$ tels que $N_j^m = N_j^c = 2N_j - 1$, $R_{jk}^m = R_{jk}^c$, $p_{jk}^m = \max\{p_{jk}^c, 1\}$, $\forall$ (j,k).

La nouvelle méthode construit l'ordonnancement du SRU en quatre étapes: (i) calculer la forme canonique modifiée; (ii) ordonnancer la forme canonique modifiée; (iii) extraire les séquences d'entrée π_r dans les ressources pour le SRU à partir forme canonique modifiée et (iv) calculer l'ordonnancement du SRU avec les séquences π_r à l'aide du théorème 3.3. Pour résumer :

Algorithme 3.7 (Méthode d'ordonnancement à l'aide des systèmes simples)

1. Déterminer la forme canonique modifiée du SRU.

2. Ordonnancer cette forme canonique modifiée à l'aide de l'algorithme 3.4 du recuit simulé sans vérification de blocage.

3. Extraire les séquences d'entrée π_r dans les ressources pour le SRU à partir de l'ordonnancement obtenu dans l'étape 2.

4. Calculer les dates de début des opérations du SRU à l'aide du théorème 3.3 avec les séquences d'entrée π_r.

Nous remarquons que les ordonnancements de la forme canonique modifiée sont également admissibles pour les SRU. Mais l'utilisation du théorème 3.3 permet l'élimination des temps morts et conduit à un meilleur ordonnancement.

III.5 Extensions

Dans ce paragraphe, nous allons aborder l'extension des résultats des paragraphes III.2-III.4 où nous avons fait l'hypothèse des temps opératoires positifs, c'est-à-dire $p_{jk} > 0$. Nous allons montrer comment prendre en compte les opérations ZD. Mais nous allons également aborder l'extension des méthodes d'ordonnancement présentées dans ce chapitre pour optimiser d'autres critères et pour prendre en compte des produits de gammes non linéaires avec opération d'assemblage et de désassemblage.

III.5.1 Prise en compte des opérations ZD

Nous examinons successivement les méthodes d'ordonnancement du paragraphe III.4. Considérons d'abord la méthode conservatrice (Heuristique 1) qui, lors de l'ordonnancement d'un travail, n'essaie pas d'utiliser les temps morts laissés par les travaux déjà ordonnancés. Comme constaté dans le paragraphe III.4.1.1, cette méthode conservatrice évite l'attente circulaire puisque les ressources sont affectées aux travaux selon le même ordre des travaux. Par conséquent, elle reste valable pour tout SRU avec ou sans opération ZD.

Par contre, l'Heuristique 2 du paragraphe III.4.2.2, qui cherche à ordonnancer chaque travail de manière optimale, ne s'applique pas directement à l'ordonnancement d'un SRU avec opération ZD. Il existe pourtant une exception pour le cas où les opérations ZD sont des opérations ZRZD, c'est-à-dire $R_{jk} = \phi$ et $p_{jk} = 0$. Dans ce cas, il est aisé de montrer qu'il n'y a pas de blocage au début d'une opération ZRZD (j,k) et au début de l'opération suivante (j, k+1). La méthode par recherche exhaustive du paragraphe III.4.3.2 et la méthode Programmation Dynamique du paragraphe III.4.3.3 peuvent être aisément adaptées pour prendre en compte ces opérations ZRZD.

Pour un SRU général, avec opération ZD, nous utilisons la méthode heuristique fondée sur les systèmes simples pour trouver un ordonnancement sans blocage. Nous notons que la forme canonique modifiée peut être définie de la même manière pour un SRU avec ou sans opération ZD. La seule différence est que les temps opératoires des opérations ZD du SRU sont modifiés alors que les opérations d'un SRU sans opération ZD restent identiques. L'Algorithme 3.7 peut être utilisé sans modification car le théorème 3.3, permettant la détermination des dates de début partant des séquences d'entrée, reste vrai pour un SRU avec l'opération ZD.

III.5.2 Prise en compte d'autres critères

Dans la pratique, la minimisation du makespan n'est pas toujours un critère pertinent. De plus en plus, le souci d'un industriel est le respect de délai et la réduction des en-cours. Il est alors nécessaire de considérer d'autres formes de critères.

Dans notre travail, grâce à la souplesse des méthodes d'ordonnancement proposées, il est aisé de prendre en compte d'autres critères réguliers comme nous le ferons dans le paragraphe III.6 pour l'exemple de Ramaswany et Joshi (86). Rappelons qu'un critère $f(\{S_{jk}\})$ d'un problème d'ordonnancement, où $\{S_{jk}\}$ est un ordonnancement, est dit régulier si pour deux solutions $\{S_{jk}\}$ et $\{S'_{jk}\}$,

$$S'_{jk} \leq S_{jk}, \forall(j,k) \Rightarrow f(\{S'_{jk}\}) \leq f(\{S_{jk}\}).$$

Les critères suivants sont réguliers :

- la durée totale : $C_{max} = Max_j \{C_j\}$,
- la somme de retard : $\sum T_j$,
- la somme d'en-cours : $\sum C_j$,

où $C_j = S_{jNj} + p_{jNj}$, $T_j = \max \{0, C_j - d_j\}$ et d_j est la date échue du travail J_j. Par contre, la minimisation de la somme de retard et d'avance, c'est-à-dire $\sum |C_j - d_j|$, n'est pas un critère régulier. La particularité d'un critère régulier est que l'ordonnancement optimal doit être calé à gauche, c'est-à-dire que les opérations doivent commencer au plus tôt tout en respectant les séquences d'entrée dans les ressources. De cette remarque, il semble naturel que, dans chaque étape d'une méthode gloutonne qui ordonnance les travaux de manière séquentielle, le travail en cours de considération doive terminer au plus tôt. Par conséquent, les méthodes heuristiques (Heuristique 1 et 2) du paragraphe III.4.2 peuvent être appliquées sans considération de la forme exacte du critère que la méthode du recuit simulé doit prendre en compte. La méthode du recuit simulé est l'algorithme 3.4 pour l'optimisation du makespan en remplaçant f(S) par la forme exacte de la valeur de critère de l'ordonnancement obtenu par l'Heuristique 1 ou par l'Heuristique 2 suivant l'ordre des travaux.

III.5.3 Gammes non linéaires

Dans cette thèse, nous nous intéressons essentiellement à l'ordonnancement des travaux avec gammes linéaires, c'est-à-dire que chaque travail est une suite d'opérations. Ceci

exclut évidemment les opérations d'assemblage où une opération peut avoir plusieurs prédécesseurs et/ou plusieurs successeurs.

Il n'est pas difficile de montrer que les résultats sur la représentation et la vérification de blocage du paragraphe III.3 s'étendent au SRU avec gammes non-linéaires. Il est bien évident que chaque travail n'est plus représenté par un chemin élémentaire mais par un RdP acyclique. La méthode conservatrice (Heuristique 1) du paragraphe III.4.2 s'applique en ordonnançant les opérations d'un travail dans un ordre correspondant à sa gamme.

Quant à l'heuristique 2, la principale difficulté réside dans la résolution du problème d'ordonnancement à un travail. Pour la méthode par recherche exhaustive, on cherche pour chaque opération une fenêtre de temps $W_k = (a_k, b_k)$ telle que les contraintes de précédence sont vérifiées et l'ordonnancement est sans blocage. Les contraintes (3.15) - (3.17) deviennent alors :

$$a_k \leq S_{jk} \leq b_k - p_{jk}, \forall k = 1, 2, ..., N_j$$

$$S_{jk} \geq S_{jl} + p_{jl}, \forall (j,l) \text{ précède } (j,k)$$

$$S_{jk} \leq b_l, \forall (j,l) \text{ précède } (j,k).$$

Le théorème 3.6 peut être démontré de manière analogue. Par contre, la vérification de blocage est plus délicate en présence d'opérations de désassemblage car le marquage M_k, c'est-à-dire l'état du système à l'instant a_k, en supposant que l'opération (j, k-1) en cours, n'est plus unique. Cette même difficulté rend la généralisation de la méthode Programmation Dynamique difficile car il n'est pas aisé de définir les indicateurs de blocage $B(k,\tau)$. Une étude plus approfondie sur l'extension de l'heuristique 2 dépasse le cadre de cette thèse et est un sujet de recherche future. Un autre sujet de recherche future intéressant est la prise en compte de la flexibilité des ressources et des gammes de fabrication.

III.6 Résultats numériques

Dans ce paragraphe, nous présentons six exemples de Systèmes à Ressources Unitaires et les ordonnancements obtenus à l'aide de l'Heuristique 1, de l'Heuristique 2 et de la méthode Programmation Dynamique. Le premier exemple est dû à Ramaswany et Joshi (96) et les autres exemples ont été générés de façon aléatoire.

III.6.1 Exemple 1

Cet exemple est dû à Ramaswany et Joshi (96), comme nous l'avons mentionné. Il s'agit d'une cellule composée de trois machines M_1, M_2, M_3, un robot de manutention T et quatre travaux J_1, J_2, J_3, J_4. Chaque travail nécessite sept opérations et chaque opération demande une seule ressource. Les temps opératoires et les opérations des travaux sont donnés dans la table III.2.

Opération	J_1	J_2	J_3	J_4
1	T,5	T,6	T,5	T,4
2	M_1,40	M_1,212	M_2,45	M_3,55
3	T,3	T,7	T,3	T,3
4	M_2,100	M_2,73	M_1,65	M_2,65
5	T,5	T,4	T,6	T,5
6	M_3,36	M_3,32	M_3,98	M_1,35
7	T,4	T,4	T,4	T,5

Table III.2 : Travaux de l'exemple de Ramaswany et Joshi (96)

Minimisation du makespan

Les deux heuristiques, 1 et 2, et la méthode Programmation Dynamique sont utilisées pour trouver l'ordonnancement de cet exemple. Nous avons utilisé la méthode du recuit simulé pour optimiser l'ordre des travaux. Nous avons fait plusieurs essais avec différentes valeurs des paramètres T_0 et r, utilisés dans le recuit simulé.

Les charges des ressources sont données dans la table III. 3. L'ordonnancement obtenu en utilisant l'Heuristique 1 est donné dans la table III.4. Les résultats obtenus en utilisant l'Heuristique 2 et la méthode Programmation Dynamique ont conduit au même ordonnancement et il est donné dans la tables III.5 et le diagramme de Gantt est donné dans la figure III.8.

Ressource	Charge
M_1	352
M_2	283
M_3	221
T	74

Table III.3 : Les charges des ressources de l'exemple de Ramaswany et Joshi (96)

	J_1	J_2	J_3	J_4
Opération	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
1	1, 5	194, 199	533, 537	759, 762
2	6, 45	200, 411	538, 582	763, 817
3	46, 48	412, 418	583, 585	818, 820
4	49, 148	419, 491	586, 650	821, 885
5	149,153	492, 495	651, 656	886, 890
6	154,189	496, 527	657, 754	891, 925
7	190,193	528, 532	755, 758	926, 930
Makespan = 930				
Temps de CPU = 1.10^{-4} secondes				

Table III.4 : Ordonnancement de l'exemple de Ramaswany et Joshi (96) par l'Heuristique 1

	J_1	J_2	J_3	J_4
Opération	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
1	178, 182	230, 235	1, 5	6, 9
2	183, 226	236, 447	6, 50	10, 64
3	227, 229	448, 454	51, 53	65, 67
4	230, 329	455, 527	54, 118	68, 132
5	330, 334	528, 531	119, 124	133, 137
6	335, 370	532, 563	125, 222	138, 172
7	371, 374	564, 568	223, 226	173, 177
Makespan = 568				
Temps de CPU = 31 secondes				

Table III.5 : Ordonnancement de l'exemple de Ramaswany et Joshi (96) par l'Heuristique 2 et par la méthode Programmation Dynamique

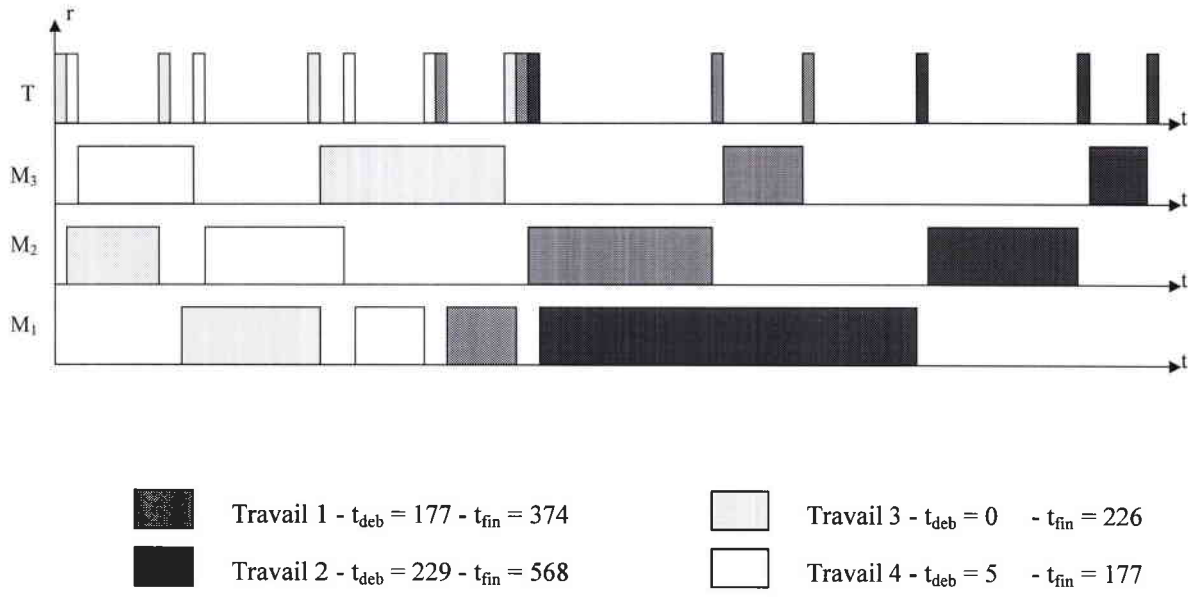

Figure III.8 : Diagramme de Gantt de l'ordonnancement donné dans la table III.5

Le critère utilisé pour obtenir les ordonnancements donnés dans les tables III.4 et III.5 est la minimisation du makespan. Nous avons obtenu les valeurs du makespan 930, 568 et 568, à l'aide de l'Heuristique 1, de l'Heuristique 2 et de la méthode Programmation Dynamique, respectivement. Ramaswany et Joshi (96) ont utilisé le même critère et une approche par programmation linéaire en nombres mixtes. Ils ont obtenu la valeur (optimale) 560 pour le makespan.

Une comparaison entre le résultat obtenu par Ramaswany et Joshi et le résultat obtenu par l'heuristique 2 et la méthode Programmation Dynamique, montre que nous avons obtenu un ordonnancement proche de l'optimum. De plus, nous remarquons que les temps de CPU sont faibles et ils sont donnés dans les tables III.4 et III.5.

Minimisation de la somme de retards et de la somme d'en-cours

Nous considérons deux autres critères : la minimisation de la somme de retards et la minimisation de la somme d'en-cours, c'est-à-dire :

$$\text{Min} \sum_j \max (0, C_j - d_j) ,$$

$$\text{Min} \sum_j C_j ,$$

où C_j est la date de fin du travail j et d_j est la date échue du travail j.

La table III.6 donne les résultats obtenus à l'aide de l'heuristique 2 et de la méthode Programmation Dynamique avec le critère de la minimisation de la somme de retards et pour des dates échues différentes.

Pour la minimisation de la somme d'en-cours, les deux méthodes donnent un ordonnancement de la valeur de critère de 1345 et du makespan 568.

Essai	**Date échue de chaque travail**				**Somme de retards**	**Makespan**
	d_1	d_2	d_3	d_4		
1	200	400	200	200	368	568
2	400	400	400	400	168	568
3	500	500	500	500	68	568
4	200	200	200	200	568	568
5	1000	200	200	200	342	627

Table III.6 : Ordonnancement pour la minimisation de la somme de retards

III.6.2 D'autres exemples

Ces exemples ont été générés de façon aléatoire. Chaque exemple est composé de neuf ou dix travaux et de huit, neuf ou dix types de ressources. Chaque travail nécessite entre trois à cinq opérations et chaque opération demande entre un à trois types de ressources. Les temps opératoires et les opérations des travaux sont donnés dans les tables III.7, III.10, III.13, III.16 et III.19. Les charges des ressources sont données dans les tables III.8, III.11, III.14, III.17 et III.20. Les ordonnancements obtenus à l'aide de l'Heuristique 2 sont donnés dans les tables III.9, III.12, III.15, III.18 et III.21.

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	$R_1R_2R_3$, 1	$R_1R_3R_4$, 3	R_5, 9	R_6, 8	R_4R_7, 7
J_2	R_2R_4, 4	R_5, 7	R_7, 10	R_4R_8, 1	
J_3	R_1R_5, 9	$R_3R_4R_5$, 5	R_6R_7, 8		
J_4	R_7R_9, 10	R_1R_3, 8	R_4, 5		
J_5	R_2, 9	$R_1R_4R_6$, 4	R_9, 4		
J_6	R_1R_2, 3	R_4R_7, 4	R_1R_4, 3		
J_7	$R_1R_6R_9$, 10	$R_2R_3R_6$, 9	R_6R_8, 10		
J_8	R_4, 7	R_7, 8	R_2, 4	R_3, 3	
J_9	R_1, 6	$R_3R_7R_9$, 7	$R_1R_4R_{10}$, 9		
J_{10}	R_2R_9, 7	R_5R_7, 1	R_4R_8, 6	R_5, 3	

Table III.7 : Travaux de l'exemple 2

Ressource	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
Charge	56	37	36	58	34	49	55	17	38	9

Table III.8 : Les charges des ressources de l'exemple 2

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 1	2, 4	5, 13	14, 23	24, 30
J_2	12, 15	16, 30	31, 40	41, 41	
J_3	70, 78	79, 83	84, 91		
J_4	1, 10	11, 18	19, 23		
J_5	57, 65	66, 69	70, 73		
J_6	79, 91	92, 95	96, 98		
J_7	24, 33	34, 42	43, 52		
J_8	5, 11	12, 19	20, 23	24, 26	
J_9	34, 42	43, 49	50, 58		
J_{10}	50, 56	57, 58	59, 64	65, 67	
Makespan = 98 (Remarque : le makespan obtenu avec l'Heuristique 1 est 151)					

Table III.9 : Ordonnancement de l'exemple 2 par l'Heuristique 2

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	R_9, 7	R_3R_6, 2	R_1,8	$R_1R_6R_8$, 9	
J_2	R_6, 10	$R_2R_3R_4$, 3	R_5R_6, 7		
J_3	R_5, 10	R_2R_8, 1	$R_3R_5R_7$, 4	$R_3R_4R_7$,10	
J_4	$R_2R_4R_9$, 9	$R_2R_4R_7$, 9	R_8, 6	R_3, 9	
J_5	R_2R_6, 9	R_1, 3	R_2R_7, 3	R_2R_9, 4	R_1R_8, 3
J_6	$R_5R_7R_8$, 10	$R_6R_7R_8$, 9	R_1R_4, 10		
J_7	R_8, 7	R_9, 8	R_6, 4	R_9, 3	
J_8	R_8, 6	$R_1R_6R_8$, 7	$R_1R_4R_7$, 9		
J_9	R_2R_9, 7	R_4, 7	R_6R_8, 3	R_3, 5	
J_{10}	R_1R_7, 9	$R_3R_4R_8$, 10	$R_1R_3R_4$, 5		

Table III.10 : Travaux de l'exemple 3

Ressource	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
Charge	63	45	48	72	31	60	63	71	38

Table III.11 : Les charges des ressources de l'exemple 3

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	57, 83	84, 85	86, 93	94, 102	
J_2	48, 57	58, 60	61, 67		
J_3	11, 20	21, 21	22, 29	30, 39	
J_4	84, 92	93, 102	103, 108	109, 117	
J_5	22, 30	31, 39	40, 43	44, 56	57, 59
J_6	1, 10	11, 19	20, 29		
J_7	22, 28	29, 36	37, 40	41, 43	
J_8	29, 40	41, 47	48, 56		
J_9	103, 109	110, 116	117, 119	120, 124	
J_{10}	60, 68	69, 78	79, 83		
Makespan = 124 (Remarque : le makespan obtenu avec l'Heuristique 1 est 195)					

Table III.12 : Ordonnancement de l'exemple 3 par l'Heuristique 2

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	R_9, 7	R_3R_6, 2	R_1,8	$R_1R_6R_8$, 9	
J_2	R_6, 10	$R_2R_3R_4$, 3	R_5R_6, 7		
J_3	R_5, 10	R_2R_8, 1	$R_3R_5R_7$, 4	$R_3R_4R_7$,10	
J_4	$R_2R_4R_9$, 9	$R_2R_4R_7$, 9	R_8, 6	R_3, 9	
J_5	R_2R_6, 9	R_1, 3	R_2R_7, 3	R_2R_9, 4	R_1R_8, 3
J_6	$R_5R_7R_8$, 10	$R_6R_7R_8$, 9	R_1R_4, 10		
J_7	R_8, 7	R_9, 8	R_6, 4	R_9, 3	
J_8	R_8, 6	$R_1R_6R_8$, 7	$R_1R_4R_7$, 9		
J_9	R_2R_9, 7	R_4, 7	R_6R_8, 3	R_3, 5	

Table III.13 : Travaux de l'exemple 4

Ressource	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
Charge	49	45	33	57	31	60	54	61	38

Table III.14 : Les charges des ressources de l'exemple 4

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 13	14, 22	23, 30	31, 39	
J_2	52, 76	77, 79	80, 86		
J_3	1, 13	14, 22	23, 26	27, 36	
J_4	59, 67	68, 76	77, 82	83, 91	
J_5	40, 48	49, 51	52, 54	55, 58	59, 61
J_6	87, 96	97, 105	106, 115		
J_7	62, 68	69, 86	87, 90	91, 93	
J_8	1, 6	7, 13	14, 22		
J_9	23, 36	37, 48	49, 51	52, 56	
Makespan = 115 (Remarque : le makespan obtenu avec l'Heuristique 1 est 180)					

Table III.15 : Ordonnancement de l'exemple 4 par l'Heuristique 2

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	$R_1R_2R_8$, 4	R_4, 6	R_1, 8		
J_2	R_1R_7, 10	R_8, 8	R_2R_8, 7	R_6 R_7, 4	R_3R_8, 5
J_3	R_1, 6	$R_1R_3R_8$, 3	$R_4R_7R_8$, 9		
J_4	$R_4R_5R_7$, 7	R_1R_5, 10	R_6R_7, 8		
J_5	$R_1R_5R_7$, 3	$R_4R_7R_8$, 4	R_2, 4	R_4, 1	
J_6	R_4, 6	R_8, 3	R_6, 2		
J_7	R_2R_8, 1	$R_4R_6R_8$, 9	R_3R_8, 7	R_2R_4, 4	R_6, 9
J_8	R_2, 4	R_3, 3	R_5, 2	R_6, 9	R_1R_8, 9
J_9	R_1R_7, 9	R_3R_6, 9	$R_1R_4R_5$, 2	$R_1R_3R_4$, 7	R_1R_6, 9
J_{10}	$R_3R_4R_7$, 8	R_3, 9	R_7, 6	R_8, 1	$R_2R_3R_7$, 4

Table III.16 : Travaux de l'exemple 5

Ressource	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
Charge	80	28	55	63	24	59	72	70

Table III.17 : Les charges des ressources de l'exemple 5

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	83, 88	89, 100	101, 108		
J_2	1, 17	18, 25	26, 32	33, 36	37, 41
J_3	109, 114	115, 117	118, 126		
J_4	22, 28	29, 38	39, 46		
J_5	56, 58	59, 62	63, 73	74, 74	
J_6	29, 41	42, 46	47, 48		
J_7	1, 1	2, 10	11, 17	18, 21	22, 30
J_8	74, 77	78, 80	81, 82	83, 91	92, 100
J_9	47, 55	56, 64	65, 66	67, 73	74, 82
J_{10}	81, 88	89, 97	98, 103	104, 104	105, 108
Makespan = 126 (Remarque : le makespan obtenu avec l'Heuristique 1 est égal a 214)					

Table III.18 : Ordonnancement de l'exemple 5 par l'Heuristique 2

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	$R_1R_2R_8$, 4	R_4, 6	R_1, 8		
J_2	R_1R_7, 10	R_8, 8	R_2R_8, 7	R_6R_7, 4	R_3R_8, 5
J_3	R_1, 6	$R_1R_3R_8$, 3	$R_4R_7R_8$, 9		
J_4	$R_4R_5R_7$, 7	R_1R_5, 10	R_6R_7, 8		
J_5	$R_1R_5R_7$, 3	$R_4R_7R_8$, 4	R_2, 4	R_4, 1	
J_6	R_4, 6	R_8, 3	R_6, 2		
J_7	R_2R_8, 1	$R_4R_6R_8$, 9	R_3R_8, 7	R_2R_4, 4	R_6, 9
J_8	R_2, 4	R_3, 3	R_5, 2	R_6, 9	R_1R_8, 9
J_9	R_1R_7, 9	R_3R_6, 9	$R_1R_4R_5$, 2	$R_1R_3R_4$, 7	R_1R_6, 9

Table III.19 : Travaux de l'exemple 6

Ressource	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
Charge	80	24	34	55	24	59	54	69

Table III.20 : Les charges des ressources de l'exemple 6

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	84, 87	88, 93	94, 101		
J_2	37, 46	47, 54	55, 61	62, 65	66, 70
J_3	66, 71	72, 74	75, 83		
J_4	47, 53	54, 65	66, 73		
J_5	10, 12	13, 16	17, 27	28, 28	
J_6	29, 34	35, 37	38, 39		
J_7	88, 93	94, 102	103, 109	110, 113	114, 122
J_8	62, 74	75, 77	78, 102	103, 111	112, 120
J_9	1, 9	10, 18	19, 20	21, 27	28, 36
Makespan = 122					
(Remarque : le makespan obtenu avec l'Heuristique 1 est 195)					

Table III.21 : Ordonnancement de l'exemple 6 par l'Heuristique 2

III.7 Conclusion

Dans ce chapitre, nous nous sommes limités aux Systèmes à Ressources Unitaires, c'est-à-dire aux job-shops MRB dans lesquels les ressources sont toutes différentes. Nous avons formulé le problème d'ordonnancement et nous avons montré que, pour les systèmes simples, les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage. Les RdP ont été utilisés pour modéliser les SRU et l'ordonnancement, ce qui a permis la vérification de blocage pour un état donné et la vérification de la faisabilité d'un ordonnancement donné. Dans la suite, nous avons proposé deux méthodes heuristiques gloutonnes pour ordonnancer un SRU afin de minimiser des critères réguliers, tout en évitant les blocages. En constatant que ces méthodes sont plus efficaces pour les systèmes simples, car la vérification de blocage n'est pas nécessaire, nous avons proposé une autre méthode d'ordonnancement d'un SRU général en passant par un système simple "équivalent". Les résultats numériques sont encourageants.

