RAIRO. RECHERCHE OPÉRATIONNELLE

GOTHA

Les problèmes d'ordonnancement

RAIRO. Recherche opérationnelle, tome 27, n° 1 (1993), p. 77-150

<http://www.numdam.org/item?id=RO_1993__27_1_77_0>

Recherche opérationnelle/Operations Research
(vol. 27, n° 1, 1993, p. 77 à 150)

LES PROBLÈMES D'ORDONNANCEMENT (*)

par GOTHA ([1])

Résumé. — *Bien que les nouvelles méthodes de conception des systèmes de production aient tendance à diminuer la taille de certains problèmes d'ordonnancement en divisant les systèmes en cellules flexibles élémentaires, la diversité, la complexité et l'importance dans le monde industriel des problèmes d'ordonnancement demeurent très grands.*

La littérature scientifique est très riche en articles dans ce domaine, avec des points de concentration très importants sur des problèmes ou des méthodes particuliers. Nous présentons ici un état de l'art réalisé par onze chercheurs français. Il s'intéresse, dans une présentation qui se veut pédagogique, à l'ensemble du domaine étudié dans la littérature, en cherchant à faire profiter le lecteur au maximum des expériences propres de chacun des rédacteurs, ce qui explique les développements plus importants de certaines parties alors que d'autres aspects ne sont qu'évoqués avec références à d'autres états de l'art complémentaires.

Mots clés : Ordonnancement, algorithmes exacts, algorithmes approchés.

Abstract. — *Although recent methods used for designing production systems tend to reduce the size of some scheduling problems by dividing production systems into elementary flexible cells, the diversity, the complexity and the existence in the industrial world of scheduling problems remain very important.*

The scientific literature is filled with papers on the subject with some very concentrated points upon particular problems or methods. This paper is a state of the art written by eleven French researchers. It tries to present the whole domain in a pedagogic form and to make the reader take advantage of the peculiar experience of each writer. This explains the development more important of some parts while other aspects are only recalled with references to complementary specific surveys or papers.

Keywords: Scheduling, exact algorithms, approximation algorithms.

PLAN

(*) Reçu janvier 1991.

([1]) J. Carlier, P. Chrétienne, J. Erschler, C. Hanen, P. Lopez, A. Munier, E. Pinson, M.-C. Portmann, C. Prins, C. Proust, P. Villon.

Recherche opérationnelle/Operations Research, 0399-0559/93/01 77 74/$ 9.40

INTRODUCTION

Le GOThA est un groupe de recherche informel qui a pour but de faire se rencontrer des chercheurs d'horizons divers travaillant sur les problèmes d'ordonnancement. GOThA est un sigle pour Groupe d'Ordonnancement Théorique et Appliqué. L'ensemble des membres du groupe a non seulement mené des recherches théoriques mais aussi traité de nombreux cas réels.

La somme de nos expériences nous permet de tirer quelques conclusions. Les problèmes réels sont *mal résolus* et paraissent *a priori très complexes*;

ils peuvent cependant souvent être résolus de façon très satisfaisante. Ces problèmes sont *distincts les uns des autres* et ne peuvent pas être traités efficacement à l'aide d'un outil standard. Les décideurs ignorent souvent *l'origine des difficultés* de leur résolution. Une partie au moins de la recherche théorique est proche des cas réels. Les outils théoriques permettent à l'heure actuelle de résoudre certains problèmes de grande taille (par exemple le job-shop). Les *bonnes heuristiques*, suffisantes le plus souvent, sont des sous-produits d'*études théoriques fines*.

Ces conclusions expliquent notre démarche dans cet article. Notre but n'est pas d'être exhaustif mais de sensibiliser le lecteur à des problèmes dont l'importance échappe de moins en moins aux informaticiens mais encore trop souvent aux industriels. Nous avons ici quatre motivations principales. La première est de présenter les outils de base que tout praticien du domaine doit connaître (méthode potentiels-tâches, réseaux de Petri temporisés, méthodes de circuit critique, algorithmes de Johnson, de Jackson, de Smith et de Mac Naughton). Notre deuxième motivation est d'initier le lecteur au vocabulaire qui permet d'aborder la littérature foisonnante du domaine et à la reconnaissance des principaux problèmes étudiés (problèmes d'ateliers : flow-shop, job-shop et open-shop, problèmes à contraintes cumulatives). La troisième motivation est de décrire les différentes approches pour aborder ces problèmes (décomposition, améliorations successives, énumération implicite fondée sur des ensembles dominants). La quatrième motivation est de présenter des problèmes d'ordonnancement nouveaux (cycliques et distribués) dont l'étude récente est particulièrement importante pour les problèmes issus de l'informatique.

La première partie est une présentation générale des problèmes d'ordonnancement, de leur modélisation et des approches générales permettant de construire des solutions. La deuxième partie décrit les outils fondamentaux : la *méthode des potentiels,* bien connue, pour les problèmes sans contraintes de ressource, l'*algorithme du décalage* pour les problèmes à ressources consommables et la *méthode du circuit critique* pour le problème central répétitif. La troisième partie traite des problèmes d'ordonnancement à une et m machines (identiques) dont la résolution est à la base de problèmes plus généraux. Nous mettons surtout l'accent sur quelques résultats récents concernant le cas d'une seule machine, et nous présentons les derniers travaux sur les *problèmes d'ordonnancement distribués* pour lesquels la prise en compte de temps de communication entre machines est de première importance. La quatrième partie porte sur les problèmes d'atelier : flow-shop, job-shop, open-shop. Les concepts fondamentaux sur lesquels sont

fondés les algorithmes de résolution sont analysés et la méthode la plus efficace de résolution du problème de job-shop est décrite. Nous présentons également l'analyse sous contraintes qui consiste à caractériser les solutions admissibles d'un problème d'ordonnancement sans chercher à optimiser un critère particulier. La cinquième et dernière partie traite les *problèmes d'ordonnancement répétitifs*. On y montre que les réseaux de Petri temporisés constituent un bon outil de modélisation et l'on y présente quelques résultats récents pour des problèmes à une ressource et à plusieurs ressources.

1. PRÉSENTATION DES PROBLÈMES D'ORDONNANCEMENT

Le but de cette première partie est de présenter les problèmes d'ordonnancement. Nous rappelons d'abord les différents paramètres d'un tel problème. Puis nous expliquons comment les réseaux de Petri temporisés permettent de modéliser avec un seul formalisme la plupart de ces problèmes : il s'agit donc d'un excellent outil de simulation. Nous présentons ensuite les principes généraux des méthodes de résolution approchée. Ils seront illustrés dans le corps de l'article pour différents problèmes spécifiques.

1.1. Définitions. Typologie et modélisations

Ordonnancer un ensemble de *tâches,* c'est programmer leur exécution en leur allouant les *ressources* requises et en fixant leurs dates de début. La théorie de l'ordonnancement traite de modèles mathématiques mais analyse également des situations réelles fort complexes; aussi le développement de méthodes utiles ne peut-il être que le fruit de contacts entre la théorie et la pratique. Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie : l'informatique (les tâches sont les programmes; les ressources sont les processeurs, la mémoire, ...), la construction (suivi de projet), l'industrie (activités des ateliers en gestion de production et problèmes de logistique), l'administration (emplois du temps) [CARL 82 *b*].

Dans un problème d'ordonnancement interviennent deux notions fondamentales : les tâches et les ressources. Une ressource est un moyen, technique ou humain, dont la disponibilité limitée ou non est connue *a priori*. Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource.

La résolution d'un problème d'ordonnancement doit concilier deux objectifs. L'aspect *statique* consiste à générer un plan de réalisation des travaux sur la base de données prévisionnelles. L'aspect *dynamique* consiste à prendre des décisions en temps réel compte tenu de l'état des ressources et de l'avancement dans le temps des différentes tâches.