Ce chapitre a également conduit à des sujets de recherche futures : la prise en compte des gammes non-linéaires de production (opérations d'assemblage / désassemblage) et la prise en compte de la flexibilité des ressources et des gammes.

Chapitre IV

Ordonnancement des Systèmes à Ressources Multiples

IV.1 Introduction

Nous abordons, dans ce chapitre, l'ordonnancement des job-shops MRB généraux, c'est-à-dire des Systèmes à Ressources Multiples avec opérations MRMU (*Multi-Resource Multi-Units*). Un cas intermédiaire entre les job-shops MRB généraux et les Systèmes à Ressources Unitaires étudiés dans le chapitre précédent concerne les Systèmes à Ressources Multiples avec des opérations SU (*Single Unit*) qui nécessitent, au plus, une unité de chaque type de ressource à la fois. Ceux-ci comprennent comme cas particulier les flow-shops hybrides et les job-shops hybrides qui sont des flow-shops ou des job-shops avec machines identiques. Ils incluent également comme cas particuliers les job-shops avec AGV, comme les moyens de transport, systèmes que nous considérons plus en détail dans le chapitre V. Les opérations MU (*Multi-Units*), qui peuvent avoir besoin de plusieurs unités du même type de ressource, permettent la prise en compte des contraintes de stockage dans le cas des produits de différentes tailles, des produits nécessitant plusieurs outils du même type, etc. On peut dire que les opérations MRMU augmentent, de manière significative, la puissance de modélisation et la généralité des job-shops MRB.

Malheureusement, le gain en généralité est accompagné d'une perte en solvabilité du problème d'ordonnancement. Comme nous allons le montrer dans ce chapitre, la vérification de blocage devient beaucoup plus difficile. Ceci peut être considéré comme une parallèle de l'extrême difficulté de la vérification de blocage d'un RdP généralisé avec poids par rapport à la relative simplicité d'un RdP ordinaire de poids unitaire. C'est pour cette raison que les systèmes simples, pour lesquels nous montrerons à nouveau que les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage et jouent un rôle encore plus important que dans le cas d'un SRU. Une autre nouveauté, par rapport aux SRU, est le problème d'affectation des ressources aux travaux à cause des ressources identiques. On peut alors dire que l'ordonnancement d'un job-shop MRB général est bien plus compliqué que l'ordonnancement d'un SRU.

Ce chapitre est organisé de la manière suivante. Le paragraphe IV.2 rappelle la formulation du problème d'ordonnancement d'un job-shop MRB général. Mais le résultat le plus important concerne les systèmes simples pour lesquels nous montrerons que les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage. Le paragraphe IV.3 aborde la représentation d'un job-shop MRB à l'aide des RdP généralisés avec poids. Nous montrerons la difficulté de la vérification de blocage. Nous aborderons également la faisabilité d'un ordonnancement lorsque les séquences d'entrée dans chaque ressource sont données. Le paragraphe IV.4 propose deux heuristiques similaires à celles proposées dans le chapitre III pour les SRU. La méthode conservatrice, Heuristique 1, s'applique à tout job-shop MRB. La méthode la plus élaborée, Heuristique 2, est d'abord développée pour les systèmes simples et ensuite étendue aux job-shops MRB généraux comme nous l'avons fait dans le chapitre III. Le paragraphe IV.5 présente les résultats numériques et le paragraphe IV.6 est une conclusion.

IV.2 Reformulation du problème d'ordonnancement

Nous rappelons d'abord la définition d'un job-shop MRB et son ordonnancement. Un job-shop MRB est composé de m types de ressources R = {1, 2, ..., r, ..., m}. Il y a H_r unités de ressources du type r. Un ensemble de travaux J = {J_1, J_2, ..., J_j, ..., J_n} doit être réalisé. Chaque travail J_j nécessite N_j opérations {O_{j1}, O_{j2}, ..., O_{jk}, ..., O_{jNj}}. Chaque opération nécessite un temps opératoire p_{jk} et la présence simultanée d'un ensemble de ressources ; ce besoin est représenté par un vecteur $\mathbf{h}_{jk}$ dont chaque élément h_{jkr} indique le nombre de ressources r nécessaires. Nous notons également R_{jk} l'ensemble des types de ressources nécessaires pour l'opération (j,k). La propriété "Retenir et Attendre" définit la condition de passage d'une opération à l'opération suivante.

Le problème d'ordonnancement consiste à choisir, pour chaque type de ressource $r \in R$, une séquence π_r d'entrée des opérations nécessitant les ressources r et la date de début S_{jk} pour chaque opération (j,k) afin de minimiser la durée totale de l'ordonnancement et éviter les blocages. Plus précisément, le problème est :

$$\text{Min} \underset{j}{\text{MAX}} \{C_j\},$$

sous les contraintes suivantes :

$$C_j = S_{jNj} + p_{jNj}\ ,\ \forall 1 \le j \le n \tag{4.1}$$

$$S_{jk} + p_{jk} \le S_{jk+1},\ \forall 1 \le j \le n,\ \forall 1 \le k \le N_j \tag{4.2}$$

$$\sum_{(j,k)} h_{jkr}\, \mathbf{1}\{S_{jk} \le t < S_{jk+1}\} \le H_r,\ \forall t\ ,\ \forall\ r \in R \tag{4.3}$$

$$S_{\pi_r[i]} \le S_{\pi_r[i+1]}\ ,\ \forall\ r \in R,\ \forall i \tag{4.4}$$

Les séquences d'entrée π_r ne conduisent pas au blocage (4.5)

où (4.1) définissent la date de fin des travaux, (4.2) sont les contraintes de précédence, (4.3) les contraintes de capacité des ressources, (4.4) assurent la cohérence entre les dates de début et les séquences d'entrée des ressources et (4.5) est la contrainte de l'absence de blocage.

Nous savons maintenant que, pour un job-shop MRB général, les contraintes de précédence (4.2) et de capacité des ressources (4.3) n'impliquent pas l'absence de blocage. Comme dans le cas des SRU, nous montrons qu'il existe une exception importante pour les systèmes simples.

Dans un job-shop MRB général, une opération (j,k) est dite G-opération si $\mathbf{h}_{j,k-1} \leq \mathbf{h}_{jk}$ et elle est dite R-opération si $\mathbf{h}_{jk} \geq \mathbf{h}_{j,k+1}$. Un job-shop MRB est dit système canonique si pour chaque opération (j,k), l'opération (j, k-1) est une R-opération si l'opération (j,k) n'est pas une G-opération. Il est dit système simple s'il est un système canonique et si tous les temps opératoires sont positifs, c'est-à-dire $p_{jk} > 0$. La caractéristique d'un système canonique est que, lors du passage de deux opérations consécutives, soit le travail libère des ressources, soit il nécessite de nouvelles ressources mais jamais les deux à la fois.

Théorème 4.1 : Pour un système simple, un ordonnancement $\{S_{jk}\}$ satisfaisant les contraintes de précédence (4.2) et les contraintes de capacité de ressources (4.3) est sans blocage et donc admissible. Les séquences d'entrée π_r, qui respectent les dates de début et qui, pour les opérations (j,k) avec la même date de début, donnent priorité aux opérations dont (j,k-1) est une R-opération, sont sans blocage.

Preuve par contradiction :

En prenant en compte les caractéristiques des job-shops MRB généraux, la preuve est similaire à celle du théorème 3.1. Supposons qu'il existe un instant t tel que toute opération (j,k) / $S_{jk} < t$ peut commencer à la date S_{jk} tout en respectant la propriété "Retenir et Attendre" et qu'il existe une opération (i,l) / $S_{il} = t$ qui ne peut pas démarrer à l'instant t sans violer la propriété "Retenir et Attendre". Deux cas sont possibles :

- Cas (i) : l'opération (i,l-1) est une R-opération, c'est-à-dire $\mathbf{h}_{i,l-1} \geq \mathbf{h}_{il}$. Puisque les capacités de ressources sont respectées, alors les ressources $\mathbf{h}_{i,l-1}$ sont encore retenues par le travail i à l'instant t-. Selon la propriété "Retenir et Attendre", les ressources $\mathbf{h}_{i,l-1}$ ne pourrons pas être libérées avant le début de l'opération (i,l). Puisque $\mathbf{h}_{i,l-1} \geq \mathbf{h}_{il}$, les ressources nécessaires pour l'opération (i,l) sont disponibles à l'instant t et l'opération (i,l) peut commencer à la date S_{il}, ce qui contredit l'hypothèse de la preuve.

- Cas (ii) : l'opération (i,l) est une G-opération, c'est-à-dire $\mathbf{h}_{i,l-1} \leq \mathbf{h}_{il}$. Puisque (i,l) ne peut pas commencer à l'instant t et que les contraintes de capacité de ressources sont vérifiées, il existe une ressource $r \in R_{il}$ et une opération (j,k) avec $j \neq i$, telle que $r \in R_{jk}$ et $S_{j,k+1} = t$; au moins une unité de la ressource r doit passer du travail j au travail i au début de la période t mais l'opération (j, k+1) ne peut pas commencer à l'instant t non plus (voir figure IV.1).

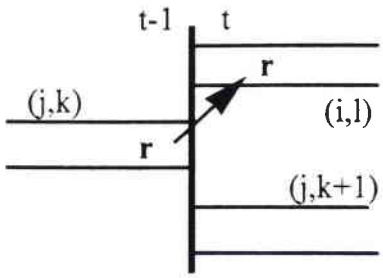

Figure IV.1 : Une situation de blocage

Compte tenu de la propriété d'un système simple, l'opération (j,k) est une R-opération, c'est-à-dire $\mathbf{h}_{jk} \leq \mathbf{h}_{j,k+1}$. Selon le résultat du cas (i), l'opération (j,k+1) peut commencer à la date t et les ressources r nécessaires à l'opération (i,l) peuvent être libérées au début de la période t. Cela prouve, par l'absurde, que toutes les nouvelles ressources nécessaires pour l'opération (i,l) sont libres ou peuvent être libérées au début de la période t. Alors, l'opération (i,l) peut commencer comme prévu à l'instant t. Cela contredit l'hypothèse de la preuve et conclut la preuve.

Q.E.D.

De cette preuve, nous constatons que d'autres séquences π_r sans blocage peuvent être obtenues en respectant simplement les dates de début S_{jk} et la propriété "Retenir et Attendre".

Nous remarquons que chaque type de ressource peut avoir plusieurs unités de ressources identiques. Se pose naturellement la question d'affectation de ces ressources identiques aux opérations.

Plus précisément, soit $\overline{R}_r$ l'ensemble des ressources du type r et $\overline{R}$ l'ensemble des ressources, c'est-à-dire $\overline{R} = \bigcup_{r \in R} \overline{R}_r$. Nous parlerons de la ressource (r,i) pour désigner la $i^{ème}$ ressource du type r. Par conséquent,

$$\overline{R}_r = \{(r,i) / (r,i) \in \overline{R}\}.$$

Le problème d'affectation consiste à déterminer, pour chaque opération (j,k) et pour chaque type de ressource $r \in R_{jk}$, un ensemble de ressources du type r, où $\overline{R}_{jkr} \subseteq \overline{R}_r$, tel que $|\overline{R}_{jkr}| = h_{jkr}$ et que seules les ressources $\overline{R}_{jkr}$ soient utilisées pour l'opération (j,k).

Une affectation permet de transformer un Système à Ressources Multiples en un Système à Ressources Unitaires. Une affectation étant donnée, les ressources peuvent alors

être considérées comme toutes différentes. Dans ce cas, nous pouvons définir un Système à Ressources Unitaires composé des ressources dans $\overline{R}_r$ dont l'opération (j,k) nécessite les ressources dans $\bigcup_{r \in R_{jk}} \overline{R}_{jkr}$.

Théorème 4.2 : Pour tout ordonnancement $\{S_{jk}\}$ respectant les contraintes de capacité des ressources, tel que $S_{jk} < S_{j,k+1}$, $\forall(j,k)$, il existe une affectation $\{\overline{R}_{jkr}\}$ respectant la capacité de chaque ressource $r_i \in \overline{R}$. De plus, il existe une affectation $\{\overline{R}_{jkr}\}$ conduisant à un Système à Ressources Unitaires Simple si le job-shop MRB est un système simple.

Preuve :

Nous démontrons d'abord la première partie du théorème par construction. Pour cela, nous déterminons l'affectation des ressources aux opérations de manière progressive dans l'ordre chronologique. Plus précisément, nous utilisons l'algorithme suivant pour calculer l'affectation :

Algorithme 4.1 (Affectation des ressources)

1. Initialisation : t = 1 et toutes les ressources sont libres.

2. Libération des ressources des opérations terminant à la fin de la période t-1 :
   Pour chaque travail J_j / $\exists$ une opération (j,k) avec $S_{j,k+1} = t$, libérer les ressources actuellement affectées à J_j qui ne sont pas nécessaires pour la nouvelle opération (j, k+1), c'est-à-dire libérer $h_{jkr} - h_{j,k+1,r}$ unités de ressource de $\overline{R}_{jkr}$ pour tout type de ressource r tel que $h_{jkr} > h_{j,k+1,r}$.

3. Obtention de nouvelles ressources pour les opérations commençant au début de t :

   Pour chaque travail J_j / $\exists$ une opération (j,k) avec $S_{j,k+1} = t$, affecter à J_j $h_{j,k+1,r} - h_{jkr}$ nouvelles unités de ressources pour tout type de ressource r tel que $h_{jkr} < h_{j,k+1,r}$.

4. t = t + 1 et aller à l'étape 2.

Dans cet algorithme, au début de chaque nouvelle période t et après l'étape 2, chaque travail J_j garde seulement les ressources déjà affectées à J_j et encore nécessaires dans la période t. Il reste alors suffisamment de ressources libres pour l'étape 3 car, sinon, la somme des ressources de chaque type r déjà affectées aux travaux et encore nécessaires dans la période t, plus les nouvelles ressources r nécessaires pour les opérations démarrant au début de t dépasse H_r, ce qui implique que la contrainte de capacité de ressources r est violée dans la

période t par l'ordonnancement $\{S_{jk}\}$. Cela contredit l'hypothèse du théorème et montre par l'absurde que l'algorithme 4.1 construit une affectation respectant la capacité de chaque ressource $r \in \overline{R}$.

La deuxième partie du théorème est une conséquence directe de l'algorithme 4.1.

Q.E.D.

Théorème 4.3 : Considérons un système simple. Pour tout ordonnancement $\{S_{jk}\}$ satisfaisant les contraintes de capacité des ressources et de précédence, il existe une affectation $\{\overline{R}_{jkr}\}$ telle que $\{S_{jk}\}$ et $\{\overline{R}_{jkr}\}$ définissent un ordonnancement sans blocage.

Preuve :
Triviale au vu des théorèmes 4.2 et 3.1.

Q.E.D.

Pour conclure ce paragraphe, comme dans le cas des SRU, l'ordonnancement d'un système simple se réduit à déterminer les dates de début des opérations S_{jk} afin de :

$$\text{Min } \underset{j}{\text{MAX}} \{C_j\},$$

sous les contraintes (4.1), (4.2) et (4.3).

IV.3 Représentation à l'aide de RdP et vérification de blocage

Nous abordons dans ce paragraphe (i) la représentation par des RdP des job-shops MRB généraux, (ii) la détection de blocage pour un état du système donné, (iii) la représentation des ordonnancements, (iv) la vérification de faisabilité des ordonnancements et (v) la construction d'un ordonnancement optimal connaissant (a) la séquence d'entrée de chaque type de ressource ou (b) l'affectation des ressources aux opérations et la séquence d'entrée dans chaque ressource.

IV.3.1 Modélisation des job-shops MRB à l'aide des RdP

Dans ce paragraphe, nous proposons une méthode systématique pour la modélisation des job-shops MRB. Contrairement à la modélisation des SRU, nous utilisons ici des RdP généralisés avec poids. La méthode se décompose en deux étapes.

La première étape est la représentation du processus de fabrication de chaque travail J_j. Le modèle RdP, est identique à celui d'un SRU, est un chemin élémentaire (voir la figure IV.2). Il contient N_j+2 places dont une place de source et une place de puits représentent respectivement le début et la fin du travail J_j et les autres places représentent les opérations. Des temps sont associés aux places correspondant aux temps opératoires. Les franchissements des transitions sont instantanés. Les places sont également appelées des places-opérations.

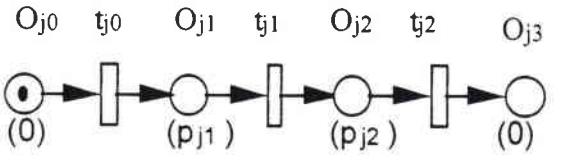

Figure IV.2 : Le modèle RdP d'un travail de deux opérations

La deuxième étape complète le modèle RdP en prenant en compte les ressources nécessaires pour chaque opération. Elle représente chaque ressource $r \in R$ par une place r avec initialement H_r jetons correspondant à la disponibilité de la ressource. Ces nouvelles places sont appelées places de ressources.

Un arc relie une place de ressource r à une transition t_{jk} si de nouvelles ressources r sont nécessaires pour l'opération (j, k+1), c'est-à-dire $h_{jkr} < h_{j.k+1.r}$. Nous associons un poids $(h_{j.k+1.r} - h_{jkr})$ à cet arc.

Un arc relie une transition t_{jk} à une place de ressource r si la fin de l'opération (j, k) libère des ressources r, c'est-à-dire $h_{jkr} > h_{j.k+1.r}$. Le poids associé à cet arc est $(h_{jkr} - h_{j.k+1.r})$. La propriété "Retenir et Attendre" est respectée car les ressources nécessaires à l'opération (j,k) ne peuvent être libérées qu'au franchissement de la transition t_{jk}, c'est-à-dire au début de l'opération (j, k+1).

La figure IV.3 est le modèle RdP final d'un job-shop MRB composé de deux travaux $J_1 = \{(r_1\ r_2, 5), (2r_1\ r_2, 4)\}$ et $J_2 = \{(r_1, 3), (r_2, 4)\}$ où le travail J_1 a besoin d'une part, d'une ressource r_1, d'une ressource r_2 et d'un temps opératoire de 5 unités pour sa première opération et, d'autre part, d'une autre unité de ressource r_1 et de 4 unités de temps pour sa deuxième opération.

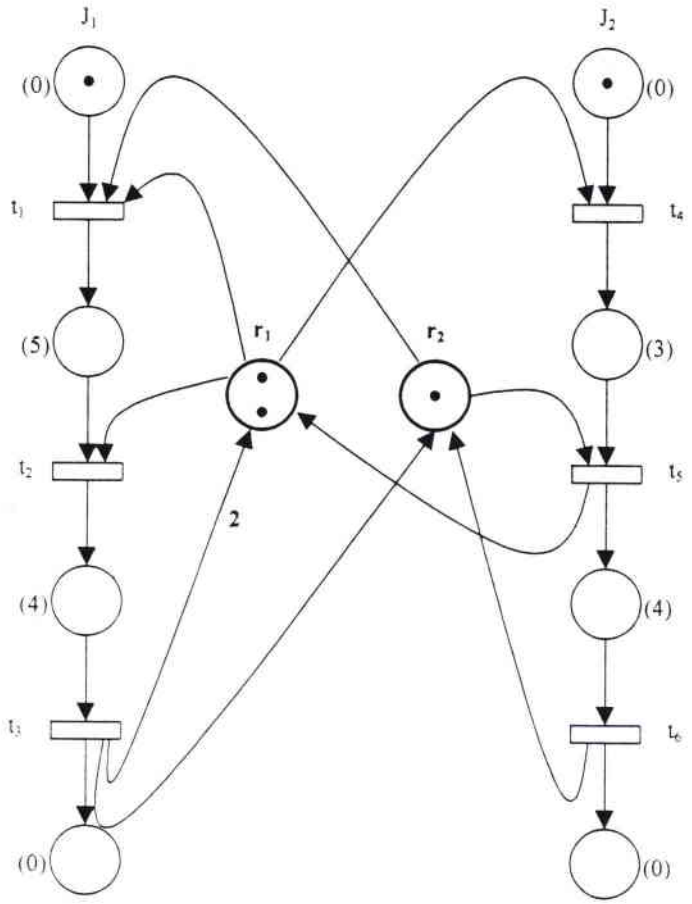

Figure IV.3 : Le modèle RdP d'un job-shop MRB

IV.3.3 La vérification de blocage pour un état donné

Comme dans les SRU, les job-shops MRB généraux sont sujets au blocage et notre objectif est de trouver l'ordonnancement optimal sans blocage. Compte tenu de la nature des méthodes d'ordonnancement proposées dans cette thèse, nous avons besoin en particulier de la détection de blocage dans un ordonnancement respectant les contraintes de précédence et de capacité des ressources et, surtout, de la vérification de blocage pour un état donné. L'objectif est de nous assurer de l'absence de travaux ne pouvant pas avancer à cause de la propriété "Retenir et Attendre".

En tenant compte du modèle RdP, l'ordonnancement respectant les contraintes de précédence et de capacité de ressources se traduit par les dates de franchissement des transitions S_t. $\forall t \in T$ où T est l'ensemble des transitions. Le problème de la vérification de blocage consiste à déterminer, pour tout instant $\tau/S_t = \tau$, pour au moins une $t \in T$, une

séquence de franchissement des transitions t y que $S_t = \tau$. Si une telle séquence n'existe pas, nous disons que l'ordonnancement $\{S_t\}$ est avec blocage à l'instant τ.

Contrairement au cas des SRU avec temps opératoires positifs, il n'est pas possible d'utiliser la notion de siphon pour la vérification de blocage. La raison principale vient du fait que les transitions à franchir au même moment ne sont pas en conflit dans le cas d'un SRU avec $p_{jk} > 0$, alors qu'elles le sont dans un job-shop MRB général. Par exemple, pour le système de la figure IV.3, l'ordonnancement $\{S_{t_1} = 0, S_{t_2} = 5, S_{t_3} = 9, S_{t_4} = 0, S_{t_5} = 5, S_{t_6} = 9\}$ respecte les contraintes de précédence et de capacité des ressources et, à la date t = 0, les transitions à franchir t_1 et t_4 sont en conflit structurel.

De plus, dans le cas d'un SRU, l'ordre de franchissement des transitions n'est pas important à condition de respecter les règles de franchissement. Ceci est malheureusement faux dans le cas d'un job-shop MRB général. Un exemple est donné dans la figure IV.4 où les transitions t_1, t_2, t_3 sont à franchir au même moment. Dans cet exemple, franchir t_1 d'abord conduit au blocage et franchir t_3 d'abord permet d'éviter le blocage.

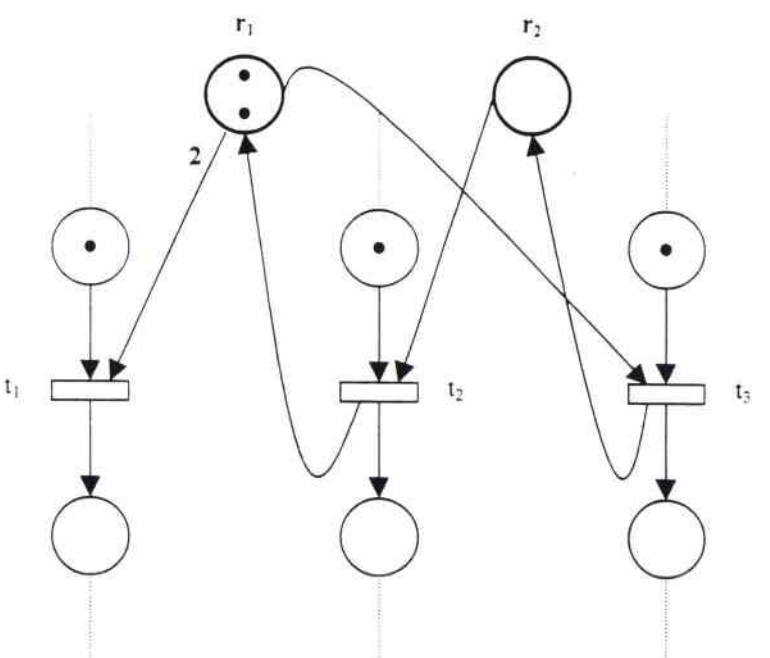

Figure IV.4 : Un état d'un job-shop MRB

Pour toutes ces raisons, nous proposons une méthode de vérification exhaustive en considérant toutes les permutations possibles des transitions à franchir au même moment. Cette méthode est, bien entendu, combinatoire alors que la méthode fondée sur la notion de siphon est de complexité linéaire. Aussi, pour cette raison, nous déconseillons l'utilisation de cette méthode si le nombre de transitions à franchir au même moment peut être important. Dans ce cas, les méthodes fondées sur les systèmes simples sont alors particulièrement attrayantes.

Théorème 4.4 : Etant donné un ordonnancement $\{S_t, \forall t \in T\}$ satisfaisant les contraintes de précédence et de capacité des ressources, le système est sans blocage à l'instant τ où $\exists t \in T / S_t = \tau$ si, et seulement si, il existe une séquence franchissable dans le RdP $(N, M_{\tau-})$ comprenant toutes les transitions à franchir à l'instant τ, où $M_{\tau-}$ est le marquage du RdP à la fin de la période τ-1.

La preuve est triviale et donc omise.

IV.3.4 Faisabilité d'un ordonnancement donné

Nous étudions dans ce paragraphe, la vérification de la faisabilité d'un ordonnancement donné pour les job-shops MRB généraux. Nous considérons deux représentations d'un ordonnancement : (i) la séquence d'entrée dans chaque type de ressource est connue; (ii) l'affectation des ressources aux opérations $\{\overline{R}_{jkr}\}$ et la séquence d'entrée dans chaque ressource sont connues.

IV.3.4.1 Cas où la séquence d'entrée dans chaque type de ressource est connue

Nous supposons ici que les séquences π_r d'entrée dans les différents types de ressources sont données. Rappelons que, dans le cas d'un SRU, chaque ressource r est utilisée par toutes les opérations (j,k) avec $r \in R_{jk}$ et que nous avons pu représenter un ordonnancement à l'aide d'un RdP sans conflit structurel, c'est-à-dire un graphe d'événements. Ici, chaque type de ressource comporte plusieurs unités et nous ne connaissons pas, a priori, l'affectation des ressources aux opérations. C'est pour cette raison que le modèle RdP avec les séquences d'entrée, proposé ci-dessous, possède des conflits structurels.

La prise en compte des séquences d'entrée dans le modèle RdP peut être décrite comme suit : soit π_r = O(r,1), O(r,2), ..., O(r,N(r)) la séquence d'entrée dans les ressources du type r où N(r) est le nombre d'opérations nécessitant les ressources r, le nouveau RdP est obtenu en ajoutant pour chaque séquence π_r, N(r) - 1 places p(r,1), p(r,2), ..., p(r,N(r)) (voir figure IV.5). Chaque place p(r,k) relie la transition d'entrée de la place O(k,r) et la transition d'entrée de la place O(r, k+1). Ces nouvelles places sont initialement vides et elles ne sont pas temporisées.

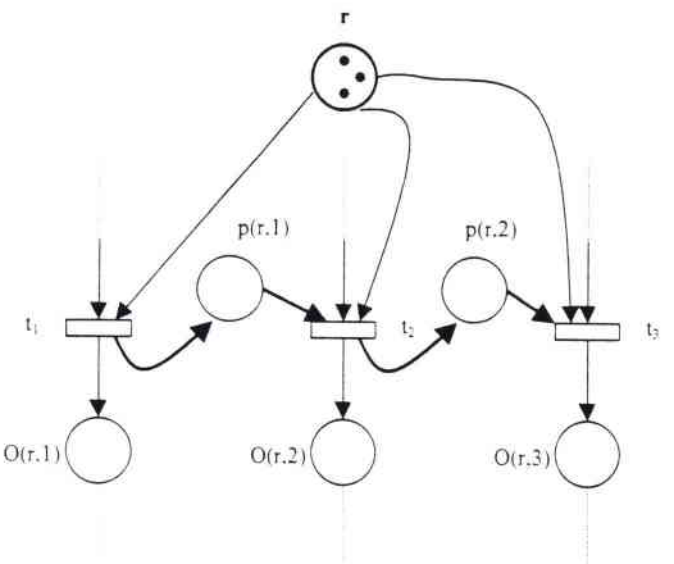

Figure IV.5 : Modélisation des séquences d'entrée dans chaque type de ressource

La figure IV.6 est le nouveau modèle du job-shop MRB de la figure IV.3, avec les séquences $\pi_{r_1} = O_{11}\ O_{21}\ O_{12}$ et $\pi_{r_2} = O_{11}\ O_{22}$.

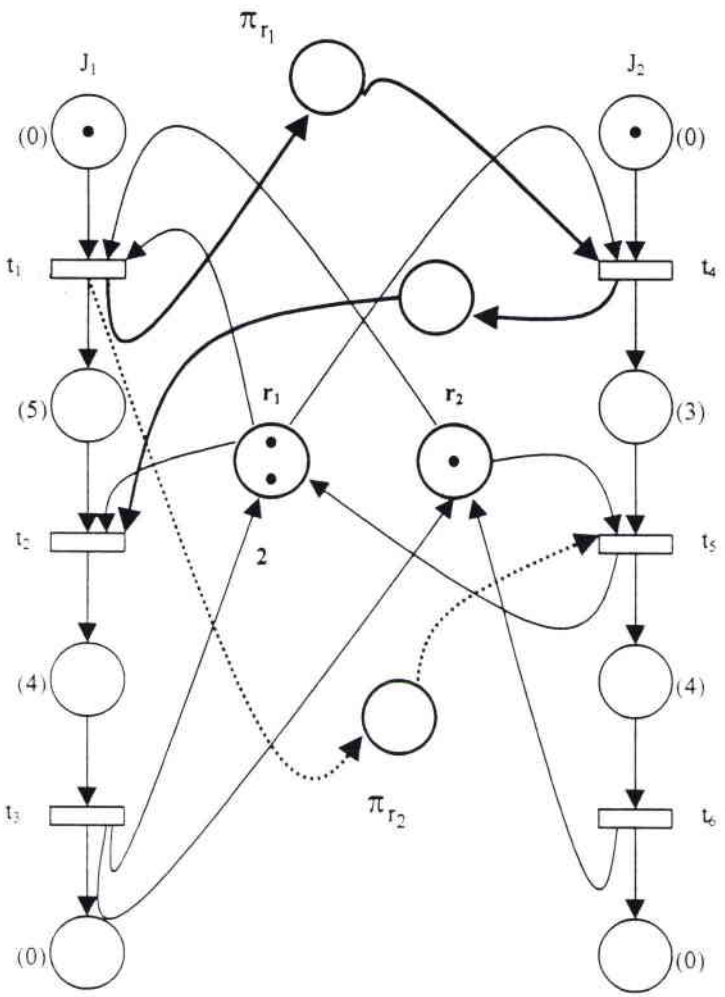

Figure IV.6 : Un modèle RdP complet avec les séquences d'entrée

La vérification de blocage est grandement simplifiée par le résultat suivant :

Théorème 4.5 : Dans un modèle RdP avec les séquences d'entrée dans tous les types de ressources, s'il existe une séquence de transitions franchissable conduisant à un blocage, alors le système est avec blocage quelle que soit la stratégie de franchissement.

Ce résultat nous permet d'examiner une seule séquence de transitions quelconque. La vérification de blocage est donc linéaire en nombre de transitions.

La preuve de ce résultat est fondée sur le caractère sans conflit effectif du nouveau RdP. Notons que chaque transition est franchie au plus une fois. Chaque séquence d'entrée π_r implique qu'au plus une transition franchissable nécessite les jetons de la place de ressource r (voir la figure IV.5). Cette transition est la transition d'entrée de la place O(r,1), ensuite celle de O(r,2), etc. Nous notons que, si le RdP considéré est un RdP ordinaire, les résultats de ce paragraphe sont des conséquences directes de Landweber et Robertson (78) pour les RdP persistants.

La preuve nécessite le résultat suivant :

Lemme 4.1 : Dans un RdP avec les séquences d'entrée connues dans tous les types de ressources, soit α et β deux séquences de transitions franchissables. Alors il existe une séquence γ, composée des transitions figurant dans β mais ne figurant pas dans α, telle que la séquence $\alpha\gamma$ est franchissable.

Preuve du lemme 4.1 :

Soit $\alpha = a_1\ a_2\ \ldots\ a_u$ et $\beta = b_1\ b_2\ \ldots\ b_v$. Soit b_k la première transition dans β mais ne figurant pas dans α, c'est-à-dire que les transitions $b_1\ b_2\ \ldots\ b_{k-1}$ figurent dans α. Il suffit de montrer que $\alpha\ b_k$ est franchissable. Pour cela, nous avons besoin des notations suivantes :

$\beta_k = b_1\ b_2\ \ldots\ b_k$: la séquence des k premières transitions de β

M_k^{β} : le marquage obtenu par la séquence β_k

M^{α} : le marquage obtenu par la séquence α

Nous savons que la transition b_k est franchissable à partir de M_{k-1}^{β} et il faut montrer que b_k est franchissable à partir de M^{α}.

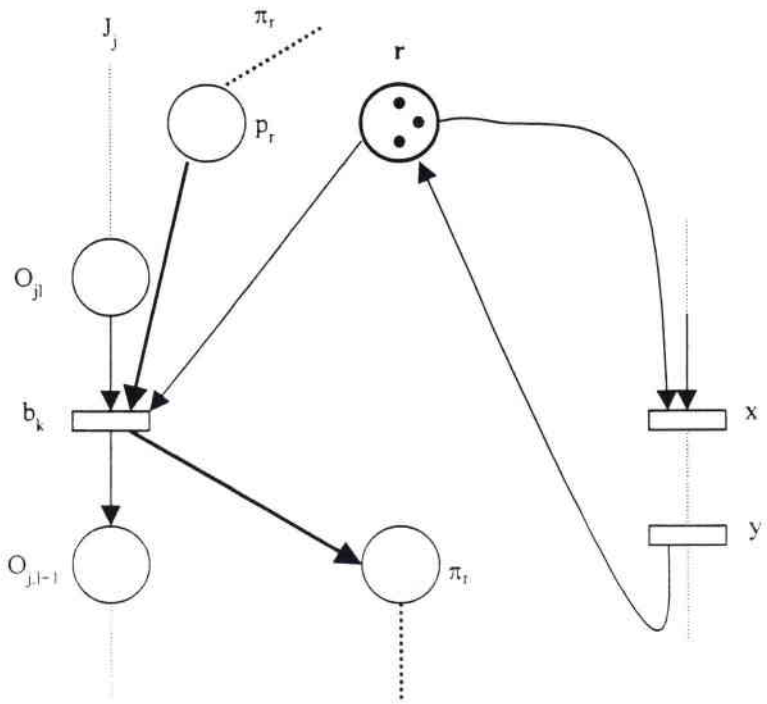

Figure IV.7 : Les places d'entrée d'une transition

Les places d'entrée de la transition b_k sont illustrées par la figure IV.7. Elles incluent une place-opération O_{jl}, une place p_r pour chaque séquence d'entrée π_r telle que $r \in R_{j,l+1}$ et les places de ressources r telles que $h_{jlr} < h_{j,l+1,r}$. Dans la suite, nous montrons que $M^{\alpha}(p) \geq M^{\beta}_{k-1}(p)$, pour toute place $p \in {}^{\bullet}b_k$. Puisque b_k est franchissable dans M^{β}_{k-1}, elle est franchissable dans M^{α}, ce qui conclut la preuve.

Puisque b_k est franchissable dans M^{β}_{k-1}, $M^{\beta}_{k-1}(O_{jl}) = 1$, ce qui implique que les transitions précédant b_k du travail J_j figurent dans β_{k-1}, alors elles figurent également dans α. Par conséquent, $M^{\alpha}(O_{jl}) = 1$.

De manière similaire, $M^{\beta}_{k-1}(p_r) = 1$, les transitions précédant b_k dans les séquences d'entrée π_r figurent dans β_{k-1} et donc dans α. D'où, $M^{\alpha}(p_r) = 1$.

Pour chaque place de ressource r, telle que $h_{j,l+1,r} > h_{jlr}$, la place r est une place d'entrée de b_k. Pour ses transitions x de sortie, c'est-à-dire $x \in r^{\bullet}$, puisque x appartient à la séquence d'entrée π_r, alors $x \in \beta_{k-1}$ si, et seulement si, $x \in \alpha$. Pour ses transitions y d'entrée, c'est-à-dire $y \in {}^{\bullet}r$, selon la définition de b_k, $y \in \beta_{k-1}$ implique $y \in \alpha$ et la réciproque est fausse. Pour

résumer, les transitions consommatrices de jetons de r figurant dans β sont celles figurant dans α et les transitions génératrices de jetons de r figurant dans β figurent dans α. D'où $M^{\alpha}(r) \geq M^{\beta}_{k-1}(\gamma)$.

Q.E.D.

Preuve du théorème 4.5 par contradiction :

Supposons qu'il y aient deux séquences α et β, telles que α conduit au blocage et β évite tous les blocages et franchit toutes les transitions. Selon le lemme 4.1, nous pouvons poursuivre la séquence α pour franchir les transitions ne figurant pas dans α. Ceci contredit l'hypothèse que la séquence α conduit au blocage. Le théorème est donc prouvé par l'absurde.

Q.E.D.

En fait, nous avons un résultat plus fort :

Corollaire 4.1 : Dans un modèle RdP avec les séquences d'entrée dans tous les types de ressources, si une transition t ne peut être franchie par une stratégie de franchissement donnée, alors elle ne peut être franchie du tout.

Le théorème 4.5 permet une vérification aisée de blocage si on connaît les séquences d'entrée dans les ressources. Se pose alors naturellement la question de l'ordonnancement optimale. Intuitivement, puisque le RdP avec les séquences d'entrée est sans conflit effectif, la stratégie de franchissement au plus tôt permet de franchir toutes les transitions au plus tôt et de minimiser la durée totale. Ceci est confirmé par le résultat suivant :

Théorème 4.6 : Etant donné les séquences π_r d'entrée dans tous les types de ressources, la stratégie de franchissement au plus tôt appliquée au RdP avec les séquences d'entrée donne l'ordonnancement optimal $\{S_t^*\}$, c'est-à-dire $S_t^* \leq S_t$, $\forall t \in T$, pour tout ordonnancement admissible $\{S_t\}$ respectant les séquences π_r.

Preuve :

Dans le RdP avec les séquences d'entrée, seules les places de ressource r sont concernées par les conflits structurels. Pour chaque place de ressource r, la séquence d'entrée $\pi_r = O(r,1), O(r,2), ..., O(r,N(r))$ implique que les jetons arrivant dans la place r sont utilisés d'abord pour la transition d'entrée de la place O(r,1), ensuite pour celle de O(r,2), puis pour celle de O(r,3), etc. Il n'y a pas de conflit effectif. De plus, nous connaissons la transition qui utilise le $n^{ème}$ jeton de la place ou qui arrive dans la place r. Ce type de RdP est dit RdP avec *switch*. Selon Baccelli et al. (92), les dates de franchissement S_t sont des fonctions monotones croissantes des temps associés aux places. Puisqu'il n'y a pas de conflit effectif, retarder volontairement le franchissement d'une transition t équivaut l'augmentation des temps (de

séjour) associés aux jetons dans les places d'entrée de t et donc, l'augmentation des S_t. La stratégie de franchissement au plus tôt est optimale.

Q.E.D.

IV.3.4.2 Cas où l'affectation des ressources aux opérations et la séquence d'entrée dans chaque ressource sont données

Dans ce paragraphe, nous distinguons les différentes unités de ressources du même type. Nous supposons que l'affectation des ressources aux opérations $\{\overline{R}_{jkr}\}$ est donnée, c'est-à-dire que nous connaissons les ressources à utiliser pour chaque opération. Nous obtenons un Système à Ressources Unitaires. Nous pouvons alors modéliser les séquences d'entrée dans les ressources en suivant les méthodes du paragraphe III.3.4.1.

Plus précisément, nous prenons en compte l'affectation et les séquences d'entrée en deux étapes. La première étape consiste à représenter chaque type de ressource par H_r places de ressources, chacune correspondant à une unité. Ces places de ressources sont reliées aux transitions en tenant compte de l'affectation et en suivant l'approche du paragraphe III.3 pour la modélisation des SRU. La figure IV.8 est le modèle RdP modifié du job-shop MRB de la figure IV.3 avec l'affectation $\overline{R}_{11r_1} = \{(r_1, 1)\}, = \overline{R}_{12r_1} \{(r_1, 1), (r_1, 2)\}, \overline{R}_{21r_1} = \{(r_1, 2)\}$.

La deuxième étape utilise la méthode du paragraphe III.3.3.1 pour compléter le modèle, en prenant en compte les séquences d'entrée dans les ressources. La figure IV.9 est le modèle final de l'exemple de la figure IV.8 avec les séquences $\pi_{(r_1,1)} = O_{11}\, O_{12}$, $\pi_{(r_1,2)} = O_{21}\, O_{12}$, $\pi_{r_2} = O_{12}\, O_{22}$.

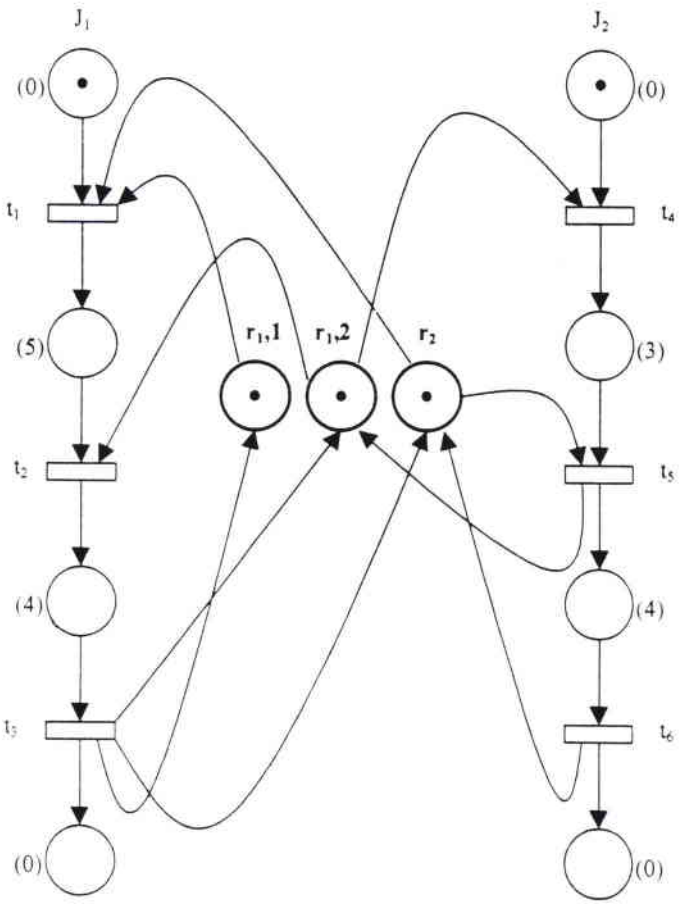

Figure IV.8 : Transformation d'un job-shop MRB en un SRU

Le théorème 3.3 implique immédiatement le résultat suivant qui permet la vérification de blocage et la détermination de l'ordonnancement au plus tôt.

Théorème 4.7 : Etant données l'affectation des ressources et les séquences d'entrée dans les ressources, le système est sans blocage si, et seulement si, chaque circuit élémentaire du RdP avec l'affectation et la séquence d'entrée contient au moins un jeton. Si le système est sans blocage, alors les dates de début au plus tôt de chaque opération est égale à la date de franchissement de la transition correspondante. Les dates de franchissement S_t peuvent être déterminées par la relation suivante :

$$S_t = \max_{t'/M_0(t',t)=0 \wedge t' \in {}^{\bullet\bullet}t} \{S_{t'} + p(t',t)\}$$

où $^{\bullet\bullet}t$ est l'ensemble des transitions reliées à t via une place, $M_0(t',t)$ et $p(t',t)$ sont respectivement le marquage initial et le temps associé à la place reliée t' à t.