Les données d'un problème d'ordonnancement sont les tâches et leurs caractéristiques, les contraintes potentielles, les ressources, et la fonction économique. On note en général $I=\{1, 2, \ldots, n\}$ l'ensemble des tâches et p_i la durée de la tâche i si cette durée ne dépend pas des ressources qui lui sont allouées. Souvent une tâche i ne peut commencer son exécution avant une *date de disponibilité* notée r_i et doit être achevée avant une *date échue* d_i. Parfois, on a une durée de latence q_i qui minore la durée entre la fin de la tâche i et la fin de l'ordonnancement. Les dates réelles de début et de fin d'exécution de la tâche i sont notées respectivement t_i et C_i. Les tâches sont souvent liées entre elles par des *relations d'antériorité*. Si ce n'est pas le cas on dit qu'elles sont *indépendantes*. La contrainte d'antériorité la plus générale entre deux tâches i et j, appelée *contrainte potentielle,* s'écrit sous la forme $t_j - t_i \geqq a_{ij}$; elle permet d'exprimer la succession simple ($a_{ij}=p_i$) et de nombreuses variantes.

Dans certains cas, une tâche, dite *morcelable* ou encore *préemptive*, peut être exécutée par morceaux. Des techniques d'optimisation issues des mathématiques du continu (par exemple la programmation linéaire) sont alors souvent utiles pour résoudre ce type de problèmes.

On distingue deux types de ressources pouvant être requises par les tâches, les ressources *renouvelables* et les ressources *consommables* [SLOW 82]. Une ressource est renouvelable si après avoir été allouée à une tâche, elle redevient disponible pour les autres. Les ressources renouvelables usuelles sont les machines, les processeurs, les fichiers, le personnel... Au contraire, une ressource est consommable si, après avoir été allouée à une tâche, elle n'est plus disponible pour les tâches restant à exécuter. C'est le cas pour l'argent, les matières premières... Une ressource peut n'être disponible qu'à certaines périodes, qu'elle soit consommable ou non. On connaît alors *a priori* sa courbe de disponibilité.

Certaines contraintes liées à la limitation des ressources nécessitent, pour leur formulation, l'introduction d'un choix sur un ensemble d'inégalités de potentiels. Ce choix représente la résolution d'un conflit pour l'utilisation des ressources renouvelables. L'ensemble des conflits peut être modélisé par l'intermédiaire d'ensembles minimaux de tâches non réalisables simultanément compte tenu des ressources disponibles. Ces ensembles sont appelés *ensembles critiques* de tâches [NABE 73]. Pour résoudre un tel conflit, il suffit d'ordonner deux tâches quelconques de l'ensemble critique correspondant.

Un cas particulier important est celui où le cardinal d'un ensemble critique est égal à deux : on parle alors de contraintes disjonctives (la paire

[i, j] constituant l'ensemble critique est appelée paire de disjonctions). Pour résoudre le conflit relatif à cet ensemble critique, i et j doivent être ordonnées dans un sens ou dans l'autre. Ainsi lorsqu'on ne s'intéresse qu'à la résolution de conflits relatifs à des ensembles critiques d'ordre deux, on parle de problème disjonctif (utilisation de robots, de machines-outils...). Dans le cas général, c'est-à-dire en présence d'ensembles critiques d'ordre supérieur à deux et lorsqu'on s'intéresse à la résolution de tous les conflits, on parlera de problème cumulatif. Par exemple si on dispose de trois maçons, il ne sera pas possible d'exécuter simultanément trois tâches nécessitant respectivement deux, un et un maçon.

Dans un problème d'ordonnancement *cyclique,* l'ensemble I est constitué de *tâches génériques.* La tâche générique i donne naissance à une infinité d'*instances.* Il faut alors ordonnancer toutes les instances de l'ensemble des tâches génériques. Ce type de problème se rencontre pour des productions en série dans l'industrie mais aussi dans le cas de calculs répétitifs en informatique, comme l'exécution d'une boucle vectorielle sur une architecture pipeline.

Les facteurs les plus importants dans l'évaluation d'une solution d'un problème d'ordonnancement sont l'utilisation efficace des ressources, le délai global, la minimisation des encours et le respect des dates échues. Les variables intervenant le plus souvent dans l'expression de la fonction économique sont : la date C_i de fin d'exécution de la tâche i, le retard T_i=max (0, C_i–d_i) de la tâche i, l'indicateur de retard U_i (U_i=0 si $C_i \leqq d_i$, U_i=1 sinon). Les critères usuels sont la durée totale $C_{\max}$=Max C_i, le plus grand retard $T_{\max}$=Max T_i, le retard moyen pondéré $\sum w_i T_i$, ou les stocks d'encours $\sum w_i C_i$. Pour un problème d'ordonnancement répétitif, le critère usuel est la maximisation du débit, c'est-à-dire du nombre moyen d'itérations par unité de temps. Tous ces critères sont dits *réguliers,* car ils sont des fonctions décroissantes des dates de fin d'exécution des tâches.

Le tableau I, qui propose une typologie des problèmes d'ordonnancement, en montre l'extrême diversité.

Pour modéliser un problème d'ordonnancement, on peut utiliser des équations mathématiques (où les contraintes sont reliées par des « et », mais aussi par des « ou exclusif » lorsqu'il y a des contraintes de ressources) ou des graphes. Jusqu'à l'introduction récente des *réseaux de Petri temporisés,* on ne pouvait pas modéliser avec un seul formalisme graphique les contraintes potentielles et les contraintes de ressources. Les réseaux de Petri temporisés

TABLEAU I

Caractéristiques	Choix possibles (non exclusifs)				
Quantité de moyens (nature)	illimitée		limitée (renouvelables)		limitée (consommables)
Organisation des moyens	une machine	*m* machines en parallèle	machines en série « flow-shop »	« job-shop »	gammes non linéaires
Contraintes de fabrication	préemption autorisée	chevauchement		temps de transit	encours limités
Types de fabrication	unitaire	atelier (petite série)		masse (grande série)	continue
Répétition des travaux	cyclique ou répétitif			à la commande ou non répétitif	
Hypothèses sur la demande	statique (connue *a priori*)			dynamique (arrivée continue en temps réel)	
Critères principaux d'appréciation des solutions .	minimisation du cycle de fabrication		minimisation des encours		minimisation des retards

sont un excellent outil de simulation et permettent d'obtenir des résultats analytiques dans le cas des ordonnancements cycliques.

Rappelons que les réseaux de Petri ordinaires sont utilisés pour modéliser le comportement dynamique de systèmes discrets. Ils permettent d'exprimer sur un même graphe les événements générateurs des changements d'état, les conditions nécessaires à ces changements d'état et les règles de calcul d'un nouvel état si l'événement correspondant se produit. Ils sont composés de deux types d'objets : les *places* et les *transitions*. Les places sont associées aux variables d'état du système, les transitions représentent les événements susceptibles de modifier l'état du système.

Lorsqu'un réseau de Petri modélise un problème d'ordonnancement, les places sont associées aux différents types de ressources et aux contraintes de succession; les transitions sont associées aux tâches; les événements (franchissement des transitions) correspondent à l'exécution des tâches.

L'exécution d'une tâche n'est possible que si elle dispose des ressources nécessaires et si les tâches qui la précèdent sont terminées; ces préconditions à l'exécution d'une tâche se traduiront par l'existence d'arcs valués par des entiers liant certaines places à la transition considérée. Lorsqu'une tâche a terminé son exécution, certaines tâches peuvent commencer leur exécution, certaines ressources peuvent être restituées, d'autres ont pu être consommées, d'autres encore ont pu être produites. Ces post-conditions de terminaison d'une tâche se traduiront à nouveau par des arcs valués liant la transition considérée à un sous-ensemble de places.

Un réseau de Petri ordinaire ne peut traduire la durée d'une tâche; il ne permet pas de représenter la mobilisation d'une ressource durant l'exécution d'une tâche car l'état de cette ressource n'est pas le même avant, pendant et après l'exécution de cette tâche.

Le modèle de base doit donc être enrichi en introduisant le temps pour tenir compte des durées d'exécution des tâches [CHRE 83]. Le franchissement d'une transition est alors composé de deux actions instantanées décalées d'une période de temps égale à la durée de la tâche; le début de franchissement réalise l'allocation des ressources, la fin de franchissement opère la restitution (éventuelle) des ressources requises par la tâche.

Nous présentons à titre d'exemples un problème d'ordonnancement modélisé par un réseau de Petri temporisé [CARL 84].