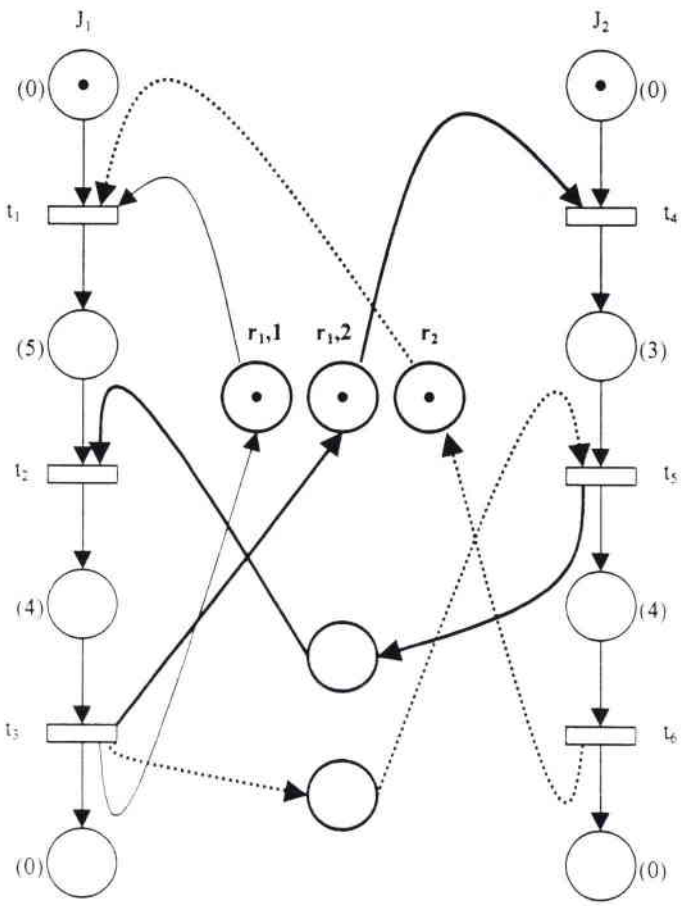

Figure IV.9 : Le modèle RdP avec l'affectation et les séquences d'entrée des ressources

IV.4 Méthodes de résolution

Nous proposons, dans ce paragraphe, des méthodes de résolution permettant la construction des ordonnancements sans blocage pour les job-shops MRB généraux. Ces méthodes peuvent être considérées comme des extensions des méthodes proposées dans le paragraphe III.4 pour les Systèmes à Ressources Unitaires.

Comme dans l'ordonnancement des SRU, les méthodes heuristiques proposées ci-dessous sont toutes des méthodes gloutonnes, aussi appelées méthodes de liste. Elles ordonnancent les travaux les uns après les autres, selon un ordre donné. Une fois ordonnancé, l'ordonnancement $\{S_{jk}\}$ des travaux déjà ordonnancés ne sera pas modifié dans la suite de la construction d'un ordonnancement. Nous utilisons également la méthode du recuit simulé pour optimiser l'ordre des travaux. Nous rappelons cette méthode dans la suite et nous

pouvons remarquer que l'idée de base est de vérifier tous les ordres possibles des travaux et de choisir l'ordre avec *Min* C_{max}. Plus précisément,

Algorithme 4.2 (Optimisation de l'ordre des travaux)

1. Choisir un ordre initial S_0 (par exemple J_1, J_2, ..., J_n) et calculer sa valeur de critère $f(S_0)$ à l'aide de l'Heuristique 1 ou de l'Heuristique 2.

2. Choisir une température initiale $T_0 > 0$ et faire $T \leftarrow T_0$.

3. Générer, au hasard, un ordre voisin S_1 (par permutation des deux travaux quelconques) et calculer sa valeur de critère $f(S_1)$ à l'aide de l'Heuristique 1 ou de l'Heuristique 2.

4. Si $f(S_1) \leq f(S_0)$, faire $S_0 \leftarrow S_1$. Sinon, faire $S_0 \leftarrow S_1$ avec la probabilité $\exp(-(f(S_1) - f(S_0)) / T)$.

5. Faire T = T r

6. Si $T > T_f$, où T_f est la température finale, aller à l'étape 3. Sinon, arrêt.

IV.4.1 Une méthode conservatrice

Nous présentons, dans ce paragraphe, une extension de la méthode conservatrice "Heuristique 1" proposée dans le paragraphe III.4.1.1 pour les SRU. Lors de l'ordonnancement d'un travail J_j, elle ordonnance ses opérations les unes après les autres, suivant le processus de fabrication.

Pour l'ordonnancement d'une opération (j,k), nous déterminons les ressources affectées $\{\overline{R}_{jkr}\}$ à l'opération (j,k) et la date de début S_{jk}. Pour cela, nous déterminons, pour chaque unité (r,i) de tout type de ressource r, la date $u_{r,i}$ à partir de laquelle la ressource (r,i) devient toujours libre. Nous ordonnons les ressources du type r selon leur date de disponibilité $u_{r,i}$. Soit $(r, [1]_{jk})$, $(r, [2]_{jk})$, …, $(r, [H_r]_{jk})$ la liste ordonnée, c'est-à-dire :

$$u_{r,[1]_{jk}} \leq u_{r,[2]_{jk}} \leq \ldots \leq u_{r,[H_r]_{jk}}$$

Il devient clair qu'à partir de la date $u_{r,[h_{jkr}]_{jk}}$, nous disposons suffisamment de ressources du type r pour l'opération (j,k). Par conséquent, l'opération (j,k) peut commencer sans risque de blocage à la date :

$$S_{jk} = \text{Max} \{S_{j,k-1} + p_{j,k-1}, \underset{r \in R_{jk}}{\text{Max}} \{u_{r,[h_{jkr}]_{jk}}\}\} \tag{4.6}$$

La date de début S_{jk} étant déterminée, nous procédons à l'affectation de ressources. L'idée est d'affecter les ressources de chaque type r de manière à maximiser la disponibilité des ressources non utilisées. Pour cela, nous choisissons les ressources r dont la date de disponibilité est maximale à condition que $u_{r,i} \leq S_{jk}$ (voir la figure IV.10). Plus précisément, si $h_{j,k-1,r} \geq h_{jkr}$, nous libérons $(h_{j,k-1,r} - h_{jkr})$ unités quelconques dans $\overline{R}_{j,k-1,r}$ et obtenons $\overline{R}_{jkr}$. Si $h_{j,k-1,r} < h_{jkr}$,

$$\overline{R}_{jkr} = \overline{R}_{j,k-1,r} \cup \{(r, [i^* - \Delta_{jkr}]_{jk}), ..., (r, [i^*]_{jk})\} \tag{4.7}$$

où $\Delta_{jkr} = h_{jkr} - h_{j,k-1,r} - 1$ et i^* est tel que $u_{r,[i^*]_{jk}} \leq S_{jk}$ et $u_{r,[i^*+1]_{jk}} > S_{jk}$.

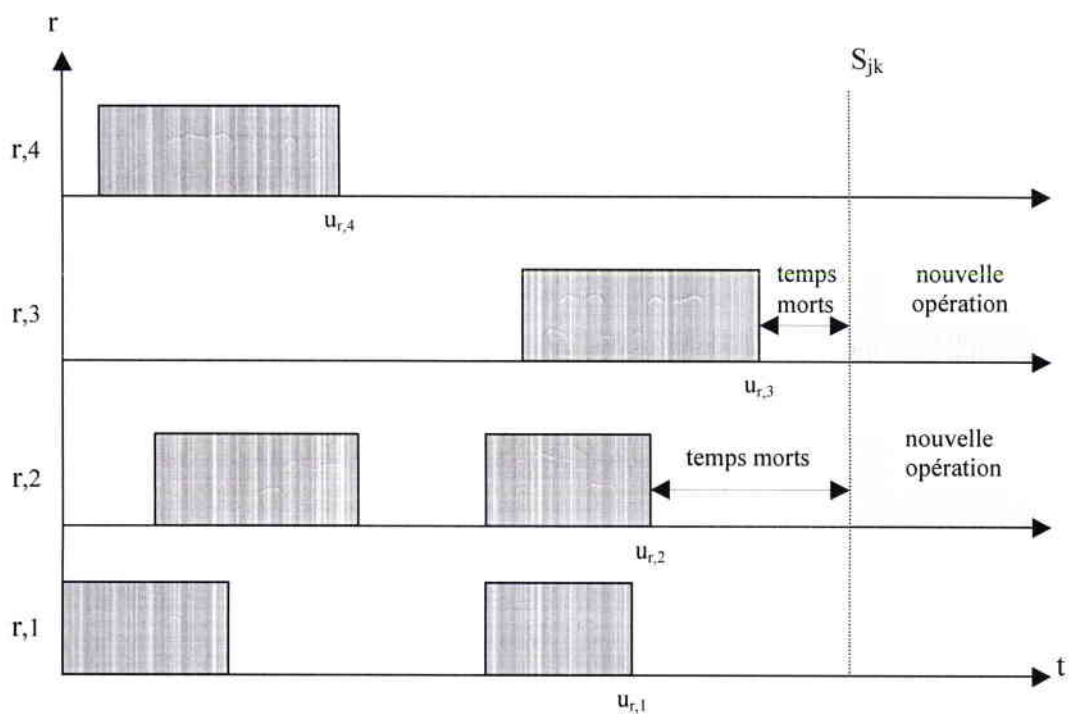

Figure IV.10 : Affectation dans la méthode conservatrice

Nous pouvons alors actualiser les indicateurs $u_{r,i}$ et la méthode peut continuer de la même manière. Plus formellement,

Algorithme 4.3 (Heuristique 1 (ordonnancement selon l'ordre $J_1, J_2, ..., J_n$))

1. Initialisation
   $u_{r,i} \leftarrow 0, \forall r \in R$ et $\forall (r,i) \in \overline{R}_r$

2. Ordonnancement dans l'ordre des travaux et des opérations :
   Pour j = 1 à n faire
   Pour k = 1 à N_j faire
   2.1. Ordonner les ressources de chaque type r telles que :
   $$u_{r,[1]_{jk}} \leq u_{r,[2]_{jk}} \leq ... \leq u_{r,[H_r]_{jk}}$$
   2.2. $S_{jk} = \text{Max} \{S_{j,k-1} + p_{j,k-1}, \underset{r \in R_{jk}}{\text{Max}} \{u_{r,[h_{jkr}]_{jk}}\}\}$
   2.3. Affectation :
   Pour r ∈ R faire
   Si $h_{j,k-1,r} \geq h_{jkr}$, libérer $h_{j,k-1,r} - h_{jkr}$ unités des ressources r pour obtenir $\overline{R}_{jkr}$. Sinon, calculer $\overline{R}_{jkr}$ à l'aide de l'équation (4.7)
   2.4. Mise à jour des dates de disponibilité :
   - $u_{r,i} \leftarrow S_{jk}, \forall (r,i) \in \overline{R}_{j,k-1,r}$ et $(r,i) \notin \overline{R}_{jkr}$
   - $u_{r,i} \leftarrow S_{jk} + p_{jk}, \forall (r,i) \in \overline{R}_{jkr}$

3. Le makespan $C_{max} \leftarrow S_{n,N_n} + p_{n,N_n}$

L'application de l'Heuristique 1 au job-shop MRB de la figure IV.3 donne l'ordonnancement suivant :

$u_{r_{1,1}} = 0$	$u_{r_{1,2}} = 0$	$u_{r_2} = 0$	$S_{11} = 0$
$u_{r_{1,1}} = 5$	$u_{r_{1,2}} = 0$	$u_{r_2} = 5$	$S_{12} = 5$
$u_{r_{1,1}} = 9$	$u_{r_{1,2}} = 9$	$u_{r_2} = 9$	$S_{21} = 9$
$u_{r_{1,1}} = 9$	$u_{r_{1,2}} = 12$	$u_{r_2} = 9$	$S_{22} = 12$
$C_{max} = 16$			

Table IV.1 : Ordonnancement d'un job-shop MRB par l'Heuristique 1

Le diagramme de Gantt donné dans la figure IV.11 donne une meilleure illustration de l'ordonnancement.

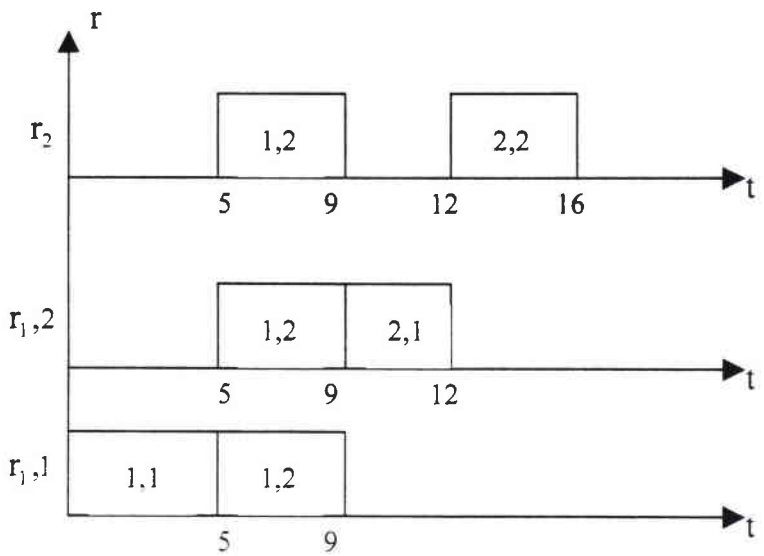

Figure IV.11 : Le diagramme de Gantt d'un ordonnancement construit par l'Heuristique 1

Théorème 4.6 : L'ordonnancement obtenu à l'aide de l'Heuristique 1 est sans blocage.

Preuve :

Il est clair que l'ordonnancement et l'affectation, obtenus à l'aide de l'Heuristique 1, respectent les contraintes de précédence et de capacité de toutes les ressources. L'affectation étant donnée, le job-shop MRB général est équivalent à un Système à Ressources Unitaires. De plus, selon l'Heuristique 1, la séquence d'entrée dans chaque ressource (r,i) est dans l'ordre des travaux et, pour chaque travail, dans l'ordre des opérations. L'ordonnancement est donc sans blocage selon les mêmes arguments que ceux utilisés dans la preuve du théorème 3.5.

Q.E.D.

Nous notons que l'Heuristique 1 s'applique à l'ordonnancement de tout job-shop MRB avec ou sans opérations ZD.

IV.4.2 Une méthode optimale locale

IV.4.2.1 Présentation de la méthode

Comme dans la méthode conservatrice (Heuristique 1), la méthode optimale locale, appelée Heuristique 2, ordonnance les travaux les uns après les autres et, ces travaux une fois ordonnancés, les dates de début d'un travail ne seront pas modifiées. La différence entre les deux méthodes réside dans l'ordonnancement de chaque travail. La nouvelle méthode ordonnance de manière optimale le travail en cours de considération et cherche à démarrer les opérations au plus tôt, tout en évitant les blocages. Plus précisément,

Algorithme 4.4 (Heuristique 2)

1. Choisir un ordre des travaux. Sans perte de généralité, soit $J_1, J_2, ..., J_n$ l'ordre.

2. Pour chaque travail J_j, résoudre le problème à un travail pour J_j en tenant compte des ordonnancements des travaux $J_1, J_2, ..., J_{n-1}$.

IV.4.4.2 Problème à un travail

Lors de l'ordonnancement d'un travail J_j, certains travaux sont déjà considérés et les dates du début de leurs opérations sont fixées. Soit

$$F = \{J_1, J_2, ..., J_{j-1}\}$$

l'ensemble des travaux déjà ordonnancés. L'ordonnancement J_j consiste à déterminer les dates du début de leurs opérations S_{jk} afin de minimiser leur date de fin C_j tout en respectant les ordonnancements des travaux dans F et en évitant les blocages. Plus formellement, il consiste à :

(SP) :

$$\text{Min } C_j,$$

sous les contraintes suivantes :

$$C_j = S_{j,N_j+1} \tag{4.8}$$

$$S_{jk} + p_{jk} \leq S_{j,k+1}, \forall 1 \leq k \leq N_j \tag{4.9}$$

$$\sum_{(i,l)/i\in F} h_{ilr}\,\mathbf{1}\{S_{il} \le t < S_{i,l+1}\} \le H_r - h_{jkr},\ \forall r \in R,\ \forall 1 \le k \le N_j\ \forall S_{jk} \le t < S_{j,k+1} \qquad (4.10)$$

$$\text{Il n'y a pas de blocage aux instants } \tau\text{- avec } \tau \in \{S_{j1}, ..., S_{jNj}\} \qquad (4.11)$$

où la relation (4.8) définit la date de fin du travail J_j, les contraintes (4.9) sont les contraintes de précédence, les relations (4.10) correspondent aux contraintes de capacité de chaque type de ressource qui assurent la disponibilité d'au moins h_{jkr} unités de ressource r pendant l'opération (j,k), la relation (4.11) est la contrainte de l'absence de blocage. Nous notons que l'ordonnancement des travaux dans F respecte les contraintes de précédence et de capacité de ressources et qu'il existe une séquence d'entrée π_r pour chaque type de ressource sans blocage.

Pour simplifier la formulation, nous introduisons la notation suivante :

$$I_{kt} = \sum_{r\in R} \Big(\sum_{(i,l)/i\in F} h_{ilr}\,\mathbf{1}\{S_{il} \le t < S_{i,l+1}\} + h_{jkr} - H_r \Big)^+, \qquad (4.12)$$

$$\forall\ 1 \le k \le N_j,\ \forall t \in \{1, 2, ..., H\},$$

où $(x)^+ = \max\{0, x\}$. De cette définition, $I_{kt} = 0$ si, et seulement si, il y a suffisamment de ressources libres dans la période t pour l'opération (j,k). Nous notons que les indicateurs I_{kt} sont des données pour le problème à un travail (SP).

Grâce à cette nouvelle notation, le problème à un travail peut être formulé comme suit: (SP) :

$$\text{Min } C_j,$$

sous les contraintes suivantes :

$$C_j = S_{j,Nj+1} \qquad (4.13)$$

$$S_{jk} + p_{jk} \le S_{j,k+1},\ \forall 1 \le k \le N_j \qquad (4.14)$$

$$I_{kt} = 0,\ \forall 1 \le k \le N_j\ \forall\ S_{jk} \le t < S_{j,k+1} \qquad (4.15)$$

$$\text{Il n'y a pas de blocage aux instants } \tau\text{- avec } \tau \in \{S_{j1}, ..., S_{jNj}\} \qquad (4.16)$$

Une comparaison avec la formulation du problème à un travail (SP) du paragraphe III.4.2 nous montre que le problème (SP) pour un job-shop MRB général est similaire au

problème (SP) pour un SRU sauf que (i) la définition des indicateurs I_{kt} est différente et (ii) la vérification de blocage est aussi différente.

De cette remarque, il est aisé de montrer que la méthode par recherche exhaustive (Algorithme 3.5) s'applique à la résolution du problème à un travail d'un job-shop MRB général, à condition d'utiliser la relation (4.12) pour le calcul des indicateurs I_{kt} et le théorème 4.4 pour la vérification de blocage (étape 4).

De manière similaire, la méthode Programmation Dynamique du paragraphe III.4.2.3 pour le problème à un travail d'un SRU peut être étendue. Il suffit pour cela d'utiliser la relation (4.12) pour le calcul des indicateurs de disponibilité des ressources I_{kt} et le théorème 4.4 pour les indicateurs de blocage B(k,t). L'algorithme 3.6 peut être utilisé en tenant compte de cette différence.

Néanmoins, compte tenu de la complexité explosive de la vérification de blocage à l'aide du théorème 4.4, la méthode optimale locale de ce paragraphe n'est efficace que (i) dans le cas d'un nombre faible de ressources et (ii) dans le cas d'un système simple pour lequel la vérification de blocage n'est pas nécessaire grâce au théorème 4.1. Pour cette raison, nous recommandons plutôt la méthode suivante fondée sur les systèmes simples pour l'ordonnancement d'un job-shop MRB général.

IV.4.3 Une méthode fondée sur les systèmes simples

Comme nous l'avons remarqué dans le paragraphe précédent, les méthodes d'ordonnancement sont plus simples et plus efficaces pour les systèmes simples car, grâce au théorème 4.1, nous n'avons pas besoin de vérifier explicitement l'existence de blocage.

Comme dans le paragraphe III.5, nous pouvons transformer un job-shop MRB général en un système simple en deux étapes : (i) définir la forme canonique d'un job-shop MRB en insérant des opérations ZD et (ii) définir la forme canonique modifiée en modifiant les temps opératoires des opérations ZD.

Plus précisément, considérons un job-shop MRB composé d'un ensemble de travaux $\{J_1, J_2, ..., J_j, ..., J_n\}$ avec $J_j = \{(R_{j1}, h_{j1}, p_{j1}), ..., (R_{jk}, h_{jk}, p_{jk}), ..., (R_{jN_j}, h_{jN_j}, p_{jN_j})\}$. Sa forme canonique est un job-shop MRB composé des travaux $\{J_1^c, J_2^c, ..., J_n^c\}$ tels que

$$J_j^c = \{(R_{j1}^c, h_{j1}^c, p_{j1}^c), (R_{j2}^c, h_{j2}^c, p_{j2}^c), ..., (R_{jN_j^c}^c, h_{jN_j^c}^c, p_{jN_j^c}^c)\}$$

où

$$N_j^c = 2N_j - 1$$

$$(R^c_{j,2k-1}, h^c_{j,2k-1}, p^c_{j,2k-1}) = (R_{jk}, h_{jk}, p_{jk}), \qquad \forall 1 \le k \le N_j$$

$$(R^c_{j,2k}, h^c_{j,2k}, p^c_{j,2k}) = (R_{jk} \cup R_{j,k+1}, h_{jk} \vee h_{j,k+1}, 0), \qquad \forall 1 \le k \le N_j.$$

avec $h_{jk} \vee h_{j,k+1}$ un vecteur dont l'élément r est $\max\{h_{jkr}, h_{j,k+1,r}\}$.

Nous définissons également un système simple, que nous appelons forme canonique modifiée du SRU, en remplaçant les opérations ZD de la forme canonique par des opérations à délai unitaire. Plus précisément, la forme canonique modifiée est un système composé des travaux $\{ J_1^m, J_2^m, \ldots, J_n^m \}$ tels que $N_j^m = N_j^c = 2N_j - 1$, $R_{jk}^m = R_{jk}^c$, $h_{jk}^m = h_{jk}^c$, $p_{jk}^m = \max\{ p_{jk}^c, 1\}$, $\forall$ (j,k).

La nouvelle méthode construit l'ordonnancement du job-shop MRB général en quatre étapes: (i) calculer la forme canonique modifiée ; (ii) ordonnancer la forme canonique modifiée; (iii) extraire les séquences d'entrée π_r dans tous les types de ressources ou extraire l'affectation des ressources aux opérations ainsi que la séquence d'entrée dans chaque ressource, à l'aide de l'algorithme 4.1, pour le job-shop MRB général à partir de l'ordonnancement de l'étape (ii) ; enfin (iv), calculer l'ordonnancement du job-shop MRB avec les séquences π_r dans tous les types de ressources, à l'aide du théorème 4.6 ou l'affectation ainsi que les séquences d'entrée, dans chaque ressource, à l'aide du théorème 4.7. Pour résumer :

Algorithme 4.5 (Méthode d'ordonnancement fondé sur les systèmes simples)

1. Déterminer la forme canonique modifiée du job-shop MRB.

2. Ordonnancer cette forme canonique modifiée à l'aide de l'algorithme du recuit simulé sans vérification de blocage.

3. Extraire la séquence d'entrée π_r dans chaque type de ressource ou l'affectation ainsi que la séquence d'entrée dans chaque ressource à l'aide de l'algorithme 4.1.