Exemple : Un ensemble de n tâches soumises à des contraintes de précédence doivent être exécutées par une machine M. Une tâche i peut soit consommer soit rapporter une somme d'argent c_i. Il s'agit alors de trouver

un séquencement des tâches sur le processeur qui puisse être financé à partir d'un capital initial C. Un extrait du réseau de Petri temporisé associé est rapporté sur la figure 1.

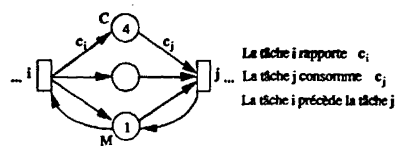


Figure 1. — Un extrait du réseau de Petri temporisé associé à un problème d'ordonnancement.

1.2. Approches de résolution

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, procédures par séparation et évaluation, théorie des graphes...). Ces méthodes garantissent en général l'optimalité de la solution fournie. Mais les algorithmes dont la complexité n'est pas polynomiale ne peuvent pas être utilisés pour des problèmes de grande taille, d'où la nécessité de construire des méthodes de résolution approchée, efficaces pour ces problèmes souvent NP-difficiles [GARE 79]. L'objectif de ce paragraphe est de présenter une typologie générale de construction de méthodes de résolution qui regroupe à la fois les méthodes exactes et les méthodes approchées.

L'analyse d'un problème particulier permet d'obtenir des propriétés sur la structure des solutions optimales, propriétés à partir desquelles on peut ou bien caractériser les solutions optimales ou bien réduire l'espace des solutions à explorer (notion de sous-ensemble dominant qui contient au moins une solution optimale). Ces résultats permettent soit de prouver l'optimalité des solutions obtenues par un algorithme soit de rendre plus efficace les méthodes présentées ci-dessous.

Nous distinguons dans la suite cinq types de méthodes : les méthodes par construction progressive, par voisinage, par décomposition, par relaxation et enfin celles liées à l'intelligence artificielle.

1.2.1. *Méthodes par construction progressive*

Les méthodes par construction progressive sont des méthodes itératives où, à chaque itération, on complète une solution partielle [NAWA 83].

En ordonnancement, on peut construire de nombreuses méthodes itératives. Par exemple celles où chaque itération place successivement toutes les opérations d'un travail, celles où au contraire chaque itération place une opération au plus sur une des machines. Ces méthodes peuvent suivre le déroulement du temps ou non (on suit le déroulement du temps si la suite des instants de début des opérations que l'on place est non décroissante). Elles peuvent également construire des ordonnancements quelconques, semi-actifs, actifs ou sans-délai [BAKE 74], [FREN 82]. Si le choix de l'opération à ajouter à l'ordonnancement est défini par l'application d'un théorème de dominance, la méthode peut fournir une solution optimale; c'est par exemple le cas pour le problème de base à une machine et la minimisation de la somme des temps de présence ou du plus grand retard (règles de Smith [SMIT 56] ou de Jackson [JACK 55]). Dans le cas général, le choix de l'opération fixé par des règles de priorité (faisant ou non intervenir le hasard) ne conduit en général qu'à des solutions approchées. C'est en particulier le cas pour la résolution des problèmes de type cumulatif par des méthodes dites « sérielles », encore appelées algorithmes de listes, qui fabriquent des solutions à partir d'une liste ordonnée des tâches. Lorsque qu'il existe des dates échues impératives, la méthode peut conduire à des solutions non réalisables; dans ce cas, on effectue en général un « backtracking » par une remise en cause récurrente du dernier choix. On développe ainsi une arborescence de recherche jusqu'à la détermination d'une solution réalisable.

1.2.2. *Méthodes par voisinage*

Contrairement aux méthodes par construction qui travaillent sur des solutions partielles, les méthodes par voisinage travaillent sur des solutions complètes. Chaque itération de la méthode par voisinage ayant pour objectif (pas toujours atteint) de passer d'une solution complète à une autre solution complète meilleure relativement au critère considéré [DANN 77].

Pour pouvoir construire une méthode par voisinage, il faut disposer d'une solution initiale complète, d'une fonction $f(x)$ qui fournit la valeur du critère pour une solution x et d'une fonction de voisinage $v(x)$ qui choisit la solution voisine de x. Pour définir $v(x)$, on peut par exemple inverser deux opérations dans un problème de séquencement, ou dans un problème à plusieurs machines affecter une opération à une autre machine, éventuellement par un tirage au hasard de l'opération et/ou de la machine.

Selon les propriétés que doit vérifier le voisin choisi, on distingue plusieurs méthodes par voisinage :

– choix du meilleur voisin meilleur que x ou arrêt s'il n'en existe pas, il s'agit alors d'une *méthode de plus forte pente*;

– choix aléatoire d'un voisin meilleur que x et arrêt s'il n'en existe pas, il s'agit alors d'une *méthode de descente*;

– choix d'un voisin quelconque; si ce voisin est meilleur que x, il est définitivement accepté; sinon il est accepté avec la probabilité exp (–perte/T) où T est une température que l'on fait décroître par paliers. Il est possible d'implémenter comme test d'arrêt l'occurence de Q itérations successives non améliorantes par rapport à la meilleure solution connue. Si l'on réalise plusieurs essais à partir de points initiaux choisis aléatoirement, il s'agit d'une méthode dite de *recuit simulé* [KIRK 82] par analogie avec la thermodynamique;

– méthode de descente ou de plus forte pente mais on accepte de « remonter » lorsqu'un optimum local est atteint tout en évitant de repasser par des solutions déjà visitées; il s'agit alors d'une *méthode Tabou* dont il existe de multiples variantes [GLOV 87], [HERZ 90].

1.2.3. *Méthodes par décomposition*

Il existe de nombreuses façons de construire des méthodes de résolution par décomposition.

– la décomposition « hiérarchique » ([AXSA 84], [BITR 82], [ERSC 85]) consiste à décomposer les problèmes en plusieurs niveaux, par exemple un niveau supérieur où on travaille sur des données agrégées et où on décide d'un certain nombre de paramètres globaux et un niveau inférieur détaillé où les décisions prises sur ces paramètres globaux deviennent des contraintes pour le niveau inférieur, ce qui limite le domaine des solutions à explorer,

– la décomposition « structurelle » ([ROY 70]) utilise la modélisation du problème et ses grandes familles d'inconnues, considère tout d'abord que les moyens sont illimités et résout le problème temporel. Les dates étant fixées, on cherche la meilleure affectation possible des moyens, puis on revient au problème temporel en ajoutant de nouvelles contraintes liées à l'utilisation des moyens...

– la décomposition de « l'ensemble des solutions du problème » conduit, si l'on veut obtenir la solution optimale, à des procédures par séparation et évaluation ([LENS 77]). Toutefois, ces méthodes exactes peuvent être transformées en heuristiques de différentes façons, par exemple, comme dans

le cas du recuit simulé, en les arrêtant lorsque pendant Q itérations on n'a pas strictement amélioré la meilleure solution trouvée ou encore, lors des séparations, en n'essayant pas toutes les valeurs possibles pour l'inconnue sur laquelle on effectue la séparation, mais seulement un échantillon de valeurs possibles...

– la décomposition « temporelle » ([PORT 88], [MEGU 88]), dans le cas des ordonnancements dynamiques où les travaux arrivent à des dates différentes et dispersées dans le temps résout lors de la première itération un problème réduit en ignorant les travaux qui arrivent au delà d'une date choisie, puis retient définitivement cet ordonnancement partiel jusqu'à une autre date déterminée. A l'itération suivante, on décalera ces deux dates de manière à construire la portion suivante de l'ordonnancement...

– la décomposition « spatiale » ([PORT 88], [CHU 92 *a*]) décompose l'atelier en sous-ateliers avec le moins possible de déplacements de produits entre les sous-ateliers grâce à des outils de technologie de groupe et construit une méthode itérative dans laquelle chaque itération ordonnance un sous-atelier (*voir* 4.1.2.2).

Il est bien sûr possible de combiner de différentes façons toutes ces méthodes par décomposition (*voir* par exemple [PORT 88] et [CHRY 91] pour des décompositions spatiales et temporelles).

Pour mettre au point certaines méthodes de décomposition hiérarchique ou certaines méthodes de décomposition spatiale, on est amené à utiliser des méthodes dites de « technologies de groupe ». La technologie de groupe consiste à effectuer des regroupements de machines et/ou des regroupements de produits de telle sorte que les familles ainsi constituées minimisent un critère donné. Les regroupements ainsi calculés peuvent avoir des conséquences physiques sur la conception des systèmes de production (*voir* par exemple [KUSI 85 *a*, 85 *b*] pour leur application dans le domaine des ateliers flexibles) ou servir simplement d'outils internes pour améliorer des méthodes d'ordonnancement ([PORT 88] et 4.1.2.2).

1.2.4. *Méthodes par modification des contraintes*

Il s'agit ici de méthodes où l'on change le modèle des problèmes que l'on a à résoudre. Ce peut être, par exemple, de transformer un flow-shop normal en un flow-shop de permutation de manière à avoir moins de solutions à explorer, mais on trouve surtout ici toutes les méthodes dites de « relaxation » : relaxation de contraintes d'intégrité, relaxation lagrangienne ([FISH 73], [VELD 91]), relaxation « surrogate » ([GLOV 75])...

Le fait de relâcher une contrainte conduit à des solutions qui ne sont plus réalisables, mais qui fournissent des évaluations par défaut qui peuvent être intégrées dans des méthodes par séparation et évaluation ou dans des méthodes itératives approchées utilisant le dual lagrangien.

1.2.5. *Méthodes liées à l'intelligence artificielle*

Il s'agit de méthodes qui utilisent des techniques de représentation des connaissances et de résolution de problèmes issues de l'intelligence artificielle. De nombreux travaux ont abordé les problèmes d'ordonnancement à l'aide de ces outils (pour une synthèse, on peut consulter [KUSI 88]). Citons par exemple la propagation par contrainte qui alliée à des règles de production permet de résoudre des problèmes d'ordonnancement (§ V), les systèmes experts (OPAL, SOJA) et les systèmes à base de connaissances utilisant de l'apprentissage, comme par exemple la mémoire artificielle ([CHU 90]). Parmi les méthodes qui cherchent à s'inspirer de phénomènes concrets, on peut également citer les algorithmes génétiques ([GOLD 89]). Comme le recuit simulé, ce sont des algorithmes aléatoires, mais au lieu de chercher à améliorer progressivement une seule solution, ils font évoluer une population de solutions qui s'améliore globalement par des techniques de reproduction, de sélection et de mutation. Ces méthodes commencent à être utilisées en ordonnancement ([BIEG 90], [FALK 91]).

Il est bien sûr possible de combiner ces techniques et de les croiser avec les autres méthodes présentées.

2. PROBLÈMES CENTRAUX

On qualifie de « central » un problème d'ordonnancement sans contraintes de ressources ou pour lequel les ressources sont en nombre suffisant quelque soit l'ordonnancement. Le but de cette deuxième partie est de présenter les méthodes de résolution de ces problèmes centraux. Elles constituent en effet les outils de base que l'on doit parfaitement connaître avant de chercher à résoudre un problème d'ordonnancement. La méthode potentiels-tâches est évidemment fondamentale, nous la rappelons brièvement. Nous décrivons également deux de ses extensions : la méthode du décalage qui permet de gérer des ressources consommables et la méthode du circuit critique qui donne une solution optimale du problème central répétitif.

2.1. Le problème central classique

Dans le problème central de l'ordonnancement, il s'agit d'ordonnancer, en une durée minimale, des tâches soumises à des contraintes de succession et de localisation temporelle. Ces contraintes sont résumées par un graphe *potentiels-tâches* $G=(X, U)$ où X est l'ensemble des tâches complété par une tâche fictive début notée 0 et une tâche fictive fin notée *, et U est formé des arcs associés aux contraintes potentielles, l'arc (i, j) valué par a_{ij} correspondant à la contrainte potentielle $t_j - t_i \geqq a_{ij}$. Un ordonnancement est un ensemble $T=\{t_i / i \in X\}$ de dates de début des tâches telles que la date de début de la tâche fictive 0 soit zéro et que, pour tout arc (i, j), $t_j - t_i \geqq a_{ij}$. Un tel ordonnancement existe si et seulement si G ne contient pas de circuit de valeur strictement positive. Dans ce cas, en notant $l(i, j)$ la valeur maximale d'un chemin de G de i à j, $E=\{e_i = l(0, i) / i \in X\}$ et $F=\{f_i = l(0, *) - l(i, *) / i \in X\}$ sont des ordonnancements optimaux dont la durée minimale $l(0, *)$ est la valeur maximale des chemins de 0 à *, appelés *chemins critiques*. L'ensemble E est l'ordonnancement *au plus tôt* et l'ensemble F est l'ordonnancement *au plus tard*. La figure 2 rapporte les ordonnancements au plus tôt et au plus tard d'un tel problème

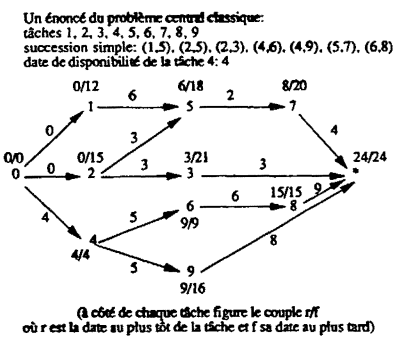


Figure 2. — Le graphe potentiel associé.

La notion d'ordonnancement au plus tard se généralise aux ordonnancements de durée $\Delta = l(0, *) + \delta$ ($\delta \geqq 0$). Si le graphe G ne contient pas d'arc $(i, 0)$ ($i \in X$), qui correspondrait à une date échue de la tâche i, l'ordonnancement *au plus tard* $F(\delta)$ est donné par : $f_i(\delta) = f_i + \delta$ pour tout i dans $X - \{0\}$ et $f_0(\delta) = 0$. Il est donc obtenu en décalant les tâches de δ.

2.2. Le problème central à ressources consommables

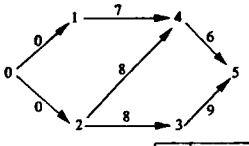
Des tâches liées par des contraintes potentielles doivent être ordonnancées en une durée minimale. Chaque tâche i consomme à son début a_i unités

d'une ressource consommable. Initialement, à la date $u_1=0$, b_1 unités de cette ressource sont disponibles. Des quantités additionnelles $b_2, \ldots, b_q$ deviennent disponibles aux dates $u_2, \ldots, u_q$. Ce modèle représente, par exemple, le financement d'une réalisation.

Un ordonnancement doit donc satisfaire, outre les contraintes potentielles, la condition associée à la consommation de la ressource que l'on peut résumer ainsi : la courbe de la demande doit être située en dessous de celle de l'offre. Cette dernière contrainte impose de consommer la ressource le plus tard possible. On peut donc limiter la recherche d'un ordonnancement optimal aux ordonnancements aux plus tard F (δ). En l'absence de dates échues, on a f_i (δ)=f_i+δ. Il suffit alors d'appliquer l'algorithme suivant :

début
Calculer l'ordonnancement F.
Décaler la courbe de la demande D (t) de F de la valeur minimale δ^* pour qu'elle soit sous la courbe de l'offre S (t).
Ecrire « F (δ^*) est un ordonnancement optimal ».
fin

Voici un exemple :



i	1	2	3	4
a_i	7	7	7	7

k	1	2	3
u_k	0	5	10
b_k	10	10	10

Figure 3. — Un énoncé du problème central à ressources consommables.

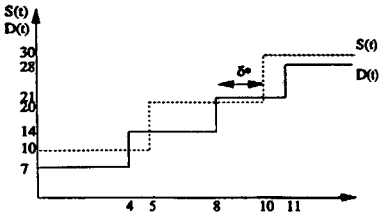


Figure 4. — Courbes de la demande et de l'offre.

La figure 3 rapporte les contraintes potentielles liant les quatre tâches 1, 2, 3 et 4 ainsi que l'échéancier. L'algorithme du décalage calcule l'ordonnancement au plus tard F : $f_0=0, f_1=4, f_2=0, f_3=8, f_4=11, f_5=17$. Puis, il détermine

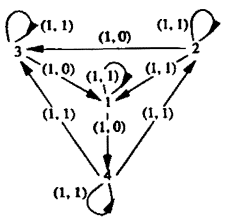
la quantité minimale dont il faut décaler la courbe de la demande pour qu'elle soit sous la courbe de l'offre (*fig.* 4). Ici $\delta^*=2$. Un ordonnancement optimal est donc : $t_0=0$, $t_1=6$, $t_2=2$, $t_3=10$, $t_4=13$, $t_5=19$. La méthode proposée se généralise pour prendre en compte des dates échues ou d'autres formes de consommation de la ressource [CARL 82 *a*, 89 *c*].