4. Calculer les dates de début des opérations du job-shop MRB avec la séquence π_r d'entrée dans chaque type de ressource, à l'aide du théorème 4.6 ou à l'aide du théorème 4.7 avec l'affectation et la séquence d'entrée dans chaque ressource.

Il est bien entendu que l'ordonnancement de la forme canonique modifiée est également admissible pour le job-shop MRB. L'étape 4 de l'algorithme permet l'élimination des temps morts et conduit à un meilleur ordonnancement.

IV.5 Résultats numériques

Dans ce paragraphe, nous présentons six exemples de Systèmes à Ressources Multiples et les ordonnancements obtenus à l'aide de l'Heuristique 2. Le premier exemple est une extension de l'exemple dû à Ramaswany et Joshi (96) et nous l'avons donné dans la table III.3 du chapitre précédent. Les autres exemples ont été générés de façon aléatoire.

IV.5.1 Exemple 1

L'exemple 1 est une extension de l'exemple de Ramaswany et Joshi (96), un Système à Ressources Unitaires déjà étudié dans le paragraphe III.6.1. Dans ce paragraphe, nous considérons les mêmes travaux $\{J_1, J_2, J_3, J_4\}$ dont les temps opératoires et les ressources nécessaires pour les opérations sont ceux de la table III.3. La différence est le changement des disponibilités des unités de chaque type de ressource. Nous considérons également les trois machines M_1, M_2 et M_3. Par contre, nous supposons qu'il y a deux moyens de transport T. La disponibilité des ressources est résumée par la table IV.2.

Ressource	M_1	M_2	M_3	T
Disponibilité	1	1	1	2

Table IV.2 : Nouvelle disponibilité des ressources pour l'exemple de Ramaswany et Joshi (96)

L'ordonnancement obtenu à l'aide de l'Heuristique 2, avec le critère de la minimisation du makespan, est donné dans la table IV.3. Nous constatons que l'ajout d'un moyen de transport permet une réduction importante de la durée totale. Le makespan est passé de 568 unités de temps avec un moyen de transport à 451 unités de temps avec deux moyens de transport. Ceci prouve l'importance de la prise en compte des ressources dites "secondaires" dans la définition de l'ordonnancement.

	J_1	J_2	J_3	J_4
Opération	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
1	1, 5	51, 118	1, 5	6, 9
2	6, 45	119, 330	6, 50	10, 118
3	46, 50	331, 337	51, 53	119, 150
4	51, 150	338, 410	54, 118	151, 222
5	151, 222	411, 414	119, 124	223, 330
6	223, 258	415, 446	125, 222	331, 365
7	259, 262	447, 451	223, 226	366, 370
Makespan = 451				
Temps de CPU = 30 secondes				

Table IV.3 : Ordonnancement de l'exemple de Ramaswany et Joshi (96) par l'Heuristique 2

IV.5.2 D'autres exemples

Ces exemples ont été générés de manière aléatoire. Nous choisissons le nombre de travaux et le nombre de types de ressources et générons à l'aide de variables aléatoires les autres paramètres du système : le nombre de ressources de chaque type, le nombre d'opérations de chaque travail, le besoin en ressources de chaque opération et son temps opératoire. Les exemples générés (les travaux et la disponibilité des ressources) sont donnés dans les tables IV.4, IV.5, IV.9, IV.10, IV.12, IV.13, IV.15, IV.16, IV.18 et IV.19. Les ordonnancements obtenus à l'aide de l'Heuristique 2 sont donnés dans les tables IV.6, IV.11, IV.14, IV.17 et IV.20. Les résultats obtenus à l'aide de l'Heuristique 2 sont donnés dans la suite de ce paragraphe.

Exemple 2

Travaux	Opérations		
	1	2	3
J_1	$1R_3$, 9		
J_2	$1R_4$, 7	$1R_2$ $1R_3$, 2	
J_3	$1R_2$, 3	$1R_1$ $1R_4$, 10	$1R_4$, 8
J_4	$1R_4$, 7	$1R_2$ $1R_3$, 2	

Table IV.4 : Travaux de l'exemple 2

Disponibilité	2	3	2	2

Table IV.5 : Disponibilité des ressources de l'exemple 2

Travaux	Opérations		
	1	2	3
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 9		
J_2	1, 7	8, 9	
J_3	1, 3	4, 13	14, 21
J_4	8, 14	15, 16	
Makespan = 21			

Table IV.6 : Ordonnancement de l'exemple 2

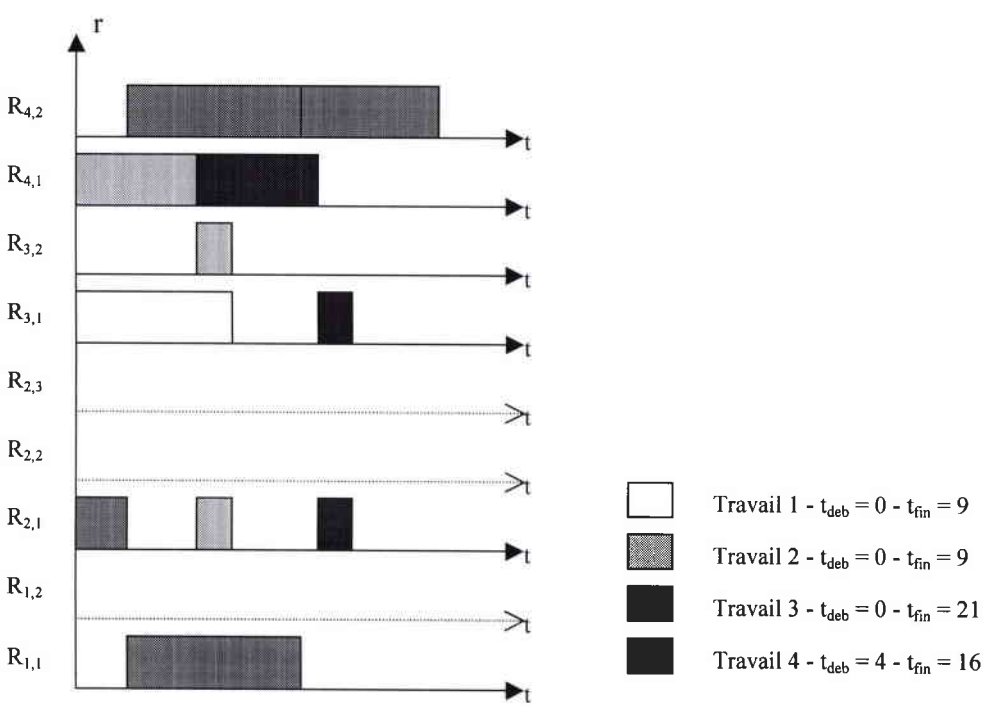

Figure IV.11 : Diagramme de Gantt de l'ordonnancement donné dans la table IV.6

Pour cet exemple, nous considérons également l'ordonnancement avec un nombre réduit de ressources . La nouvelle disponibilité des ressources est celle de la table IV.7 et le nouvel ordonnancement celui de la table IV.8. Nous constatons une augmentation du makespan.

Pour cet exemple, nous considérons également l'ordonnancement avec un nombre réduit de ressources . La nouvelle disponibilité des ressources est celle de la table IV.7 et le nouvel ordonnancement celui de la table IV.8. Nous constatons une augmentation du makespan.

Ressource	R_1	R_2	R_3	R_4
Disponibilité	2	1	1	1

Table IV.7 : Nouvelle disponibilité des ressources pour l'exemple 2

Travaux	Opérations		
	1	2	3
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 9		
J_2	29, 35	36, 37	
J_3	1, 3	4, 13	14, 21
J_4	22, 28	29, 30	
Makespan = 37			

Table IV.8 : Nouvel ordonnancement de l'exemple 2 par l'Heuristique 2 avec la disponibilité des ressources donnée dans la table IV.7

Exemple 3

Travaux	Opérations		
	1	2	3
J_1	$2R_3$, 6		
J_2	$2R_1$, 9		
J_3	$1R_3$, 6	$1R_1$, 8	$1R_1 1R_4\ 1R_6$, 9
J_4	$2R_4$, 10	$1R_1 1R_2\ 1R_3$, 3	
J_5	$2R_2\ 1R_4$, 6	$1R_4$, 10	$1R_2\ 1R_6$, 1
J_6	$2R_2\ 1R_5$, 9	$1R_2\ 1R_3\ 1R_5$, 5	

Table IV.9 : Travaux de l'exemple 3

Ressource	R_1	R_2	R_3	R_4	R_5	R_6
Disponibilité	3	5	3	3	3	5

Table IV.10 : Disponibilité des ressources pour l'exemple 3

Travaux	Opérations		
	1	2	3
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 6		
J_2	1, 9		
J_3	1, 6	7, 14	15, 23
J_4	1, 10	11, 13	
J_5	1, 6	7, 16	17, 17
J_6	1, 9	11, 15	
Makespan = 23			

Table IV.11 : Ordonnancement de l'exemple 3

Exemple 4

Travaux	Opérations		
	1	2	3
J_1	$2R_1$ $1R_2$, 1	$2R_1$, 9	$1R_2$ $1R_4$, 7
J_2	$1R_1$ $1R_4$ $1R_5$, 9	$1R_2$ $1R_4$ $1R_5$, 3	
J_3	$2R_2$ $1R_3$ $1R_4$, 7	$2R_1$ $1R_2$, 1	
J_4	$2R_2$, 1	$1R_2$ $1R_4$ $1R_5$, 5	$1R_2$, 4
J_5	$2R_2$ $1R_5$, 8	$2R_1$ $1R_3$ $1R_5$, 2	$2R_1$ $1R_2$ $1R_3$, 7
J_6	$1R_2$ $1R_3$ $1R_5$, 1	$1R_1$, 4	$2R_3$, 1
J_7	$1R_2$, 6		
J_8	$1R_1$ $1R_2$ $1R_4$, 3		
J_9	$2R_3$ $1R_4$ $1R_5$, 2	$1R_3$ $1R_4$ $1R_5$, 9	
J_{10}	$2R_1$ $1R_2$, 4		

Table IV.12 : Travaux de l'exemple 4

Ressource	R_1	R_2	R_3	R_4	R_5
Disponibilité	2	4	2	2	2

Table IV.13 : Disponibilité des ressources pour l'exemple 4

Travaux	Opérations		
	1	2	3
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 1	2, 10	11, 17
J_2	11, 19	20, 22	
J_3	18, 24	25, 25	
J_4	1, 1	2, 6	7, 10
J_5	23, 30	31, 32	33, 39
J_6	1, 10	11, 14	15, 15
J_7	2, 7		
J_8	26, 28		
J_9	25, 26	27, 35	
J_{10}	40, 43		
Makespan = 43			

Table IV.14 : Ordonnancement de l'exemple 4

Exemple 5

Travaux	Opérations		
	1	2	3
J_1	$1R_3$, 6		
J_2	$1R_1$, 9		
J_3	$1R_3$, 6	R_1, 8	
J_4	$1R_1$ $1R_6$, 10	R_6, 8	$1R_2$ $1R_6$, 7
J_5	$1R_4$ $1R_5$, 4	$1R_6$, 7	
J_6	$1R_1$ $1R_2$, 1	$1R_2$, 9	$1R_2 1R_5$, 9
J_7	$1R_2$ $1R_3$ $1R_5$, 10	$1R_2$ $1R_3$ $1R_6$, 4	
J_8	$1R_3$ $1R_5$, 8	$1R_1$ $1R_4$ $1R_6$, 3	$1R_3$ $1R_5$ $1R_6$, 4
J_9	$1R_1$, 3		
J_{10}	$1R_1$ $1R_2$ $1R_4$, 5		

Table IV.15 : Travaux de l'exemple 5

Ressource	R_1	R_2	R_3	R_4	R_5	R_6
Disponibilité	2	3	2	2	1	3

Table IV.16 : Disponibilité des ressources pour l'exemple 5

Travaux	Opérations		
	1	2	3
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 6		
J_2	1, 9		
J_3	1, 6	7, 14	
J_4	10, 19	20, 27	28, 34
J_5	1, 4	5, 11	
J_6	1, 1	2, 10	11, 19
J_7	20, 29	30, 33	
J_8	30, 37	38, 40	41, 44
J_9	2, 4		
J_{10}	15, 19		
Makespan = 44			

Table IV.17 : Ordonnancement de l'exemple 5

Exemple 6

Travaux	Opérations				
	1	2	3	4	5
J_1	$1R_2$ $2R_6$ $1R_7$, 1	$1R_1$ $1R_3$ $1R_5$, 3	$1R_5$, 9	$1R_{10}$, 8	
J_2	$1R_9$, 7	$1R_4$ $1R_7$, 2	$1R_6$ $1R_7$, 6	$1R_6$, 10	
J_3	$1R_9$, 6				
J_4	$1R_3$ $1R_5$ $1R_8$, 4	$1R_3$ $1R_4$ $1R_7$, 10			
J_5	$2R_2$ $1R_5$ $1R_{10}$, 9	$1R_4$ $1R_7$ $1R_8$, 9			
J_6	$1R_1$ $1R_9$, 9	$1R_{10}$, 3			
J_7	$2R_6$, 5	$1R_2$, 4	$1R_5$, 1	$1R_1$ $1R_4$, 3	$2R_2$ $1R_3$ $1R_4$, 3
J_8	$1R_{10}$, 5				
J_9	$1R_4$ $1R_5$ $1R_{10}$, 9	$1R_6$ $1R_7$, 8	$1R_4$, 7		
J_{10}	$1R_1$ $1R_9$, 4				

Table IV.18 : Travaux de l'exemple 6

Ressource	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
Disponibilité	2	3	2	2	1	3	1	2	1	2

Table IV.19 : Disponibilité des ressources pour l'exemple 6

Travaux	Opérations				
	1	2	3	4	5
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 1	2, 4	5, 13	14, 21	
J_2	1, 7	8, 9	10, 15	16, 25	
J_3	8, 13				
J_4	39, 46	47, 56			
J_5	14, 22	23, 31			
J_6	14, 22	23, 25			
J_7	2, 6	7, 22	23, 23	24, 26	27, 29
J_8	1, 5				
J_9	30, 38	39, 46	47, 53		
J_{10}	23, 26				
Makespan = 56					

Table IV.20 : Ordonnancement de l'exemple 6

IV.6 Conclusion

Dans ce chapitre, nous avons considéré l'ordonnancement des job-shops MRB généraux. Comme pour les Systèmes à Ressources Unitaires, nous avons montré que, pour les systèmes simples, les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage. Ce résultat est particulièrement important pour l'ordonnancement des job-shops MRB généraux. Nous avons ensuite étudié la représentation des job-shops MRB à l'aide des RdP généralisés, la vérification de blocage pour un état donné et pour un ordonnancement donné. Nous avons constaté que la vérification de blocage est beaucoup plus difficile que dans le cas des Systèmes à Ressources Unitaires. Finalement, nous avons étendu les méthodes gloutonnes du chapitre III pour l'ordonnancement des job-shops MRB généraux.

Comme dans le cas des Systèmes à Ressources Unitaires, nous pouvons aisément prendre en compte d'autres critères réguliers telles que la minimisation de la somme de retards et la somme d'en-cours. L'extension aux systèmes avec gammes non-linéaires et les flexibilités des ressources et des gammes est une recherche future importante.

Chapitre V

Le problème d'ordonnancement avec moyens de transport

V.1 Introduction

Le travail présenté dans ce chapitre a été réalisé dans le cadre du stage de fin d'étude de Mlle Elkarama (Elkarama, 98) effectué au sein du projet MACSI, à l'encadrement duquel j'ai participé. Dans ce chapitre, nous nous limitons à une classe particulière des job-shops MRB : les ateliers flexibles ou les job-shops avec moyens de transport. Un tel système est similaire à un job-shop classique mais un moyen de transport (ou AGV) est nécessaire tout au long de la réalisation de chaque travail. Nous avons donc un job-shop MRB où les ressources sont les machines et les AGV ; le processus de fabrication de chaque travail peut se mettre sous la forme suivante :

$$\{(AGV,M_{j1},p_{j1}), (AGV,0), (AGV,M_{j2},p_{j2}), (AGV,0), ..., (AGV,M_{jN_j},p_{jN_j})\}$$

où $\{M_{j1}, M_{j2}, ..., M_{jN_j}\}$ est la suite de machines à visiter.

L'objectif de ce chapitre est d'étudier l'efficacité de la méthode Relaxation Lagrangienne pour l'ordonnancement des job-shops MRB en suivant la méthodologie proposée par Luh et Hoitomt (93) que nous avons présenté dans le paragraphe I.5.5.1. Pour une meilleure compréhension de ce chapitre, nous rappelons cette méthodologie fondée sur la relaxation Lagrangienne pour l'ordonnancement des systèmes de production. Cette méthodologie consiste à (i) relaxer certaines contraintes du problème, par le biais des multiplicateurs de Lagrange, pour obtenir des problèmes relaxés faciles à résoudre, (ii) résoudre le problème dual, c'est-à-dire déterminer les multiplicateurs de Lagrange, (iii) construire un ordonnancement admissible à partir de la solution du problème dual. L'avantage de cette méthode est que la solution du problème dual fournit une borne inférieure de qualité, permettant la construction d'une bonne solution admissible et son évaluation.

Ce chapitre est organisé de la manière suivante : le paragraphe V.2 donne la formulation du problème d'ordonnancement d'un job-shop avec AGV; le paragraphe V.3 présente une relaxation Lagrangienne du problème d'ordonnancement et définit le problème relaxé et le problème dual ; le paragraphe V.4 aborde la résolution du problème relaxé et du problème dual ; le paragraphe V.5 est consacré à la construction d'un ordonnancement réalisable en utilisant la Programmation Dynamique, l'algorithme de liste et le recuit simulé; le paragraphe V.6 présente les résultats numériques ; le paragraphe V.7 considère les extensions possibles et le paragraphe V.8 est une conclusion.

V.2 Formulation du problème d'ordonnancement

La complexité du problème d'ordonnancement dans les ateliers de type job-shop incite à utiliser une approche de décomposition. Aussi la méthode de relaxation lagrangienne consiste à relaxer certaines contraintes, notamment les contraintes de capacité des ressources (machines et moyens de transport), de façon à ramener un problème que l'on peut décomposer à des sous-problèmes beaucoup plus faciles à résoudre, à l'aide des multiplicateurs de Lagrange. La solution trouvée, non nécessairement admissible, constitue un bon point de départ à un algorithme de liste qui va nous permettre de trouver une bonne solution réalisable.

V.2.1 Notations

Ressources

M	:	le nombre de types de machines
L	:	le nombre des AGV

Travaux

n	:	le nombre de travaux
O_{jk} ou (j,k)	:	la $k^{ème}$ opération du travail j
$\{O_{j1}, O_{j2}, ..., O_{jk}, ..., O_{jN_j}\}$:		la séquence d'opérations du travail j
N_j	:	le nombre d'opérations du travail j
$M_{jk} \in \{1, 2, ..., M\}$	:	la machine utilisée pour l'opération (j,k)
p_{jk}	:	le temps opératoire de l'opération (j,k)
S_{jk}	:	la date de début de l'opération (j,k)
C_{jk}	:	la date de fin de l'opération (j,k)
C_j	:	la date de fin du travail j
d_j	:	la date échue du travail j
H	:	l'horizon de temps

J	:	le critère à optimiser
w_j	:	la pondération du travail j
T_j	:	le retard du travail j, défini par : $T_j = \max(0, C_j - d_j)$

Nous utilisons également les notations suivantes :

$$1(x) = \begin{cases} 1 & \text{si x est vraie,} \\ 0 & \text{sinon.} \end{cases}$$

$$\delta_{jk}^{m} = \begin{cases} 1 & \text{si } M_{jk} = m, \\ 0 & \text{sinon.} \end{cases}$$

V.2.2 Formulation du problème

Le but est de minimiser la fonction retard quadratique pondéré :

$$J = \sum_{j=1}^{n} W_j \cdot T_j^2$$

Ce critère, dont on démontre aisément la régularité, tient compte de la pondération des travaux à réaliser, de l'importance du retard, et du fait que chaque travail devient plus critique pour chaque unité de temps passée de sa date échue. Son additivité, par ailleurs, facilite l'approche de la décomposition que l'on souhaite utiliser.

Le problème d'ordonnancement devient alors :

$$\text{Minimiser} \quad J = \sum_{j=1}^{n} W_j \cdot T_j^2$$

sous les contraintes :

$$S_{jk} + p_{jk} \leq S_{j,k+1}, \ \forall 1 \leq j \leq n, \ \forall 1 \leq k \leq N_j \qquad (5.1)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{N_j} \mathbf{1}\{S_{jk} \le t < S_{j,k+1}\}\ \delta_{jk}^{m} \le 1,\ \forall 1 \le t \le H,\ \forall\ 1 \le m \le M \qquad (5.2)$$

$$\sum_{j=1}^{n} \mathbf{1}\{S_{jk} \le t < S_{jNj} + p_{jNj}\} \le L,\ \forall 1 \le t \le H \qquad (5.3)$$

$$C_j = S_{jNj} + p_{jNj} - 1,\ \forall 1 \le j \le n \qquad (5.4)$$

La contrainte (5.1) exprime les relations de précédence entre les opérations appartenant au même travail, la contrainte (5.2) exprime que chaque machine ne peut traiter qu'une opération à la fois, la contrainte (5.3) traduit la capacité en AGV et la contrainte (5.4) donne l'expression de la date de fin pour chaque travail.

V.3 Relaxation du problème

V.3.1 Problème relaxé

L'idée est de relaxer les contraintes de capacité des ressources en machines et en AGV en introduisant les multiplicateurs de Lagrange positifs λ et μ. Le problème relaxé sera alors de la forme :

$$L(\lambda,\mu) = \min \Big(\sum_{j=1}^{n} W_j T_j^2 + \sum_{t,m} \lambda_{tm} \left\{\sum_{j,k} 1(S_{jk} \le t < S_{jk} + p_{jk})\delta_{jk}^{m} - 1\right\} + \sum_{t} \mu_t \left\{\sum_{j} 1(S_{j,1} \le t < S_{jN_j} + p_{jN_j}) - L\right\}$$

sous les contraintes :

$$S_{jk} + p_{jk} \le S_{j,k+1},\ \forall 1 \le j \le n,\ \forall 1 \le k \le N_j$$

$$C_j = S_{jNj} + p_{jNj} - 1,\ \forall 1 \le j \le n$$

Un multiplicateur λ_{tm} (respectivement μ_t) peut être interprété comme le coût de l'utilisation de la machine m (respectivement d'un AGV) pendant la date t.