2.3. Le problème central cyclique

On se donne ici un ensemble fini $I=\{1, 2, \ldots, n\}$ de *tâches génériques*. La tâche générique i induit une infinité d'*instances* (i, k), $k \in N$, toutes de durée p_i. On appelle k-ième *itération* l'ensemble $\{(i, k)/i \in I\}$. Un ensemble fini de *contraintes de précédence génériques* lie les tâches génériques. Une telle contrainte est spécifiée par un triplet (i, j, h) où i et j sont deux tâches génériques et h est un entier naturel appelé *hauteur*. Elle signifie que, pour tout entier n, l'instance $(j, h+n)$ ne peut commencer son exécution qu'après la fin d'exécution de l'instance (i, n). On suppose de plus que les intervalles d'exécution des instances d'une même tâche générique soient disjoints, c'est-à-dire que, pour toute tâche générique i, la contrainte de précédence générique $(i, i, 1)$ existe. Remarquons que si la hauteur d'une contrainte générique est nulle, elle induit des contraintes de succession entre instances d'une même itération (contraintes dites internes), sinon elle induit des contraintes de succession entre instances d'itérations distinctes (contraintes dites externes). Le critère d'optimisation est la *maximisation du débit,* c'est-à-dire de la fréquence d'exécution des itérations.

La résolution de ce problème [CHRE 85] est fondée sur l'étude des chemins de valeur maximale d'un graphe (en fait un multigraphe) appelé *graphe de précédence généralisé* dont les arcs sont étiquetés par une *valeur* et une *hauteur*. Nous présentons ci-dessous les résultats concernant le cas particulier d'un graphe fortement connexe. Les sommets du graphe sont associés aux tâches génériques. Chaque contrainte de précédence générique (i, j, h) donne lieu à un arc d'origine i, d'extrémité j, de valeur p_i et de hauteur h. La figure 5 rapporte le graphe bivalué associé à un énoncé du problème central répétitif.

Le résultat fondamental concerne le comportement asymptotique de la suite $a_n(i)$ des valeurs maximales des chemins de hauteur n et d'extrémité i. Cette suite est *K-périodique*, c'est-à-dire qu'il existe un entier K appelé *facteur de périodicité* et un réel r appelé *période* tels qu'à partir d'un certain rang N, appelé *durée du régime transitoire*, on a : $a_{n+K}(i)=a_n(i)+r$. Si l'on appelle *circuit critique* d'un graphe bivalué un circuit dont le rapport valeur sur



Données du problème central répétitif:
tâches génériques: 1,2,3,4;
contraintes de précédence génériques:
(i,i,1) pour i=1,2,3,4
(1,4,0),(2,3,0),(2,1,1),(3,1,0),(4,3,1),(4,2,1)
durées unitaires
Résultats:
Le circuit critique est le circuit
(2,3,1,4,2), de valeur 4 et de hauteur 1
la période de l'ordonnancement
au plus tôt est donc r=4,
le facteur de périodicité est K=1.

Figure 5. — Graphe bivalué associé au problème central répétitif.

hauteur est maximal, la période r est égale à la valeur du circuit critique. On montre que le facteur de périodicité est le plus petit commun multiple des hauteurs de tous les circuits critiques.

La résolution du problème central répétitif consiste alors en l'étude de son graphe bivalué associé. En effet, la date au plus tôt de l'instance (i, p) étant égale à a_p (i), l'existence, le calcul et les propriétés de l'ordonnancement au plus tôt (en particulier sa K-périodicité) sont induites par les résultats généraux précédents. La figure 5 rapporte les valeurs de r et K pour l'ordonnancement au plus tôt d'un énoncé du problème central répétitif [CHRE 85].

Une extension du problème central répétitif considérant le cas où toutes les instances de tâches sont soumises à des dates échues impératives a été étudiée dans [CHRE 91]. On y donne les conditions d'existence d'un ordonnancement au plus tard et l'on calcule cet ordonnancement dans le cas particulier important où les dates échues impératives sont les dates au plus tôt de fin des itérations.

3. PROBLÈMES À RESSOURCE UNIQUE

Le but de cette partie est de présenter quelques résultats de base sur les problèmes à une et m machines. Ces problèmes peuvent sembler très spécifiques. Pourtant leur résolution est à la base de celle de problèmes plus généraux, en particulier quand il apparaît qu'une ressource est dominante. Nous nous limitons essentiellement à deux critères : la durée totale et la somme des retards. La durée totale est le critère le plus étudié dans la mesure où d'autres critères s'y ramènent. Nous rappelons la règle de Jackson et décrivons les propriétés du problème à une machine sur lesquelles sera fondée la résolution du job-shop présentée dans la quatrième partie. La minimisation de la somme des retards nous amène à présenter les méthodes de décomposition et d'optimisation locale qui permettent de diminuer le coût des calculs tout en fournissant des solutions approchées de bonne qualité.

3.1. Problèmes à une machine

3.1.1. *Résultats sur le problème de base*

Le problème de base consiste à déterminer sur la machine un séquencement optimal de n tâches disponibles à l'instant 0 [BAKE 74]. Pour minimiser les encours, c'est-à-dire $\sum w_i C_i$, il suffit de ranger les tâches dans l'ordre des p_i/w_i croissants (règle de Smith) si les tâches sont indépendantes. Si les tâches sont dépendantes, le problème est NP-difficile dans le cas général et polynomial pour une arborescence [COFF 76]. Pour minimiser le plus grand retard, il suffit de ranger les tâches dans l'ordre des d_i croissants (règle de Jackson) et, si les tâches sont dépendantes de les ranger dans l'ordre des d'_i croissants où d'_i=min $\{d_j / j$ descendant de $i\}$. L'algorithme polynomial de Hodgson permet de minimiser le nombre de tâches en retard, il consiste à placer les tâches, une par une, dans l'ordre des dates échues croissantes, en rejetant la tâche déjà placée de plus grande durée à la fin de l'ordonnancement tant qu'un retard apparaît dans l'ensemble des tâches non rejetées. Le problème est NP-difficile pour la somme pondérée des retards, y compris lorsque tous les poids sont égaux [DU 90].

On trouve dans la littérature des travaux présentant des méthodes exactes (arborescentes [POTT 87] ou de programmation dynamique [SRIN 71]) ou des méthodes approchées résolvant différents problèmes à une machine dérivés du problème de base en ajoutant des contraintes. Citons par exemple le cas où chaque changement de tâche sur la machine s'accompagne d'un temps de réglage qui dépend des deux tâches consécutives, déterminer un ordonnancement de durée minimale est équivalent à résoudre un problème de chemin hamiltonien de coût minimum (voisin du problème du voyageur de commerce). Le paragraphe suivant présente quelques résultats récents concernant la minimisation de la somme des retards dans le cas où les tâches sont soumises à des dates de disponibilité.

3.1.2. *Minimisation de la somme des retards*

En présence de dates de disponibilité, la minimisation de la somme des retards est un problème NP-difficile [RINN 76]. Dans le cas particulier où toutes les durées sont unitaires, la règle de Jackson fournit cependant une solution optimale [JACK 55]. Des algorithmes de résolution pseudo-polynomiaux existent lorsque toutes les dates de disponibilité sont égales (par exemple [SRIN 71]), mais pas dans le cas général.

Dans le cas où les dates de disponibilité sont distinctes, une condition suffisante d'optimalité locale [CHU 89, 92 *b*] induit une nouvelle règle de

séquencement notée PRTT (pour *Priority Rule for Total Tardiness*), qui généralise la règle de Smith [SMIT 56]. Cette règle est à la base de nouveaux algorithmes approchés et a permis de définir un nouveau sous-ensemble dominant d'ordonnancements actifs, appelés T-actifs. Nous résumons ces différents résultats.

Considérons deux tâches consécutives i et j dans un ordonnancement donné et déterminons sous quelle condition placer i avant j minimise la somme des retards de ces deux tâches, sachant que la machine est disponible à l'instant Δ (*fig.* 6). En notant $T_{i,j}(\Delta)$ [respectivement $T_{j,i}(\Delta)$] la somme des retards des tâches i et j et en plaçant, au plus tôt à partir de l'instant Δ, i avant j (resp. j avant i), on cherche à quelle condition l'on a $T_{i,j}(\Delta) \leqq T_{j,i}(\Delta)$. On démontre qu'il suffit d'avoir PRTT $(i, \Delta) \leqq$ PRTT (j, Δ) où PRTT est une fonction de priorité définie par la formule : PRTT $(i, \Delta) = \max\{\Delta, r_i\} + \max\{\max\{\Delta, r_i\} + p_i, d_i\}$.

On définit un ordonnancement T-actif en considérant les ordonnancements actifs tels que si i et j sont deux tâches consécutives avec j qui suit i, alors soit la tâche i est arrivée strictement avant j ($r_i < r_j$) soit i est plus prioritaire que j selon la règle PRTT appliquée à l'instant où la machine est disponible avant l'exécution de i.

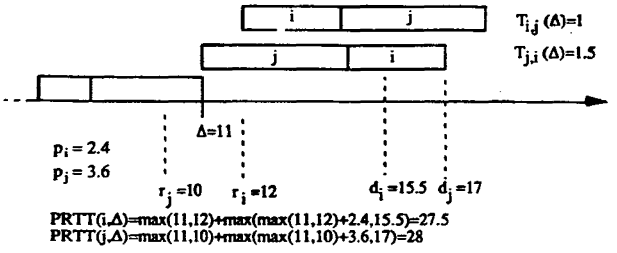


Figure 6. — Ordonnancement partiel de deux tâches.

Le sous-ensemble des ordonnancements T-actifs est dominant pour le critère « somme des retards ». En utilisant PRTT, non plus comme une condition suffisante d'optimalité locale de deux tâches, mais comme une règle de priorité locale entre les tâches, on peut construire de nombreux algorithmes approchés. On peut par exemple, utiliser PRTT comme première règle de choix dans un générateur d'ordonnancements actifs ou sans délais (la machine n'est jamais laissée inoccupée pour attendre une tâche plus prioritaire si une autre tâche est disponible). On peut également construire des algorithmes produisant des ordonnancements T-actifs, avec une phase d'insertion : en début d'itération, on place toujours une des tâches non

encore placées pour lesquelles PRTT est minimale, puis on insère des tâches devant de manière à combler le mieux possible avec les tâches les plus prioritaires l'intervalle de disponibilité existant (éventuellement) devant la tâche placée en début d'itération. Des expériences montrent qu'en moyenne, les algorithmes approchés utilisant PRTT sont meilleurs que les algorithmes approchés utilisant des notions de marge. La différence est importante lorsque les écarts-type des lois des dates d'arrivée sont grands.

3.1.3. *Minimisation de la durée totale*

On s'intéresse ici à l'ordonnancement d'un ensemble I de tâches indépendantes sur une machine unique dans le but de minimiser la durée totale. Chaque tâche est caractérisée par sa date de disponibilité r_i, sa durée p_i et sa durée de latence q_i.

Pour tout sous-ensemble K de tâches, $h(K) = \underset{i\in K}{\text{Min}}\, r_i + \sum_{i\in K} p_i + \underset{i\in K}{\text{Min}}\, q_i$ est une borne inférieure de la durée d'un ordonnancement. Il en résulte que Max $\{h(K), K \subseteq I\}$ est une évaluation par défaut de la durée minimale.

L'ordonnancement de Jackson est l'ordonnancement de liste associé à la règle de priorité MWR (Most Work Remaining, ou encore q_i maximal). Son principal défaut, comme tout algorithme de liste, est qu'il n'intègre pas la possibilité de différer l'ordonnancement d'une tâche de manière à favoriser le passage ultérieur d'une tâche plus prioritaire. Néanmoins, si f_0 est la durée de l'ordonnancement de Jackson, et si f^* est la durée minimale d'un ordonnancement, il existe un sous-ensemble $J \subset I$ tel que : soit $h(J)=f_0$ (l'ordonnancement de Jackson est alors optimal), soit $h(J)+p_c>f_0$ pour une tâche c de $I-J$ (J et c sont alors appelés respectivement ensemble et tâche critiques). De plus on a : $f^*-h(J) \leq p_c-1$ [CARL 82 *c*].

Baker et Su ont proposé une relaxation de la contrainte de non préemption pour le problème à une machine, obtenant ainsi une version préemptive de l'algorithme de Jackson : une tâche peut être interrompue si une autre tâche plus prioritaire devient disponible. L'intérêt de la solution préemptive réside principalement dans le fait que sa durée, égale à V=Max $\{h(K), K \subseteq I\}$, est minimale.

Carlier a proposé dans [CARL 82 *c*] une méthode arborescente pour le problème non préemptif basée sur ces résultats. Partant de la solution de Jackson relative au problème à une machine, il détermine l'ensemble et le couple critiques associés. Si cette solution n'est pas optimale, il crée, conformément au théorème ci-dessus, deux sous-problèmes, le premier dans lequel la tâche c est ordonnancée avant toutes les tâches de l'ensemble

critique J, le second dans lequel la tâche c passe après toutes ces tâches. Ces deux sous-ensembles de solutions sont alors évalués en calculant la solution de Jackson préemptive.

Nous allons maintenant introduire pour ce problème la notion d'arbitrage trivial. Cette notion est très utile pour résoudre des problèmes disjonctifs. Elle permet des énumérations implicites efficaces. Nous nous intéressons ici au positionnement d'une tâche c par rapport à un sous-ensemble J de tâches de $I-\{c\}$ et nous définissons les conditions (C1) (C2) (C3) suivantes :

(C1) $$r_c + p_c + \sum_{i \in J} p_j + \operatorname{Min}_{j \in J} q_j > f;$$

(C2) $$\operatorname{Min}_{j \in J} r_j + \sum_{i \in J} p_j + p_c + q_c > f;$$

(C3) $$\operatorname{Min}_{j \in J} r_j + \sum_{i \in J} p_j + \operatorname{Min}_{j \in J} q_j + p_c > f.$$

Si nous appelons solution tout ordonnancement de durée inférieure ou égale à f, nous avons les résultats suivants : si (C3) est vérifiée, alors dans toute solution, la tâche c est exécutée, soit avant soit après toutes les tâches de J. Si (C1) et (C3) sont vérifiées, alors dans toute solution, la tâche c est exécutée après toutes les tâches de J (*arbitrage trivial de type primal*); si (C2) et (C3) sont vérifiées, alors dans toute solution, la tâche c est exécutée avant toutes les tâches de J (*arbitrage trivial de type dual*).

On constate aisément que l'ensemble des résultats de type dual se déduit de ceux obtenus pour le type primal du fait du rôle symétrique joué par les dates de disponibilité et les durées de latence.

Si (C1) et (C3) sont vérifiées, alors dans toute solution, la tâche c ne peut commencer avant la date $b_{Jc}=\operatorname{Max}_{\{J \supset J'\}} \{\operatorname{Min}_{\{i \in J'\}} r_i + \sum_{\{i \in J'\}} p_i\}$. Si b_{Jc} est strictement plus grand que r_c, nous pouvons augmenter r_c en posant $r_c=b_{Jc}$.

Dans la recherche des arbitrages triviaux relatifs à une machine, il est possible de se limiter aux arbitrages dits efficaces, c'est-à-dire générant soit au moins un arc conjonctif dans le graphe disjonctif courant, soit une minoration de r_c. Ce problème peut alors se reformuler de la manière suivante : une tâche c étant fixée, rechercher un arbitrage trivial (c, J) (s'il existe) conduisant à la plus grande augmentation de r_c.

Nous avons montré dans [PINS 88] que la détermination de J peut se faire de manière implicite en recherchant seulement un sur-ensemble J d'un

ensemble J^* de tâches maximisant b_{Jc}. Un algorithme permet également la détermination de tous les arbitrages triviaux efficaces d'une clique. Il est généralisé au job-shop par la construction parallèle des ordonnancements préemptifs associés à chacune des machines, intégrant la détection des arbitrages triviaux.