Le problème relaxé peut être mis sous la forme :

$$L(\lambda,\mu) = -\sum_{t,m}\lambda_{tm} - L\sum_{t}\mu_t + \min\sum_{j=1}^{n}(W_jT_j^2 + \sum_{k=1}^{N_j}\sum_{t=S_{jk}}^{S_{jk}+p_{jk}-1}\lambda_{tm} + \sum_{t=S_{jl}}^{S_{jN_j}+p_{jN_j}-1}\mu_t)$$

Les relations de précédence lient les opérations du même travail mais les travaux eux-mêmes sont indépendants. Par conséquent,

$$L(\lambda,\mu) = -\sum_{t,m}\lambda_{tm} - L\sum_{t}\mu_t + \sum_{j=1}^{n}\left\{\min(W_jT_j^2 + \sum_{k=1}^{N_j}\sum_{t=S_{jk}}^{S_{jk}+p_{jk}-1}\lambda_{tm} + \sum_{t=S_{jl}}^{S_{jN_j}+p_{jN_j}-1}\mu_t)\right\}$$

La résolution du problème relaxé se ramène donc à la résolution des sous-problèmes niveau travail j :

$L_j(\lambda,\mu)$:

$$(\sum_{j=1}^{n}W_jT_j^2 + \sum_{k=1}^{N_j}\sum_{t=S_{jk}}^{S_{jk}+p_{jk}-1}\lambda_{tm} + \sum_{t=S_{jl}}^{S_{jN_j}+p_{jN_j}-1}\mu_t)$$

sous les contraintes :

$$S_{jk} + p_{jk} \le S_{j,k+1},\ \forall 1 \le k \le N_j$$

V.3.2. Problème dual

Le problème dual associé est :

$$L_1 = \underset{\lambda,\mu \ge 0}{\text{maximiser}}\ L(\lambda,\mu)$$

La résolution du problème dual fournit une borne inférieure de la solution optimale. En effet, les multiplicateurs sont positifs et donc, en posant $\{S_{jk}^*\}$ l'ordonnancement optimal du problème initial, $\lambda \ge 0$, $\mu \ge 0$:

$$L(\lambda,\mu) = \min(\sum_{j=1}^{n}W_jT_j^2 + \sum_{t,m}\lambda_{tm}\left\{\sum_{j,k}1(S_{jk} \le t < S_{jk}+p_{jk}).\delta_{jk}^m - 1\right\} + \sum_{t}\mu_t\left\{\sum_{j}1(S_{jl} \le t < S_{j,N_j}+p_{j,N_j}) - L\right\}$$

donc :

$$L(\lambda,\mu) \le \min \Big(\sum_{j=1}^{n} W_j T_j^{*2} + \sum_{t,m} \lambda_{tm} \Big\{ \sum_{j,k} 1(S_{jk}^{*} \le t < S_{jk}^{*} + p_{jk}).\delta_{jk}^{m} - 1 \Big\} + \sum_{t} \mu_t \Big\{ \sum_{j} 1(S_{jl}^{*} \le t < S_{j,N_j}^{*} + p_{j,N_j}) - L \Big\}$$

vu que $\{ S_{jk}^{*} \}$ est une solution admissible du problème relaxé.

Sachant que $\lambda \ge 0$ et $\mu \ge 0$ et que les contraintes des ressources sont vérifiées par l'ordonnancement $\{ S_{jk}^{*} \}$, alors :

$$\lambda_{tm} \Big\{ \sum_{j,k} 1(S_{jk}^{*} \le t < S_{jk}^{*} + p_{jk}).\delta_{jk}^{m} - 1 \Big\} \le 0 \,,\ \forall 1 \le t \le H$$

$$\mu_t \Big\{ \sum_{j} 1(S_{jl}^{*} \le t < S_{j,N_j}^{*} + p_{j,N_j}) - L \Big\} \le 0 \,,\ \forall\ 1 \le m \le M$$

et par suite :

$$L(\lambda,\mu) \le \sum_{j=1}^{n} W_j T_j^{*2} \,,\ \forall\ \lambda,\mu \ge 0$$

soit finalement :

$$L_1 = \max_{\lambda,\mu \ge 0} L(\lambda,\mu) \le J^{*}$$

V.4 Méthodologie de résolution

Avant de passer aux détails, notons que la résolution des sous-problèmes niveau travail se fait en utilisant une méthode de décomposition structurelle. En effet, l'ensemble des variables de décisions est décomposé en deux sous-ensembles, l'un contenant les dates de début des différentes opérations et l'autre, les valeurs des multiplicateurs λ et μ. Ceux-ci sont d'abord initialisés à une valeur quelconque, soit à zéro par exemple. Ensuite, on cherche à résoudre le problème relaxé et donc à trouver les dates de début associée aux valeurs des multiplicateurs. S'il y a variation du coût alors on résout le problème dual en mettant à jour

les valeurs des multiplicateurs, sinon, on arrête. L'ordonnancement trouvé à la fin constituera une bonne borne inférieure aidant à la recherche des solutions admissibles.

V.4.1 Résolution du problème relaxé

Rappelons que nous cherchons à résoudre pour chaque travail j, le sous problème $L_j(\lambda,\mu)$. Pour cela, et pour chaque travail j, nous cherchons à appliquer la Programmation Dynamique en considérant la famille des sous-problèmes suivants, avec $1 \le l \le N_j$ et $1 \le \tau \le H$:

$$F_{l\tau} = \min\left(\sum_{j=1}^{n} W_j T_j^2 + \sum_{k=l}^{N_j} \sum_{t=S_{jk}}^{S_{jk}+p_{jk}-1} \lambda_{tm} + \sum_{t=S_{jl}}^{S_{jN_j}+p_{jN_j}-1} \mu_t\right)$$

sous les contraintes :

$$S_{jl} = \tau$$

$$S_{jk} + p_{jk} \le S_{j,k+1},\ \forall 1 \le k \le N_j$$

Cette famille peut être écrite de façon récursive. Ainsi :

$$F_{l\tau} = \min_{S_{j,l+1} \ge \tau + p_{jl}} \left(F_{l+1,S_{j,l+1}} + \sum_{t=\tau}^{\tau+p_{jl}-1} \lambda_{t,M_{jl}} + \sum_{t=\tau}^{S_{j,l+1}-1} \mu_t\right),$$

$$F_{l\tau} = \min_{y \ge \tau + p_{jl}} \left(F_{l+1,y} + \sum_{t=\tau}^{\tau+p_{jl}-1} \lambda_{t,M_{jl}} + \sum_{t=\tau}^{y-1} \mu_t\right),$$

$$F_{l\tau} = \min_{y \ge \tau + p_{jl}} \left(F_{l+1,y} - \sum_{t=y}^{H} \mu_t\right) + \sum_{t=\tau}^{\tau+p_{jl}-1} \lambda_{t,M_{jl}} + \sum_{t=\tau}^{H} \mu_t,$$

par conséquent, et pour chaque travail, la connaissance des solutions des sous-problèmes $F_{N_j,\tau}$ est suffisante pour déduire le reste des sous-problèmes, et justement :

$$F_{N_j,\tau} = W_j T_j^2 + \sum_{t=\tau}^{\tau+p_{jl}-1} (\lambda_{t,M_{jl}} + \mu_t)$$

En principe, les procédures par énumération sont coûteuses (en temps de calcul et/ou en espace de mémoire), mais l'exploitation de certaines caractéristiques mathématiques du problème nous fait éviter beaucoup de redondances dans les calculs.

Posons :

$$R_\tau = \sum_{t=\tau}^{H} \mu_t$$

$$P_\tau = \sum_{t=\tau}^{\tau+p_{jl}-1} \lambda_{t,M_{jl}}$$

$$Q_\tau = \min_{y \geq \tau+p_{jl}} \left(F_{l+1,y} - \sum_{t=y}^{H} \mu_t \right)$$

Donc :

$$F_{l,\tau} = Q_\tau + P_\tau + R_\tau$$

Le vecteur R_τ est calculé une seule fois pour tous les travaux, en utilisant le fait que :

$$\begin{cases} R_H = \mu_H \\ R_\tau = R_{\tau+1} + \mu_t \;\; \tau = H-1, H-2, \ldots, 2, 1 \end{cases}$$

Pour le vecteur Q_τ, et connaissant $F_{N_j,\tau}$, on peut déduire que :

$$Q_\tau = \min \left(Q_{\tau+1}, F_{l+1,\tau+p_{jl}} \right)$$

Une fois ce calcul fait, l'ordonnancement concernant le travail j est déduit par les relations :

$$S_{jl} = \arg\min_{\tau} \; F_{l\tau}$$

$$S_{j,l+1} = \inf_{y} \left\{ \; y \geq S_{jl} + p_{jl} \,/ F_{l+1,y} + R_{S_{jl}} - R_y = F_{lS_{jl}} \; \right\}$$

où argmin désigne l'argument qui minimise l'expression qui le suit.

A cette étape, et connaissant les variables de décision $S_{jk}(1 \leq j \leq n\ ;\ 1 \leq k \leq N_j)$, on doit :

1 – calculer le coût du sous-problème du niveau travail j : $L_j(\lambda,\mu) = F_{lS_{jl}}$,

2 – calculer le coût du problème :

$$L(\lambda,\mu) = -\sum_{t,m}\lambda_{tm} - \sum_{t}\mu_t + \sum_{j}L_j(\lambda,\mu)\ ,$$

3 – comparer ce coût avec celui de l'itération précédente. S'ils sont égaux (à une erreur près), alors arrêter ; sinon, passer à la résolution du problème dual.

V.4.2 Résolution du problème dual

La résolution du problème dual suppose la mise à jour des multiplicateurs de Lagrange de manière à maximiser le coût associé au problème relaxé. Pour cela, nous adoptons la méthode des sous-gradients qui renouvelle la valeur des multiplicateurs pour la relaxation :

$$\theta_i^{(n+1)} = \theta_i^{(n)} + \alpha.\nabla_i L_l^{(h)}$$

avec :

θ_i : un des multiplicateurs λ ou μ

n : le nombre d'itérations

$\nabla_i L_l$: le sous-gradient de L_l par rapport à θ_i

α : le pas de déplacement que nous définirons plus tard

ainsi.

$$\lambda_{tm}^{(n+1)} = \lambda_{tm}^{(n)} + \alpha_l g_{tm}(\lambda^{(n)},\mu^{(n)})$$

$$\mu_t^{(n+1)} = \mu_t^{(n)} + \alpha_l h_t(\lambda^{(n)},\mu^{(n)})$$

et sachant que

$$L(\lambda,\mu) = \min\Big(\sum_{j=1}^{n} W_j T_j^2 + \sum_{t,m}\lambda_{tm}\Big\{\sum_{j,k} 1(S_{jk} \leq t < S_{jk} + p_{jk}).\delta_{jk}^m - 1\Big\} + \sum_{t}\mu_t\Big\{\sum_{j} 1(S_{jl} \leq t < S_{j.N_j} + p_{j.N_j}) - L\Big\}$$

alors,

$$g_{tm}(\lambda^{(n)},\mu^{(n)}) = \frac{\partial L_1}{\partial \lambda_{tm}} = \sum_{j,k} 1(S_{jk}^{(n)} \le t < S_{jk}^{(n)} + p_{jk}).\delta_{jk}^{m} - 1$$

$$h_t(\lambda^{(n)},\mu^{(n)}) = \frac{\partial L_1}{\partial \mu_t} = \sum_{j} 1(S_{j1}^{(n)} \le t < S_{jN_j}^{(n)} + p_{jN_j}) - L$$

Nous sommes arrivés au choix du pas de déplacement. En effet, il nous faut un pas qui décroît au fur et à mesure que nous nous approchons de l'optimum, tout en assurant une vitesse de convergence raisonnable. Pour ce faire, nous adoptons la plus récente des méthodes des sous-gradients d'Armijo (Polak, 91).

Remarque :

Le coût du problème relaxé L est une fonction concave. En effet :

Soit $\lambda_1, \lambda_2, \mu_1, \mu_2 \ge 0$, et $0 \le \theta \le 1$. Soit λ_0, μ_0 tels que :

$$\lambda_0 = \theta\lambda_1 + (1-\theta)\lambda_2,$$

$$\mu_0 = \theta\mu_1 + (1-\theta)\mu_2.$$

$$L(\lambda_0,\mu_0) = -\sum_{t,m}\lambda_{tm} - L\sum_{t}\mu_t + \sum_{j=1}^{n}\left\{ (W_j T_j^2 + \sum_{k=1}^{N_j}\sum_{t=S_{jk}}^{S_{jk}+p_{jk}-1}\lambda_{tm}^{0} + \sum_{t=S_{j1}}^{S_{jN_j}+p_{jN_j}-1}\mu_t^{0})\right\}$$

donc :

$$L(\lambda_0,\mu_0) = \theta(-\sum_{t,m}\lambda_{tm}^{1} - L\sum_{t}\mu^{1} + \sum_{j=1}^{n}\left\{ (W_j T_j^2 + \sum_{k=1}^{N_j}\sum_{t=S_{jk}^{0}}^{S_{jk}^{0}+p_{jk}-1}\lambda_{tm}^{1} + \sum_{t=S_{jk}^{0}}^{S_{jN_j}^{0}+p_{jN_j}-1}\mu_t^{1})\right\}$$

$$+ (1-\theta)(-\sum_{t,m}\lambda_{tm}^{2} - L\sum_{t}\mu^{2} + \sum_{j=1}^{n}\left\{ (W_j T_j^2 + \sum_{k=1}^{N_j}\sum_{t=S_{jk}^{0}}^{S_{jk}^{0}+p_{jk}-1}\lambda_{tm}^{2} + \sum_{t=S_{jk}^{0}}^{S_{jN_j}^{0}+p_{jN_j}-1}\mu_t^{2})\right\}$$

par suite :

$$L(\lambda_0,\mu_0) \geq \theta\left(-\sum_{t,m}\lambda^1_{tm} - L\sum_t \mu^1 + \sum_{j=1}^{n}\left\{ W_j T_j^2 + \sum_{k=1}^{N_j}\sum_{t=S^0_{jk}}^{S^0_{jk}+p_{jk}-1}\lambda^1_{tm} + \sum_{t=S^0_{jk}}^{S^0_{jk}+p_{jN_j}-1}\mu^1_t \right)\right\}$$

$$+ (1-\theta)\left(-\sum_{t,m}\lambda^2_{tm} - L\sum_t \mu^2 + \sum_{j=1}^{n}\left\{ W_j T_j^2 + \sum_{k=1}^{N_j}\sum_{t=S^0_{jk}}^{S^0_{jk}+p_{jk}-1}\lambda^2_{tm} + \sum_{t=S^0_{jk}}^{S^0_{jk}+p_{jN_j}-1}\mu^2_t \right)\right\}$$

soit alors :

$$L(\lambda_0,\mu_0) \geq \theta.L(\lambda_1,\mu_1) + (1-\theta).L(\lambda_2,\mu_2)$$

Méthode d'Armijo

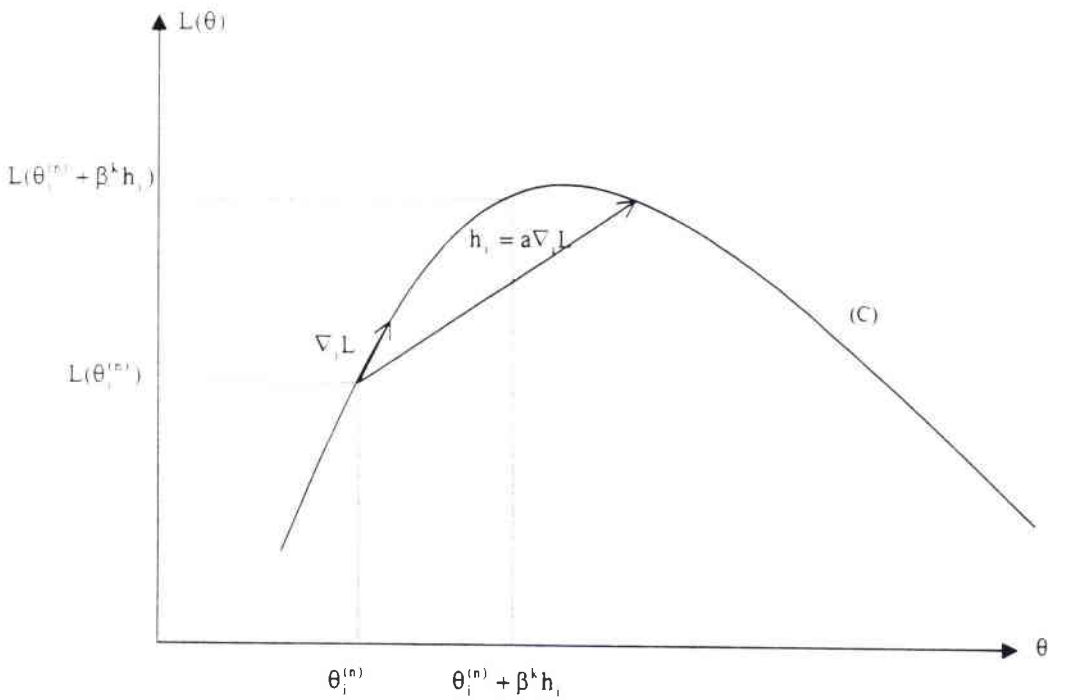

Figure V.1 : Courbe du coût relaxé

Soit (C) la courbe représentant le 'coût relaxé' en fonction du vecteur θ. L'idée est que, tant que $\nabla_i L \neq 0$, faire :

1- Tracer la droite (D) passant par le point de départ θ_i et de pente $a.\nabla_i L$, a étant un paramètre préfixé entre 0 et 1 et $\nabla_i L$ le sous gradient de L par rapport à θ_i.

2- Sachant que la partie admissible, c'est-à-dire la partie qui contient l'optimum, est la partie de la courbe (C) au-dessus de la droite (D), chercher le plus grand pas β^k

($\beta \in]0,1[$; $k \in \mathbb{Z}$), tel que $\theta_i + \beta^k.h_i$ sera dans la partie admissible, c'est à dire que l'inégalité sera vérifiée :

$$L(\theta_i + \beta^k h_i) - L(\theta_i) \geq a\,\beta^k \left\|\nabla L(\theta_i)\right\|^2$$

avec $h_i = a\nabla_i L$, la direction du sous-gradient ajustée. D'où, l'algorithme d'Armijo (Polak , 91).

Algorithme d'Armijo

Paramètres : $\alpha, \beta \in (0,1)$, $k^* \in Z$ et (λ_0, μ_0).

Etape 1 : Faire i = 0

Etape 2 : Calculer la direction de recherche $h_i = \alpha \nabla_i L_1(x_i)$

Etape 3 : Calculer le pas β_i qui vérifie :

$$L_1(\theta_i + \beta^{k_i} h_i) - L_1(\theta_i) \geq \alpha\beta^{k_i} \left\|\nabla L_1(\theta_i)\right\|^2$$

$$L_1(\theta_i + \beta^{k_i+1} h_i) - L_1(\theta_i) < \alpha\beta^{k_i+1} \left\|\nabla L_1(\theta_i)\right\|^2$$

Etape 4 : Faire $\theta_{i+1} = \theta_i + \beta^{k_i} h_i$, remplacer i par i+1 et retourner à l'étape 1.

Il y a deux problèmes qui apparaissent parfois avec cette méthode. Le premier est que notre position est encore loin de l'optimum alors que le pas de déplacement est faible. Le deuxième est le problème d'oscillation de la solution entre deux valeurs.

Pour remédier à ces problèmes, d'une part nous fixons un nombre d'itérations au bout desquelles si nous n'arrivons pas à améliorer notre solution, nous faisons décroître le nombre k et donc, nous augmentons le pas de déplacement du facteur $1/\beta$; d'autre part, nous cherchons, chaque fois qu'il est possible, à avoir le pas de déplacement le plus grand possible.

Rappelons que nous sommes en train de chercher un pas α tel que:

$$\theta_i^{(n+1)} = \theta_i^{(n)} + \alpha.\nabla_i L_1$$

$\theta_i^{(n-1)}$ étant positif, la valeur maximale que peut prendre le pas de déplacement est :

$$\alpha_{max} = \min_{i/\nabla_i L<0} \frac{\theta_i^{(n)}}{-\nabla_i L}$$

Alors, deux cas se présentent :

1^er cas : $\alpha_{max}>0$

si

$$L(\theta^{(n)} + \alpha_{max}\nabla L) > L(\theta^{(n)})$$

alors

$$\theta^{(n-1)} = \theta^{(n)} + \alpha_{max}\nabla L$$

sinon, appliquer la méthode d'Armijo.

2^ème cas : $\alpha_{max}=0$

$$h_i = \begin{cases} \nabla_i L & \text{si} \quad \theta_i^{(n)} > 0 \quad \text{ou} \quad \nabla_i L \geq 0 \\ 0 & \text{si} \quad \theta_i^{(n)} = 0 \quad \text{et} \quad \nabla_i L < 0 \end{cases}$$

Appliquer la méthode d'Armijo avec direction de recherche.

V.5 Construction d'un ordonnancement réalisable

La solution trouvée par relaxation Lagrangienne n'est pas nécessairement réalisable. En effet, il se peut que le meilleur coût soit obtenu en "abusant" de la capacité de certaines ressources, que a soient des machines ou des AGV.

Dans cette section, nous allons donc construire un ordonnancement réalisable à partir de la solution associée au problème dual. Pour cela, nous allons utiliser respectivement la Programmation Dynamique, un algorithme de liste et, éventuellement, l'algorithme du recuit simulé.

V.5.1 Programmation Dynamique

Comme première étape, nous allons utiliser la Programmation Dynamique pour déterminer une priorité pour chaque travail. Le critère de priorité utilisé sera que le travail le plus prioritaire est le travail au plus grand coût $L_j(\lambda^*,\mu^*)$. La Programmation Dynamique est utilisée de la même manière qu'avant, sauf qu'ici, chaque fois, on tient compte des travaux déjà ordonnancés.

L'algorithme à appliquer est alors le suivant :

1- Mettre les travaux à ordonnancer dans une première liste, $list_1$.

2- S'il n'y a plus de travaux dans $list_1$, alors arrêter.

3- Appliquer la Programmation Dynamique aux travaux figurant dans $list_1$ et calculer l'ordonnancement ainsi que les coûts associés aux travaux.

4- Prendre le travail au plus grand coût et l'arranger dans une liste $list_2$ avec la date de début associée.

La liste $list_2$ contient alors les travaux ordonnés dans le sens décroissant des priorités. Elle sera le point de départ de notre algorithme de liste.

V.5.2 Algorithme de liste

Dans l'ordre décrit par la liste $list_2$, nous choisissons un travail et, chaque fois qu'un AGV est disponible, nous ordonnançons les différentes opérations de ce travail tout en tenant compte de :

- la date de disponibilité des différentes opérations : pour la première opération, ce sera la date de disponibilité de l'AGV utilisé et, pour les autres opérations, ce sera la date de fin de l'opération juste avant, augmentée d'une unité.

- les contraintes de capacité des machines : en cherchant à ordonnancer les différentes opérations, il ne faut pas oublier qu'une machine ne peut traiter qu'une opération à la fois.

Sachant que L_1, la solution du problème dual, constitue une borne inférieure de la solution optimale recherchée J*, pour tout coût J correspondant à un ordonnancement réalisable, la relation suivante est vérifiée :

$$L_1 \leq J^* \leq J$$

Nous définissons le gap de dualité comme étant la quantité $\frac{J - L_1}{J}$ avec :

J : le coût de la solution admissible trouvée,

L_1 : le coût associé au problème relaxé.

Plus le coût de l'ordonnancement réalisable est proche du coût associé au problème dual (donc plus le gap de dualité est faible), plus on est sûr qu'on est proche de l'optimum et, donc, que la solution est bonne.

Pour des problèmes de petite taille, et en utilisant juste les procédures déjà décrites (la programmation dynamique suivie de l'algorithme de liste), les solutions trouvées sont bonnes (voir les exemples d'applications dans le paragraphe V.6), voire parfois optimales. Avec des problèmes de plus grande taille, l'erreur entre la solution admissible et la solution associée au problème dual devient plus grande ; dans ce cas, la solution trouvée constituera un point de départ pour une technique élaborée de recherche par voisinage qu'est celle du recuit simulé.

V.5.3 Technique du recuit simulé

Cette méthode consiste à parcourir aléatoirement un chemin dans le graphe de voisinage en admettant la possibilité de retenir un voisin moins bon pour éviter le piège des optimums locaux.