3.2. Machines à exemplaires multiples

3.2.1. *Problèmes classiques à m machines*

On suppose ici que chaque tâche requiert une machine et que l'on dispose de m exemplaires identiques de cette machine. On considère dans toute cette section le critère de minimisation de la durée totale. Dans le cadre de l'informatique on peut ainsi modéliser les m processeurs d'une machine parallèle. Ces problèmes sont presque toujours NP-difficiles. Seuls les cas particuliers de durées unitaires et deux machines ou de durée unitaire et arbre de précédence donnent lieu à des algorithmes de listes optimaux.

Les études théoriques ont surtout porté sur la recherche d'algorithmes approchés avec garantie de performance. Dans le cas de tâches indépendantes, Graham a montré [GRAH 79] que la liste des tâches rangées par durées décroissantes impliquait une erreur relative majorée par 1/3-1/3 m. D'autres bornes, en général très difficiles à prouver mathématiquement, ont été obtenues pour d'autres listes, souvent en liaison avec des études sur le problème de bin-packing [COFF 76]. De nombreuses approches heuristiques ont été envisagées avec divers critères pour le problème non préemptif, dont celles de Parker *et al.*, [PARK 77], de Dogramaci et Surkis, [DOGR 79], ou de Sumichrast et Baker, [SUMI 87].

Dans le cas de tâches préemptives, les problèmes sont plus simples. L'algorithme classique de Mac Naughton concerne des tâches indépendantes et fournit un ordonnancement de durée minimale égale à B=Max $\{\text{Max } p_i, \sum p_i/m\}$ avec au plus m–1 préemptions pour une complexité en $O(n)$: il place les tâches progressivement sur chaque machine dans l'intervalle de temps $[0, B]$.

Si la tâche i doit être exécutée dans l'intervalle $[r_i, d_i]$, on résout polynomialement le problème en le modélisant par une recherche de flot maximal dans un réseau de transport biparti. La suite des valeurs distinctes prises par les d_i et les r_i découpe l'intervalle [Min r_i, Max d_i] en une suite d'intervalles consécutifs $I(k)$ de longueur $a(k)$. Un arc de capacité Min $\{p_i, a(k)\}$ lie la tâche i à chaque intervalle $I(k)$ contenu dans $[r_i, d_i]$. Le flux entrant dans la tâche i doit être égal à p_i et le flux sortant d'un intervalle $I(k)$ doit être inférieur ou égal à $m\,a(k)$.

Lorsque les tâches sont indépendantes et que leur durée dépend de la machine qui leur est allouée (p_{ik} est la durée de la tâche i sur la machine k), le problème est résolu en déterminant d'abord la solution optimale d'un programme linéaire qui fournit les durées globales de passage de chaque tâche sur les machines, puis en calculant le découpage en morceaux et l'ordonnancement par un algorithme itératif fondé sur un calcul de flot canalisé à chaque itération [SLOW 78].

3.2.2. *Problèmes d'ordonnancement distribué*

Dans certains problèmes d'ordonnancement, une contrainte de succession est associée à un *transfert* entre les machines qui exécutent les deux tâches. Dans les applications informatiques par exemple, il peut s'agir d'un transfert de données entre les deux tâches. La durée de ce transfert dépend des deux tâches mais aussi de la « distance » entre les deux machines. Ces problèmes sont bien sûr plus difficiles que les problèmes d'ordonnancement classiques à exemplaires multiples puisqu'ils les contiennent comme cas particuliers.

Les recherches théoriques ont d'abord porté sur le problème central où le nombre de processeurs n'est pas limité et les distances entre les machines sont égales. Si la tâche i est un prédécesseur immédiat de la tâche j, le temps du transfert de la tâche i à la tâche j est nul si ces deux tâches sont exécutées sur la même machine, sinon il est strictement positif et on le note c_{ij}. Nous nous limiterons dans la suite du paragraphe à ce problème et à certains de ses cas particuliers.

Une hypothèse importante est la possibilité ou non de *dupliquer* les tâches de manière à réaliser un plus grand nombre de transferts sur une même machine et de s'affranchir ainsi le plus possible des délais de communication. Dans les problèmes issus de la production, cette hypothèse n'est pas réaliste. Par contre dans les applications informatiques où les tâches représentent des programmes à exécuter sur un réseau de processeurs, elle peut être tout à fait pertinente dans la mesure où on ne surcharge pas trop les processeurs. La figure 7 montre sur un exemple le gain qui peut résulter de l'hypothèse de duplication.

Considérons d'abord le cas où la duplication est interdite. Le problème central est NP-difficile même pour des graphes de précédence dont la structure est assez simple [CHRE 90 *a*]. Par contre, dans le cas d'une arborescence et si les temps de communication sont inférieurs aux temps d'exécution des tâches, le problème est résolu par un algorithme de complexité $O(n^2)$ fondé sur la programmation dynamique [CHRE 89]. Pour un graphe quelconque et des

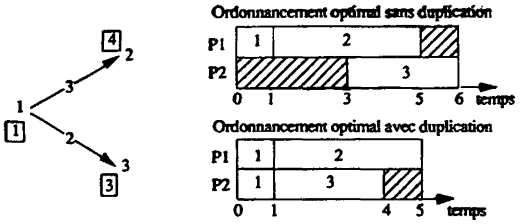


Figure 7.

temps de communication inférieurs aux temps d'exécution, le problème est NP-difficile, même dans le cas où les durées d'exécution et de communication sont unitaires [PICO 91].

Si la duplication est autorisée, le problème central distribué est NP-difficile même dans le cas d'une anti-arborescence de profondeur deux [CHRE 90 *b*]. Par contre si les temps de communication sont inférieurs aux durées des tâches, le problème est résolu par un algorithme de complexité $O(n^2)$, illustré par l'exemple des figures 8, 9, 10 [COLI 90].

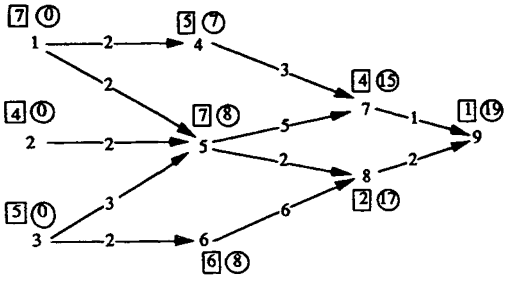


Figure 8. — A côté de chaque tâche figure sa durée d'exécution encadrée et sa date au plus tôt encerclée. Sur chaque arc figure le temps de communication.

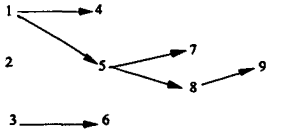


Figure 9.

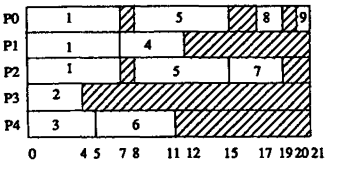


Figure 10.

La première phase de l'algorithme calcule, en balayant les sommets du graphe de précédence selon une liste topologique, une borne inférieure b_i de la date de début d'exécution de toute copie de la tâche i.

Si $P(i)$ désigne l'ensemble des prédécesseurs immédiats de la tâche i, les bornes b_i sont calculées comme suit : soit s une tâche de $P(i)$ telle que $b_s+p_s+c_{si}$ soit maximal, alors b_i est la plus grande des deux valeurs b_s+p_s et Max $\{b_k+p_k+c_{ki}/k \in P(i)-\{s\}\}$. La figure 8 rapporte ces bornes pour notre exemple.

La deuxième phase de l'algorithme construit le graphe critique des arcs de précédence (i, j) tels que $b_i+p_i+c_{ij}>b_j$. Il est clair que, s'il existe un ordonnancement pour lequel toutes les copies débutent leur exécution à leur borne inférieure, alors une copie de j et sa copie de i « fournisseur » doivent être placées sur la même machine. Le graphe critique est une famille d'arborescences. Pour notre exemple, il est donné sur la figure 9.