L'algorithme appliqué est alors de la forme :

a) Données :

$\{S^0_{jk}\}_{1\leq j\leq n;1\leq k\leq N_j}$: l'ordonnancement trouvé par application de la Programmation Dynamique, suivie de l'algorithme de liste,

L_0 : le coût associé à l'ordonnancement $\{S^0_{jk}\}_{1\leq j\leq n;1\leq k\leq N_j}$.

b) Paramètres :

$n \in \mathbb{N}$: le nombre d'itérations ou essais souhaités,

$\Upsilon \in]0,1[$: le facteur multiplicatif permettant de passer d'une itération à une autre,

P_r : la probabilité avec laquelle on retient une solution moins bonne.

c) Etapes :

1- Faire :

$$T_0 \leftarrow \frac{-L_0}{L_n(P_r)},$$

$T_f \leftarrow \gamma^n T_0$,

$T \leftarrow T_0$.

2- Générer au hasard une solution voisine $\{S^0_{jk}\}_{1\le j\le n;1\le k\le N_j}$ correspondant au coût L. La solution est voisine dans le sens où on prend au hasard deux travaux successifs et on inverse leur ordre d'exécution.

3- Si $L < L_0$, alors $S^0_{jk} \leftarrow S_{jk}$ et $S^{opt}_{jk} \leftarrow S^0_{jk}$, $\forall$ j, k; sinon générer au hasard un nombre $\lambda \in [0,1]$. Si $\lambda < e^{\frac{L-L_0}{T}}$, alors $S^0_{jk} \leftarrow S_{jk}$.

4- Faire $T \leftarrow \gamma T$.

5- Si $T > T_f$, alors retourner à l'étape 2, sinon arrêter.

Pour résoudre le problème d'ordonnancement par Relaxation Lagrangienne, nous avons d'abord relaxé le problème en introduisant les multiplicateurs de Lagrange λ et μ, ensuite résolu le problème relaxé par programmation dynamique, puis résolu le problème dual en utilisant la méthode d'Armijo et, enfin, construit un ordonnancement réalisable par Programmation Dynamique, méthodes sérielles et méthode du recuit simulé.

Cette approche a plusieurs avantages. En effet, elle permet de tenir compte de l'indisponibilité des machines et des moyens de transport pour l'exécution de certains travaux. Elle permet aussi, et c'est le plus important, d'évaluer l'ordonnancement réalisable obtenu car la comparaison de l'ordonnancement réalisable avec la solution duale est une majoration de l'erreur par rapport à la valeur optimale du critère.

V.6 Résultats Numériques

Après avoir implanté différentes procédures de résolution, à savoir l'approche de relaxation lagrangienne, un algorithme de liste et la technique du recuit simulé, nous nous proposons de vérifier l'efficacité de ces procédures à travers des exemples pratiques. Aussi, considérons-nous les exemples d'application suivants :

V.6.1 Exemple 1

Le premier exemple considéré est l'exemple N°1 dans Blazewicz et al. (96). Il s'agit d'ordonnancer un job-shop à trois machines, quatre travaux à pondérations égales à l'unité et

quatre AGV. Donc à priori, on est en présence d'un problème de job-shop ordinaire où il n'y a pas de problème de conflit sur les AGV. L'horizon de temps est 25 jours (c'est-à-dire que H=25 et que l'unité de temps est le jour). Toutes les machines sont disponibles à partir de la date 1 pour tout l'horizon de temps. La date de disponibilité pour tous les travaux est égale à 1.

Chaque travail est composé de trois opérations à réaliser sur les trois machines. Les données du problème d'ordonnancement sont décrites dans le tableau suivant :

Travaux	Opérations		
	O_{j1}	O_{j2}	O_{j3}
J_1	M_1, 4	M_2, 3	M_3, 2
J_2	M_2, 1	M_1, 4	M_3, 4
J_3	M_3, 3	M_2, 2	M_1, 3
J_4	M_2, 3	M_3, 3	M_1, 1

Table V.1: Travaux de l'exemple 1

Résultats

Le coût associé au problème dual est L_1=474.99. La solution réalisable trouvée (en appliquant seulement la Programmation Dynamique), dans un temps de résolution de 0.45s sur la station Alpha, est représentée dans la table V.2. La valeur associée du critère est de J = 475. Le gap de dualité est alors de 2.10^{-3} %.

Travaux	Opérations		
	O_{j1}	O_{j2}	O_{j3}
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1 , 4	5, 7	8, 9
J_2	1, 1	5, 8	10, 13
J_3	1, 3	8, 9	10, 12
J_4	2, 4	5, 7	9, 9

Table V.2 : Ordonnancement de l'exemple 1

La solution trouvée est optimale car on peut facilement la vérifier pour un problème de cette taille. En plus, elle est fournie dans un temps faible (0.45s).

La représentation graphique sur le diagramme de Gantt donne une idée de la répartition des machines entre les différentes opérations dans le temps. Pour les AGV, le

problème ne se pose pas pour cet exemple, puisque le nombre des AGV est le même que celui des travaux.

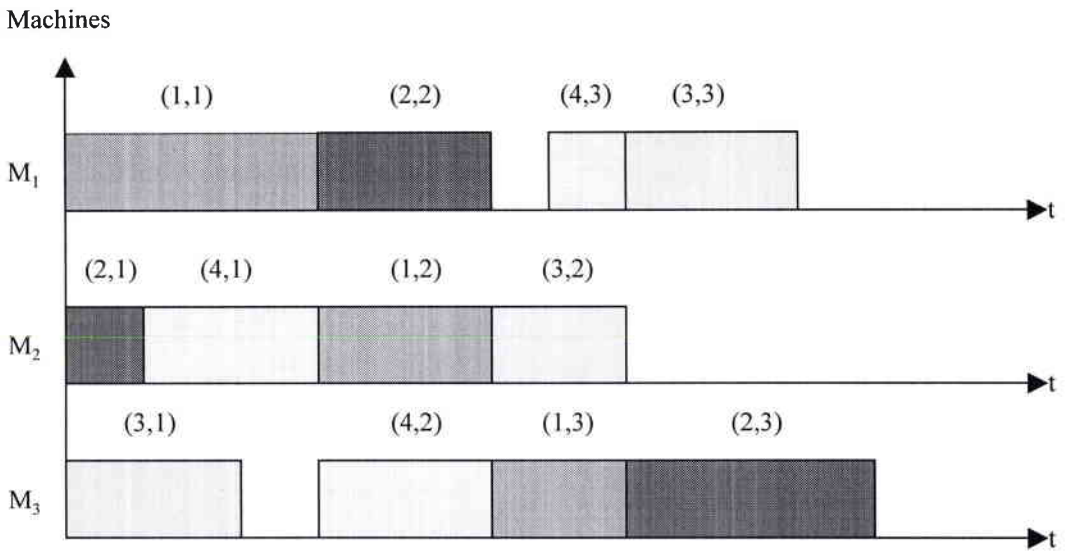

Figure V.2: Diagramme de Gantt représentant la solution de l'exemple 1

V.6.2 Exemple 2

Nous traitons le même exemple que l'exemple précédent, seulement cette fois-ci, le nombre des AGV est réduit à trois. Donc, on ne peut pas lancer à l'exécution plus de trois travaux à la fois. Les résultats trouvés sont représentés dans la table V.3.

Travaux	Opérations		
	O_{j1}	O_{j2}	O_{j3}
	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 4	6, 8	9, 10
J_2	9, 9	10, 13	14, 17
J_3	1, 3	4, 5	6, 8
J_4	1, 3	4, 6	9, 9

Table V.3 : Ordonnancement de l'exemple 2

Le coût associé au problème dual est L_l=531.01. A partir de ce coût, on trouve la solution admissible, représentée dans la table V.3. La solution est trouvée au bout de 1.58s et

elle correspond à une valeur de critère J = 534, d'où un gap de dualité égale à 0.56%. Le diagramme de Gantt de la solution trouvée est donné dans la figure V.3.

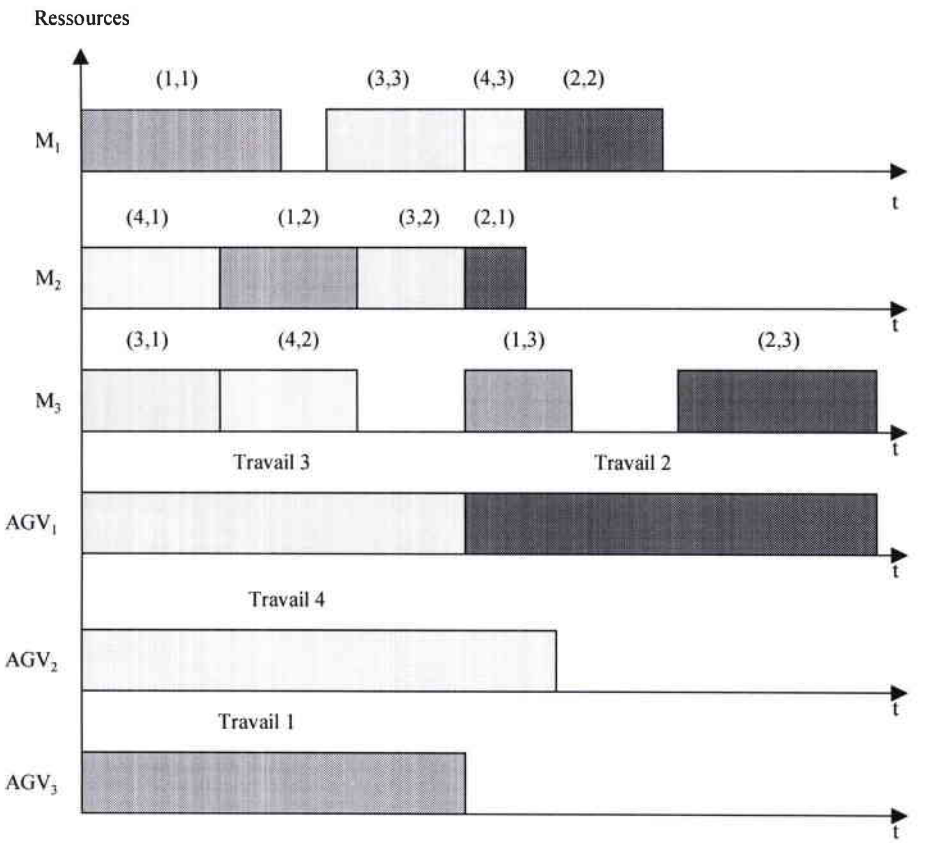

Figure V.3 : Diagramme de Gantt représentant la solution de l'exemple 2

V.6.3 Exemple 3

Le troisième exemple est généré aléatoirement. C'est encore un exemple de problème d'ordonnancement dans un job-shop ordinaire (sans les contraintes de moyens de transport) mais de taille plus grande. On a dix travaux, dix AGV et cinq machines. La $k^{ème}$ opération du travail j est réalisée sur la machine (j+k-1) mod 5. La durée opératoire de chaque opération et la date échue de chaque travail sont générés aléatoirement des intervalles [1,10] et [1,25] respectivement. Les pondérations des travaux sont prises égales à 1 et l'horizon de temps est égal à 200.

Les données de ce problème sont fournies par la table V.4, avec l'ordonnancement admissible trouvé. Les dates échues utilisées pour les travaux J_1 à J_{10} ont été 9, 24, 10, 24, 1, 13, 7, 1, 15 et 6, respectivement.

La solution du problème dual de ce problème est L_1=9319.92. Elle permet de trouver la solution représentée dans la table V.5, au bout de 15.53s. Le coût de cet ordonnancement est de J=11197, il en résulte un gap de dualité de l'ordre de 16.7 %.

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	M_1, 4	M_2, 8	M_3, 4	M_4, 5	M_5, 3
J_2	M_2, 8	M_3, 1	M_4, 6	M_5, 2	M_1, 4
J_3	M_3, 7	M_4, 1	M_5, 9	M_1, 2	M_2, 2
J_4	M_4, 5	M_5, 6	M_1, 6	M_2, 3	M_3, 4
J_5	M_5, 3	M_1, 7	M_2, 3	M_3, 1	M_4, 4
J_6	M_1, 8	M_2, 9	M_3, 6	M_4, 8	M_5, 1
J_7	M_2, 5	M_3, 5	M_4, 8	M_5, 3	M_1, 9
J_8	M_3, 3	M_4, 6	M_5, 6	M_1, 2	M_2, 8
J_9	M_4, 3	M_5, 2	M_1, 4	M_2, 4	M_3, 6
J_{10}	M_5, 7	M_1, 6	M_2, 5	M_3, 5	M_4, 6

Table V.4: Travaux de l'exemple 3

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	1, 4	6, 13	14, 17	23, 27	28, 30
J_2	37, 44	45, 45	48, 53	54, 55	56, 59
J_3	4, 10	11, 11	19, 27	28, 29	30, 31
J_4	12, 16	31, 36	38, 43	45, 47	48, 51
J_5	1, 3	5, 11	14, 16	18, 18	19, 22
J_6	30, 37	48, 56	57, 62	63, 70	71, 71
J_7	1, 5	27, 31	32, 39	40, 42	44, 52
J_8	1, 3	4, 9	13, 18	19, 20	21, 28
J_9	1, 3	4, 9	12, 15	17, 20	21, 26
J_{10}	6, 12	21, 26	32, 36	37, 41	42, 47

Table V.5 : Ordonnancement de l'exemple 3

Remarquons d'une part, que le gap de dualité exprimant une majoration de l'erreur par rapport à la solution optimale augmente avec la taille du problème et, d'autre part, que le temps de résolution devient plus important. En effet, sachant que l'espace de recherche augmente exponentiellement avec la taille du problème, la probabilité de trouver exactement l'optimum diminue.

V.6.4 Exemple 4

Nous traitons l'exemple précédent avec un nombre des AGV réduit à 3. Donc a priori, nous ne pouvons pas lancer plus de trois travaux à la fois

Les données de ce problème sont fournies par la table V.6 et l'ordonnancement est donné dans la table V.7.

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
J_1	M_1, 4	M_2, 8	M_3, 4	M_4, 5	M_5, 3
J_2	M_2, 8	M_3, 1	M_4, 6	M_5, 2	M_1, 4
J_3	M_3, 7	M_4, 1	M_5, 9	M_1, 2	M_2, 2
J_4	M_4, 5	M_5, 6	M_1, 6	M_2, 3	M_3, 4
J_5	M_5, 3	M_1, 7	M_2, 3	M_3, 1	M_4, 4
J_6	M_1, 8	M_2, 9	M_3, 6	M_4, 8	M_5, 1
J_7	M_2, 5	M_3, 5	M_4, 8	M_5, 3	M_1, 9
J_8	M_3, 3	M_4, 6	M_5, 6	M_1, 2	M_2, 8
J_9	M_4, 3	M_5, 2	M_1, 4	M_2, 4	M_3, 6
J_{10}	M_5, 7	M_1, 6	M_2, 5	M_3, 5	M_4, 6

Table V.6: Travaux de l'exemple 4

Travaux	Opérations				
	O_{j1}	O_{j2}	O_{j3}	O_{j4}	O_{j5}
	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f	t_s, t_f
J_1	79, 82	83, 90	91, 94	95, 99	100, 102
J_2	55, 62	63, 63	67, 72	73, 74	83, 86
J_3	34, 40	41, 41	42, 50	51, 52	53, 54
J_4	55, 59	60, 65	66, 71	72, 74	75, 78
J_5	34, 36	39, 45	46, 48	49, 49	60, 63
J_6	1, 8	9, 17	18, 23	24, 31	33, 33
J_7	1, 5	6, 10	11, 18	19, 21	22, 30
J_8	1, 3	4, 9	10, 15	16, 17	18, 25
J_9	64, 66	67, 68	72, 75	76, 79	80, 85
J_{10}	26, 32	33, 38	39, 43	44, 48	49, 54

Table V.7 : Ordonnancement de l'exemple 4

La solution du problème dual de ce problème est L_1=19989. La solution admissible, représentée dans la table V.7, est trouvée au bout de 42.15s. Elle correspond à une valeur de critère de J=29898, d'où un gap de dualité de l'ordre de 33.1 %.

Nous remarquons que le gap de dualité exprimant une majoration de l'erreur par rapport à la solution optimale, augmente avec la taille du problème. C'est le cas aussi pour le temps de résolution. En effet, l'espace de recherche augmente exponentiellement avec la taille du problème et donc, la probabilité de l'erreur dans le temps et dans l'espace augmente.

V.7 Vers une approche intégrée

Nous avons utilisé l'approche de relaxation lagrangienne dans le cadre particulier d'un job-shop dans lequel :

- Une machine ne peut traiter qu'une opération à la fois,
- Une opération ne peut être réalisée que sur une machine particulière,
- Les AGV sont identiques et peuvent transporter les différents travaux.

Ceci, en considérant, comme critère, la minimisation de la somme des retards pondérés.

Cependant, cela ne restreint pas le domaine d'application de cette approche. En effet, il peut s'étendre aux ateliers de type flow-shop dans le but d'optimiser un critère quelconque tel que le makespan, avec considération du cas général où :

- Chaque opération peut être réalisée sur un ensemble de machines ; donc, l'ensemble de variables de décision n'est plus restreint aux dates de début, il contient aussi les machines choisies pour traiter telle ou telle opération.

- Chaque machine a une certaine capacité éventuellement dépendante des périodes du temps.

- Les AGV ne sont pas identiques : chaque AGV a une certaine catégorie de travail qu'il peut transporter.

- Le temps d'indisponibilité des machines entre le traitement d'une opération et de l'opération suivante est fixé.

- Le temps d'indisponibilité des moyens de transport entre le déchargement d'un travail et le chargement du travail suivant est fixé.

V.8 Conclusion

L'objectif de ce chapitre a été l'analyse des problèmes d'ordonnancements dans les ateliers flexibles du type job-shop, en tenant compte des contraintes des moyens de transport. Après avoir modélisé mathématiquement le problème, nous avons proposé une méthodologie de résolution basée sur l'approche de Relaxation Lagrangienne, en prenant comme critère la minimisation de la somme des retards pondérés. L'utilisation de cette approche suppose :

- la relaxation des contraintes de capacité des ressources en machines et en moyens de transport en introduisant les multiplicateurs de Lagrange λ et μ.

- La résolution du problème relaxé en le décomposant à des sous-problèmes dont chacun est résolu par Programmation Dynamique.

- La résolution du problème dual en utilisant la plus récente des méthodes d'Armijo.

La solution fournie par cette méthode est non nécessairement réalisable ; aussi, cherchons-nous à construire une solution admissible en utilisant la Programmation Dynamique, une méthode de liste et, éventuellement, la technique du recuit simulé.

Enfin, la dernière partie de ce paragraphe a revêtu un caractère pratique. Nous avons implanté l'approche de Relaxation Lagrangienne et traité des exemples d'ateliers pour mieux en illustrer l'efficacité.

En guise de conclusion, nous avons examiné la possibilité de généralisation de cette approche, en jouant soit sur la nature des ressources, soit sur le type des ateliers.

Conclusion générale

Conclusion générale

Dans cette thèse, nous avons introduit les job-shops Multi-Ressources avec Blocage (job-shops MRB) pour prendre en compte la variété de ressources impliquées dans une production et le phénomène de blocage. Les job-shops MRB sont une extension des job-shops classiques enrichis de deux caractéristiques nouvelles : les opérations à multi-ressources et la propriété "Retenir et Attendre" pour la succession des opérations. Nous avons montré que les job-shops MRB constituent un cas général pour étudier l'ordonnancement des systèmes de production.

Nous avons conduit, dans le chapitre II, une étude détaillée de la complexité du problème d'ordonnancement des job-shops MRB en nous appuyant sur des résultats existants pour des cas classiques et en développant des résultats nouveaux pour les autres cas. Cette étude montre que l'ordonnancement des job-shops MRB est un problème difficile car il devient très vite NP-difficile au sens fort. Cette NP-complétude conditionne la suite de la thèse.

Avant de passer à la résolution du problème d'ordonnancement d'un job-shop MRB général, nous avons abordé la résolution du problème d'ordonnancement d'un job-shop MRB dans lequel toutes les ressources sont différentes, c'est-à-dire un Système à Ressources Unitaires (SRU). Nous avons montré que, pour les systèmes simples, les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage. Les RdP ont été utilisés pour modéliser un SRU et l'ordonnancement, ce qui a permis la vérification de blocage pour un état donné et la vérification de la faisabilité d'un ordonnancement donné. Ensuite, nous avons proposé deux méthodes heuristiques gloutonnes, utilisant les RdP pour la vérification de blocage, pour obtenir des ordonnancements sans blocage. Nous avons également proposé une méthode fondée sur les systèmes simples "équivalents" pour obtenir l'ordonnancement d'un SRU général.

Dans le chapitre IV, nous avons étendu les méthodes proposées dans le chapitre III, pour construire, dans ce cas, l'ordonnancement d'un job-shop MRB. Comme dans les SRU, nous avons démontré que les contraintes de précédence et de capacité des ressources impliquent l'absence de blocage pour les systèmes simples. Nous avons montré, à l'aide des modèles RdP généralisés des job-shops MRB, que la vérification de blocage devient combinatoire pour les job-shops MRB généraux. En tenant compte de cette complexité, nous avons proposé une méthode pour construire un ordonnancement d'un job-shop MRB général en passant par son système simple "équivalent".

Dans le chapitre V, nous nous sommes limités à une classe particulière des job-shops MRB, les job-shops classiques mais dont chaque travail nécessite un moyen de transport (AGV) tout au long de sa réalisation. L'objectif était d'étudier l'efficacité de la méthode Relaxation Lagrangienne pour l'ordonnancement des job-shops MRB, en suivant la méthodologie proposée par Luh et Hoitomt (93). Nous avons proposé des méthodes efficaces pour la résolution du problème relaxé, la résolution du problème dual et la construction d'un ordonnancement admissible

Le résultats numériques des chapitres III, IV et V sont encourageants et les temps de calcul des méthodes proposées sont faibles.

Au vu des résultats de cette thèse, une première extension intéressante pour la recherche future est la prise en compte de gammes de produits non linéaires comprenant des opérations d'assemblage et/ou désassemblage. Une autre recherche future intéressante est la prise en compte de la flexibilité des ressources et des gammes de produits. Le développement d'une méthode Relaxation Lagrangienne pour construire l'ordonnancement d'un job-shop MRB quelconque est une extension naturelle du chapitre V.

Plus largement, il serait intéressant d'étudier l'ordonnancement des RdP généraux en suivant la méthodologie développée dans cette thèse. Les résultats de cette thèse montrent que les RdP sont un bon outil pour la vérification de blocage. Par contre, leur utilisation dans la recherche d'ordonnancements efficaces reste ouverte. Comme nous l'avons constaté dans le chapitre I, les RdP ont prouvé leur efficacité dans la prévention, la détection et l'élimination de blocages. Une recherche future intéressante serait la prise en compte de la politique de prévention ou d'élimination de blocage, dans l'ordonnancement d'un RdP général.

Bibliographie

BIBLIOGRAPHIE

A

(Aarts et Korst, 89) : E. H. L. Aarts, J. H. M. Korst. *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester.

(Adams et al., 88) : J. Adams, E. Balas, D. Zawak. "The Shifting Bottleneck Procedure for Job-Shop Scheduling", *Management Science*, 34, pp.391-401.

(Antony, 65) : R. N. Antony. *Planning and Control Systems : A Framework for Analysis*, Harvard University Press, Cambridge, Massachusets, USA.

B

(Baccelli et al., 92) : F. Baccelli, G. Cohen, B. Gaujal. "Recursive Equations and Basic Properties of Timed Petri Nets", *Discret Event Dynamic Systems : Theory and Applications 1*, pp. 415-439.

(Baker, 74) : K. R. Baker. *Introduction to sequencing and Scheduling*, Wiley.

(Balas et Vazacopoulos, 95) : E. Balas, A. Vazacopoulos. "Guided local search with shifting bottleneck for job shop scheduling", Management Science Research Report #MSRR-609.

(Banaszak et Krogh, 90) : Z. Banaszak, B. H. Krogh. "Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows", *IEEE Transactions on Robotics and Automation*, vol. 6, No. 6.

(Barkaoui et Ben Abdallah, 94) : K. Barkaoui, I. Ben Abdallah. "An Efficient Deadlock Avoidance Control Policy in FMS Using Structural Analisys of Petri Nets", IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, Texas, USA.

(Bénassy, 87) : J. Bénassy. *La gestion de production*, Hermes, Paris.

(Bitran et Hax, 77) : G. R. Bitran, A. C. Hax. "On the design of Hierarchical Production Planning Systems", *Decison Sciences*, vol. 8, No. 1.

(Blazewicz et al., 96) : J. Blazewicz, W. Domschke, E. Pesch. " The job shop scheduling problem: Conventional and new solution techniques ", *European Journal of Operational Research*, 93, pp. 1-33.

(Bracker et Chapman, 85) : W. E. Bracker, W. Chapman. "Optimization of a programable hoist using linear programing" , *Printed Circuit Fabrication*, vol. 8, pp. 39-42.

(Brucker et Krämer, 95) : P. Brucker, A. Krämer. "Shop scheduling problem with multiprocessor tasks on dedicated processors", *Annals of Operation Research*, 57, pp. 13-27.

C

(Campbell et al., 70) : H. G. Campbell, R. A. Dudeck, M. L. Smith, "A Heuristique Algorithm for the n-Job, m-machine sequencing problem", *Management Science*, 16-10, pp. 630-637.

(Camus, 97) : H. Camus. "Conduite de Systèmes Flexibles de Production Manufacturière par Composition de Régimes Permanents Cycliques : Modélisation et Evaluation de Performances à l'aide des Réseaux de Petri". Thèse de Doctorat de l'Université des Sciences et Technologies de Lille, France.

(Carlier, 82) : J. Carlier. "The one-machine sequencing problem", *European Journal of Operational Research*, 11, pp. 42-47.

(Carlier et Chrétienne, 88) : J. Carlier, P. Chrétienne. *Problèmes d'ordonnancement: modélisation / compléxité/ algorithmes*, Masson, Paris.

(Carlier et Pinson, 89) : J. Carlier, E. Pinson. "An Algorithm for solving the Job-Shop Problem", *Management Science*, vol.35, pp. 164-176.

(Caux et al., 93) : C. Caux, H. Pierreval, M. C. Portmann. "Les algorithmes génétiques et leur application aux problèmes d'ordonnancement", *Revue APII.*, 29, pp. 151-161.

(Chang et Liao, 94) : S.-C. Chang, D.-Y. Liao. "Scheduling Flexible Flow Shops with No Setup Effects", *IEEE Transactions on Robotics and Automation*, vol. 10, No. 2, pp. 112-122.

(Chen et al., 94) : H. Chen, C. Chu, J.-M. Proth. "Cyclic Scheduling of a hoist time window constraints", Rapport de Recherche, No. 2307, INRIA-Lorraine, France.

(Cho et al., 95) : H. Cho, T. K. Kumaran, R. A. Wysk. "Graph-Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems", *IEEE Transactions on Robotics and Automation*, vol. 11, No. 3, pp. 413-421.

(Chu et Xie, 97) : F. Chu, X. -L. Xie. "Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming", *IEEE Transactions on Robotics and Automation*, vol. 13, No. 6, pp. 793-804.

(Coffman et al., 71) : E. G. Coffman Jr., M. Elphiek, A. Shoshani. "System deadlocks", *Comput. Surveys*, vol. 3, No . 2, pp. 67-78.

(Collart-Dutilleul et Denat, 97) : S. Collart-Dutilleul, J. -P. Denat. "P-Time Petri nets and robust control of electroplating lines using several hoists", IFAC-IFIP-IMACS-CIS'97, Belfort, France, pp. 501-506.

(Czerwinski et Luh, 94) : C. S. Czerwinski, P. B. Luh. "Scheduling Products with Bills of Materials Using an Improved Lagrangian Relaxation Technique", *IEEE Transactions on Robotics and Automation*, vol. 10, No. 2.

D

(Dallery et al., 90) : Y. Dallery, R. David, X.-L. Xie. "Approximate Analysis of Transfer Lines with Unreliable Machines and Finite Buffers", *IEEE Transactions on Automatic Control*, vol. 34, pp. 943-953.

(Dallery et Gershwin, 92) : Y. Dallery, S. B. Gershwin. "Manufacturing Flow Line Systems : A Review of Models and Analytical Results", *Queueing Systems*, vol. 12.

(Damasceno et Xie, 98-a) : B. C. Damasceno, X. Xie. "Deadlock-free scheduling of Manufacturing Systems with Multiples Resources", Proc. INCOM 98 ("9th. Symposium on Information Control in Manufacturing"), Metz-Nancy, France.

(Damasceno et Xie, 98-b) : B. C. Damasceno, X. Xie."Integration of scheduling and deadlock avoidance of generalized job-shops with multiples resources", Proc. EURO XVI (16th. European Conference on Operational Research), Brussel, Belgium.

(Damasceno et Xie, 98-c) : B. C. Damasceno, X. Xie. "Scheduling and Deadlock Avoidance of a Flexible Manufacturing System", Proc. IEEE-SMC 98 (IEEE International Conference on Systems, Man and Cybernetics), San Diego, USA.

(Damasceno et Xie, 99) : B. C. Damasceno, X. Xie. "Deadlock-free Scheduling of Manufacturing Systems Using Petri Nets and Dynamic Programming", accepted to 14th IFAC World Congress, China.

(Dantizig et al., 54) : G. B. Dantizig, D. R. Fulkerson, S. M. Johnson. "Solution of a Large-Scale Traveling-Salesman Problem", *Operations Research*, vol.2, pp. 393-410.

(Dantizig, 63) : G. B. Dantizig. *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, USA.

(Dauzère-Pérès et Paulli, 97) : S. Dauzère-Pérès, J. Paulli. "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search", *Annals of Operational Research*, 70, pp. 281-306.

(Dauzère-Pérès et al., 98) : S. Dauzère-Pérès, W. Roux, J. B. Lasserre. "Multi-ressource shop scheduling with ressource flexibility", *European Journal of Operational Research*, vol. 107, No. 2, pp. 289-305.

(Di Cesare et al., 93) : F. Di Cesare, G. Harhalakis, J. -M. Proth, M. Silva, F. Vernadat. *Practice of Petri Nets in Manufacturing*, Chapman & Hall.

(Drozdowski, 96) : M. Drozdowski. "Scheduling Multiprocessor tasks - An overview", *European Journal of Operational Research*, 94, pp. 215-230.

(Du et Leung, 89) : J. Du, J. Y. -T. Leung. "Complexity of Scheduling Parallel Task Systems", *SIAM J. Disc. Math.*, vol. 2, No. 4, pp. 473-487.

E

(Elkarama, 98) : L. Elkarama. "Ordonnancement de la production d'ateliers flexibles en tenant compte des moyens de transport", Projet de fin d'études, avant-projet MACSI, INRIA-Lorraine, Metz, France.

(Ezpeleta et al., 95) : J. Ezpeleta, J. M. Colom, J. Martínez. "A Petri net Based Deadlock Prevention Policy for Flexible Manufacturing Systems", *IEEE Transactions on Robotics and Automation*, vol. 11, No. 2, pp. 173-184.

F

(Fanti et al., 97) : M. P. Fanti, B. Maione, S. Mascolo, B. Tuchiano. "Event-Based Feedback Control for Deadlock Avoidance in Flexible Production Systems", *IEEE Transactions on Robotics and Automation*, vol. 13, No. 3, pp. 347-363.

G

(Garey et Johnson, 79) : M. R. Garey, D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

(Gendreau et al., 94) : M. Gendreau, A. Hertz, G. Laporte. "A Tabu Search Algotithm for the Vehicle Routing Problem". *Management Science*, 40, pp. 1276-1290.

(Gershwin, 89) : S. B. Gershwin. "Hierarchical Flow Control : A Framework for Scheduling and Planning Discret Events in Manufacturing Systems", *Proc. IEEE*, 77(1), pp. 195-209.

(Giard, 88) : V. Giard. *Gestion de la Production*, Economica, Paris, 2eme édition.

(Gilmore et Gomory, 64) : P. C. Gilmore, R. E. Gomory. "Sequency a One-State-variable Machine: A Solvable Case of the Travelling Salesman Problem", *Operations Research*, 12, pp. 655-679.

(Glover, 86) : F. Glover. "Future paths for integer programming and links for artificial intelligence", *Computers & Operations Research*, 13, pp. 533-549.

(Glover et al., 92) : F. Glover, E. Taillard, D. de Werra. "A User's Guid to Tabu search", *Annals of Operations Research*, 41, pp. 3-28.

(Gomory, 58) : R. E. Gomory. "Outline of an Algorithm for Integer Solutions to Linear Programs", *Bulletin of the American Mathematical Society*, vol. 64, pp. 275-278.

(GOThA, 93) : GOThA. "Problèmes d'Ordonnancement", *Recherche Opérationnelle / Operational Research*, vol. 27, No. 1, pp. 77-150.

H

(Hall et Sriskandarajah, 96) : N. G. Hall, C. Sriskandarajah. "A Survey of Machine Scheduling Problems with blocking and no-wait in Process", *Operations Research*, vol. 44, No. 3.

(Hall et al., 97) : N. G. Hall, H. Kamoun, C. Sriskandarajah. "Scheduling in Robotic Cells: Classification, two and three Machine Cells", *Operations Research*, vol. 45, No. 3.

(Hax et Meal, 75) : A. C. Hax, H. C. Meal. "Hierarchical integration of production planning and scheduling", *Studies in Management Sciences*, *Logistics*, vol. I, pp. 53-69.

(Holland, 75) : J. H. Holland. *Adaptation in Natural and Artificial Systems*, the University of Michigan press: Ann Arbor, MI.

(Hoogeveen et al., 94) : J. A. Hoogeveen, S. L. Van de Velde, B. Veltman. "Complexity of scheduling multiprocessor tasks with prespecified processor allocations", *Discrete Applied Mathematics*, 55, pages 259-272, 1994.

(Hsieh et Chang, 94) : F. -S. Hsieh, S. -C. Chang. "Dispatching-Driven Deadlock Avoidance Controller Synthesis for Flexible Manufacturing Systems. *IEEE Transactions on Robotics and Automation*, vol. 10, No. 2, pp. 196-209.

J

(Jeng et al., 96) : M. D. Jeng, S. C. Chen, C. S. Lin. " A Search Approach based on Petri Net Theory for FMS Scheduling", 13th IFAC World Congress, San Francisco, CA, USA, vol. B, pp. 55-60.

K

(Kirkpatrick et al., 83) : S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi. "Optimization by Simulated Annealing", *Science*, vol. 220, No. 4598.

(Ku et Karim, 90) : H. -M. Ku, I. Karim. "Completion Time Algorithms for Serial Multiproduct Batch Processes with Shared Storage", *Computers Chem. Engr.*, vol. 14, No. 1, pp. 49-69.

(Kusiac, 90) : A. Kusiak. *Intelligent Manufacturing Systems*, Englewood Clifs, Prentice Hall, New Jersey, USA.

L

(Landweber et Robertson, 78) : L. H. Landweber, E. L. Robertson. "Properties of Conflit-Free and Persistent Petri Nets", *Journal of Association for Computing Machinery*, vol. 25, No. 3, pp. 352-364.

(Laurière, 87) : J. -L. Laurière. *Intelligence Artificielle : résolution de problèmes par l'Homme et la machine*, Eyrolles.

(Lawler, 77) : E. L. Lawler. "A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness", *Ann. Discret Math.*, 1, pp. 331-342.

(Lee et Di Cesare, 94-a) : D. Y. Lee, F. Di Cesare, " Scheduling Flexible Manufacturing Systems using Petri nets and Heuristic Search", *IEEE Transactons on Robotics and Automation*, vol. 10, No. 2.

(Lee et Di Cesare, 94-b) : D. Y. Lee, F. Di Cesare. " Integrated Scheduling of Flexible Manufacturing Systems Employing Automated Guided Vehicles", *IEEE Transactons on Industrial Electronics*, vol. 41, No. 6.

(Lei et Wang, 89) : L. Lei, T. Wang. "On the optimal cyclic schedules of single hoist electroplating processes", working paper # 89-0006, Rutgers University.

(Le Moal et Tarondeau, 79) : P. Le Moal, J. C. Tarondeau. "Un défi à la fonction production", *revue française de gestion*, pp. 9-14.

(Lenstra, 83) : J. K. Lenstra. "Integer Programming with fixed number of variables", *Mathemetics of Operation Research*, vol. 8, pp. 538-548.

(Lenstra et al., 77) : J. K. Lenstra, A. H. G. Rinnooy Kan, P. Brucker. "Complexity Machine Scheduling Problems", *Ann. Discrete Math.*, 1, pp. 343-362.

(Lin, 64) : S.Lin. "Computer Solutions of the Traveling Salesman Problem", *Bell Systems Technical Journal*, 44, pp. 2245-2269.

(Luh et Hoitomt, 93) : P. B. Luh, D. J. Hoitomt, " Scheduling of Manufacturing Systems Using the Lagrangian Relaxation Technique ", *IEEE Transactions on Automatic Control*, vol. 38, No. 7.

(Lundy et Mees, 86) : M. Lundy, A. Mees. "Convergence of Annealing Algorithm", *Mathematical Programming*, 34, pp. 111-124.

M

(Manier et Baptiste, 94) : M. -A. Manier, P. Baptiste. "Etat de l'art: ordonnancement de robots de manutention en galvanoplastie", *APII*, vol. 28, pp. 7-35.

(Minoux, 83) : M. Minoux. *Programmation Mathématique : théorie et algorithmes*, Dunod, Paris.

(Murata, 89) : T. Murata. "Petri Nets : Properties, Analysis and Applications", *Proceedings of IEEE*, vol. 77, No. 4, pp. 541-580.

P

(Philips et Unger, 76) : L. W. Philips, P. S. Unger. "Mathematical programing solution of a hoist scheduling program", *AIIE Transactions*, vol. 8, No. 2, pp. 219-225.

(Polak, 91) : E. Polak, " Optimization - Algorithms and Consistent Approximations", *Applied Mathematical Sciences* - vol. 124, ed. Springer- Verlag, USA.

(Potts et Van Wassenhove, 82) : C. N. Potts, L. N. Van Wassenhove. "A decomposition algorithm for the single machine total tardiness problem", *Operations Research Lett.*, pp. 177-181.

(Proth et al., 97) : J. -M. Proth, L. -M. Wang, X. Xie. "A Class of Petri Nets for Manufacturing System Integration", *IEEE Transactons on Robotics and Automation*, vol. 13, No. 3, pp. 317-326.

(Proth et Xie) : J. -M. Proth, X. Xie. *Petri Nets - A Tool for Design and Management of Manufacturing Systems*, John Wiley & Sons.

R

(Ramaswany et Joshi, 96) : S. E. Ramaswany, S. B. Joshi. " Deadlock-free Schedules for Automated Manufacturing Workstations", *IEEE Transactons on Robotics and Automation*, vol. 12, No. 3.

(Ramchandani, 74) : C. Ramchandani. "Analysis of Asynchronous Concurrent Systems Using Petri Nets", Rapport Technique No. 120, Laboratory for Computer Science, MIT, Cambridge, MA.

(Reveliotis et al., 97) : S. A. Reveliotis, M. A. Lawley, P. M. Ferreira. "Polynomial-Complexity Deadlock Avoidance Policies for Sequential Ressource Allocation Systems", *IEEE Transactions on Robotics and Automation*, vol. 42, No. 10, pp. 1344-1357.

(Richard, 97) : P. Richard. " Contribution des réseaux de Petri à l'étude de problèmes de recherche opérationnelle", Thèse de Doctorat de l'Université de Tours, France.

(Rinnooy Kan, 76) : A. H. G. Rinnooy Kan. *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague.

(Rippin, 93) : D. W. T. Rippin. "Batch Process Systems Engineering: A Retrospective and Prospective Review", *Computers Chim. Engr.*, vol. 17, Suppl., pp. S1-S13.

S

(Sahinidis et Grossmann, 91) : N. V. Sahinidis, I. E. Grossmann. "Reformulation of Multiperiod Milp Models for Planning and Scheduling of Chemical Process", *Computers Chem. Engr.*, vol. 15, No. 4, pp. 255-272.

(Savi, 94) : V. M. Savi. "Conception préliminaire des systèmes de production à l'aide des réseaux de Petri: évaluation des performances", Thèse de Doctorat de l'Université de Metz, France.

(Sawik, 90) : T. Sawik. "Modelling and scheduling of a Flexible Manufacturing System", *European Journal of Operational Research*, 45, pp. 77-190.

(Schrage et Baker, 78) : L. Schrage, K. R. Baker. "Dynamic programming solution of sequencing problems with precedence constraints". *Operations Research*, 26, pp. 444-449.

(Sethi et al., 92) : S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, W. Kubiak. "Sequencing of Parts and Robot Moves in a Robotic Cell", *International Journal of Flexible Manufacturing Systems*, 4, pp. 331-358.

(Shapiro et Nuttle, 88) : G. W. Shapiro, H. L. W. Nuttle. "Hoist schedulong for a PCB electroplating facility". *IIE Transactions*, vol. 20, No. 2, pp. 157-167.

(Sifakis, 78) : J. Sifakis. "Use of Petri Nets for Performance Evaluation", *Acta Cybernet*, vol. 4, No. 2, pp. 185-202.

(Sifakis, 80) : J. Sifakis. "Performance Evaluation of Systems using Petri Nets", Net Theory and Application, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, FRG.

(Slot et Vliet, 94) : P. Slot, M. van Vliet. "Analytical Model for FMS Design Optimization : A Survey", *The International Journal of Flexible Manufacturing Systems*, 6, pp. 209-233.

(Soriano et Gendreau, 97) : P. Soriano, M. Gendreau. "Fondements et Applications des Méthodes de Recherche avec Tabous". *Recherche Opérationnelle/Operations Research*, vol. 31, No. 2, pp. 133-159.

(Steck, 85) : K. E. Steck. "Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems", *Annals of Operations Research*, 3, pp. 3-12.

T

(Tomastik et al., 96) : R. N. Tomastik, P. B. Luh, G. Liu. "Scheduling Flexible Manufacturing Systems for Apparel Production", *IEEE Transactions on Robotics and Automation*, vol. 12, No. 5, pp. 789-799.

V

(Van Larhoven et al., 92) : P. J. M. Van Laarhoven, E. H. L. Aarts, J. K. Lenstra. "Job-Shop Scheduling by Simulated Annealing", *Operations Research*, vol. 40, No. 1, pp. 113-125.

(Vignier et al., 97) : A. Vignier, J.-C. Billault, C. Proust. "Les problèmes de flow-shop hybrides : état de l'art", *RAIRO-RO, sous presse*, 1997.

(Viswanaradham et al., 90) : N. Viswanaradham, Y. Narahari, T. L. Johnson. "Deadlock prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri net Models", *IEEE Transactions on Robotics and Automation*, vol. 6, No. 6.

W

(Wang, 95) : L. Wang. "Gestion Hiérarchisée de Systèmes de Production Discrets: Une Approche Basée sur les Réseaux de Petri", Thèse de Doctorat de l'Université de Metz, France.

(Wang et Xie, 96) : L. -M. Wang, X. -L. Xie. "Modular modeling using Petri nets", *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 800-809.

X

(Xie, 89) : X. Xie. "Contrôle Hierarchique d'un Système de Production soumis à Perturbations", Thèse de Doctorat de l'Université de Nancy I, France.

(Xie, 95) : X. Xie. "Analyse, Conception et Contrôle des Systèmes de Production", Habilitation à Diriger des Recherches en Sciences, Université de Metz, France.

(Xing et al., 96) : K. Y. Xing, B. S. Hu, H. X. Chen. "Deadlock Avoidance Policy for Petri Net Modeling of Flexible Manufacturing Systems with Shared Resources", *IEEE Transactions on Automatic and Control*, vol. 41, No. 2, pp. 289-295.

(Xiong et Zhou, 98) : H. Xiong, M. C. Zhou. "Scheduling of Semiconductor test facility Petri nets and Hybrid Heuristic Search", *IEEE Transactions on semiconductor Manufacturing* , vol. 11, No. 3, pp. 384-393.

Z

(Zuberek, 95) : W. M. Zuberek. "Application of timed Petri nets to modeling and analysis of flexible manufacturing cells", Technical Report #9503, Department of Computer Science, Memorial University of New Foundland, St. John's, NF, Canada

UNIVERSITE DE METZ

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Nom et Prénom de l'auteur : Mademoiselle CAMARGO DAMASCENO Bérénice

Titre de la thèse : « Ordonnancement des systèmes de production multi-ressources avec la prise en compte de blocages. »

Date de soutenance : le 4 mars 1999

Président du jury : M. C. PROUST

Membres du jury : M.M. BAPTISTE, DALLERY, DAUZERE-PERES, GENTINA, PEREIRA DE OLIVEIRA, PROUST et XIE

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

M. DAUZERE-PERES absent

Résumé

L'automatisation des systèmes de production conduit à l'utilisation d'une grande variété de ressources de fabrication ; un aspect longtemps négligé dans la planification et l'ordonnancement de production et qui pose un problème de coordination de l'ensemble de ressources. Dans ce travail, nous proposons une approche intégrée d'ordonnancement qui prend en compte simultanément cette variété de ressources et le problème de blocage. Pour cela, nous proposons un modèle d'ordonnancement, appelé job-shop Multi-Ressources avec Blocage ou job-shop MRB. Les deux caractéristiques saillantes du modèle que nous proposons sont : (i) les opérations nécessitant simultanément des ressources de différents types que nous appelons opérations à ressources multiples ; (ii) la contrainte "retenir et attendre" pour le passage d'une opération à l'opération suivante du même travail, c'est-à-dire que les ressources nécessaires pour une opération ne sont libérées qu'au début de l'opération suivante. Nous montrons que le problème est fortement combinatoire. Pour cela, nous proposons des méthodes heuristiques, utilisant les Réseaux de Petri pour la détection de blocage et la programmation dynamique, pour construire des ordonnancements sans blocage efficaces dans un temps raisonnable. Enfin, nous considérons un cas particulier : les job-shops avec moyens de transport et proposons une méthode de relaxation Lagrangienne pour construire des ordonnancements efficaces. Les résultats numériques obtenus attestent l'efficacité des méthodes proposées.

Mots-clés :

Ordonnancement, Systèmes de Production, Opérations à Ressources Multiples, Réseaux de Petri, Blocage.

Abstract

This work is motived by the extensive use of a wide range of manufacturing resources in today's production systems. We propose an integrated approach for scheduling and deadlock avoidance of such systems. For this purpose, we propose a scheduling model, called job-shop with Multiple Resources and Blocking. In this model, an operation may need more than one resource and it is called Multi-Resources operation. Further, upon the completion of an operation of the same job cannot be released and the remaining resources are released at the beginning of the next operation. The later characteristic is called "hold while waiting" property. The scheduling problem consists in sequency the operations such that deadlocks are avoided and the total duration, i.e. makespan, is minimized. We prove that the problem is strongly NP-Hard. This leads us to propose efficient heuristic methods to construct deadlock-free schedules. Finally, we consider a class of job-shops with Multiple Resources and Blocking : the job-shops with AGV. A Lagrangien Relaxation approach is proposed to solve the scheduling problem. Numerical results are given to show the efficiency the proposed methods.

Key words

Scheduling, Manufacturing Systems, Multi-Resources Operations, Petri Nets, Deadlock.