La troisième phase construit un ordonnancement au plus tôt pour lequel chaque copie est exécutée à sa borne inférieure calculée lors de la première phase. Cet ordonnancement utilise autant de processeurs que de feuilles de l'arborescence et le nombre de copies d'une tâche i est le nombre de feuilles de la sous-arborescence de racine i. L'ordonnancement optimal de notre exemple est représenté sur la figure 10.

Il apparaît donc que, même en l'absence de contraintes de ressources, les problèmes d'ordonnancement avec temps de communication sont difficiles. En présence d'un nombre limité de machines, des algorithmes approchés avec garantie de performance fondés sur des listes [HWA 89] ont été développés. Pour le problème central, un algorithme approché fondé sur la solution optimale d'un problème relaché fournit une garantie absolue dans le cas d'une arborescence [PICO 91]. Les méthodes exactes, à notre connaissance, concernent des problèmes spécifiques.

4. PROBLÈMES À RESSOURCES MULTIPLES

Les problèmes d'ordonnancement à contraintes de ressources renouvelables sont souvent appelés problèmes d'ateliers car on les rencontre dès qu'il faut gérer la production. Toutefois il s'agit de problèmes très généraux car ils apparaissent dans d'autres contextes, en particulier en Informatique. Le but de cette partie est de présenter la classification usuelle de ces problèmes (flow-shop, job-shop et open-shop) et surtout de décrire des méthodes permettant de les traiter. Nous y décrivons aussi les problèmes d'ordonnancement à contraintes cumulatives qui se posent dans les ateliers, en particulier, quand la

main d'œuvre est en quantité limitée. Le problème du voyageur de commerce qui surgit dès qu'il y a des réglages d'outils dont les durées sont dépendantes des pièces à usiner est mentionné.

4.1. Problèmes d'atelier

Dans le cas des problèmes d'atelier, une tâche est une opération, une ressource est une machine et chaque opération nécessite pour sa réalisation une machine. Dans le modèle de base de l'ordonnancement d'atelier, l'atelier est constitué de m machines, n travaux (jobs) disponibles à la date 0 doivent être réalisés, un travail i est constitué de n_i opérations, l'opération j du travail i est notée (i, j) où $(i, 1)<(i, 2)< \ldots <(i, n_i)$ si le travail i possède une gamme ($A<B$ signifie A précède B). Une opération (i, j) utilise la machine $m_{i,j}$ pendant toute sa durée $p_{i,j}$ et ne peut être interrompue.

Une classification peut s'opérer selon l'ordre d'utilisation des machines pour fabriquer un produit (*gamme de fabrication*). On rencontre : des ateliers à *cheminement unique* où toutes les gammes sont identiques (*flow-shop*), des ateliers à *cheminements multiples* où chaque produit ou famille de produits possède (*job-shop*) ou ne possède pas (*open-shop*) une gamme spécifique. Les dates de début des opérations (i, j) constituent les inconnues du problème et leur détermination définit l'*ordonnancement*. Les contraintes sont de type disjonctif; le choix d'une séquence sur une machine résout donc les conflits d'utilisation de la machine.

Le modèle de base peut être étendu de diverses façons en prenant en compte, par exemple : l'existence de dates échues impératives, l'interruption possible d'une opération (avec ou sans mémoire du travail effectué), les temps de préparation, une disponibilité des machines variable dans le temps, des ressources à capacité non unitaire, l'utilisation de plusieurs ressources pour une opération.

4.1.1. *Le flow-shop*

4.1.1.1. Le flow-shop statique de base

Nous rappelons que le problème de base du flow-shop est le cas particulier du problème de base d'ordonnancement d'atelier pour lequel le cheminement est unique : les n travaux utilisent les m machines dans l'ordre $1, 2, \ldots, m$. Un *ordonnancement de permutation* est un ordonnancement qui conserve le même ordre des travaux sur toutes les machines. Dans le cas de deux ou trois machines, le sous-ensemble des ordonnancements de permutation est

dominant mais dans le cas de m machines, la propriété n'est plus vraie. Sauf spécification contraire les techniques présentées dans la suite concerneront la recherche du meilleur ordonnancement de permutation. Cette restriction (interdiction de dépassements entre travaux) est souvent naturelle en pratique et sinon on utilise les techniques du problème de job-shop.

Selon une notation classique, ce problème s'écrit $[n/m/X, \pi\ 1, \ldots, \pi\ k/\partial]$ où n est le nombre de travaux, m le nombre de machines, X la nature des gammes (F pour un flow-shop); $\pi\ 1$ à $\pi\ k$ seront les contraintes additionnelles au problème de base défini ci-dessus; ∂ est le critère retenu (par exemple, minimiser la durée totale d'exécution).

4.1.1.2. Les travaux de S. M. Johnson

Johnson proposa, en 1954, un algorithme optimal pour le $[n/2/F/C_{\max}]$ [JOHN 54]. On y applique, pas à pas, la règle suivante : « si J_i et J_j sont deux travaux de durées respectives $(p_{i,1}, p_{i,2})$ et $(p_{j,1}, p_{j,2})$ sur les machines 1 et 2, et si $\min(p_{i,1}, p_{j,2}) \leqq \min(p_{i,2}, p_{j,1})$ alors J_i précède J_j ».

Appelons **J** cet algorithme de résolution du $[n/2/F/C_{\max}]$. On remarquera que **J** tend à placer en tête de l'ordonnancement les travaux ayant des temps opératoires plus faibles sur la première machine et, en fin d'ordonnancement, ceux ayant des temps opératoires plus faibles sur la seconde. Nous retiendrons aussi que **J** minimise la somme **Dn (S)** des temps morts sur la seconde machine associée à un ordre S. On montre également que : $\mathbf{Dn}(\mathbf{S}) = \max_{1 \leqq r \leqq n} \left(\sum_{i=1,r} \mathbf{p_{i,1}} - \sum_{i=1,r-1} \mathbf{p_{i,2}} \right)$.

Une approche de type chemin critique [McMA 71] conduit aussi au même résultat : on pourra intervertir, dans S, les deux travaux i et i+1 sans augmenter $C_{\max}$ si

$$\operatorname{Max}(p_{i+1,1}+p_{i,1}+p_{i,2},\ p_{i+1,1}+p_{i+1,2}+p_{i,2}) \leqq \operatorname{Max}(p_{i,1}+p_{i+1,1}+p_{i+1,2},\ p_{i,1}+p_{i,2}+p_{i+1,2})$$

En effet, si i précède i+1 la date de fin d'exécution de i+1 sur la seconde machine est :

$$C_{i+1,2} = \operatorname{Max}(C_{i-1,1} + p_{i,1}+p_{i+1,1}+p_{i+1,2},\ C_{i-1,1} + p_{i,1}+p_{i,2}+p_{i+1,2},\ C_{i-1,2}+p_{i,2}+p_{i+1,2})$$

et si i+1 précède i :

$$C_{i,2} = \operatorname{Max}(C_{i-1,1} + p_{i+1,1}+p_{i,1}+p_{i,2},\ C_{i-1,1} + p_{i+1,1}+p_{i+1,2}+p_{i,2},\ C_{i-1,2}+p_{i+1,2}+p_{i,2})$$

En comparant ces deux dates, on établit l'inégalité ci-dessus, équivalente à la condition de Johnson. Il est fondamental de se souvenir de ces différents aspects pour la compréhension de nombreux travaux de recherche relatifs à ce domaine. On trouve d'autres démonstrations de l'optimalité de **J** basées soit sur la notion de chaînes [COFF 76], soit sur la programmation dynamique [BELL 82].

Johnson proposa également une résolution d'un [$n/3/F$, conditions de dominance/$C_{\max}$] par l'algorithme **J'** suivant : on applique **J** sur deux machines fictives dont les pseudo-temps d'exécution sont respectivement $p'_{i,1} = p_{i,1} + p_{i,2}$ et $p'_{i,2} = p_{i,2} + p_{i,3}$, pour chaque travail i. La solution n'est optimale que si l'une des deux valeurs min $\{p_{i,1}\}$ ou min $\{p_{i,3}\}$ est supérieure ou égale à max $\{p_{i,2}\}$.

Giglio et Wagner étudièrent le comportement de **J'** sur des [$n/3/F/C_{\max}$] quelconques en 1964. On trouve une explication géométrique de **J** dans [CONW 67] et on sait que **J** résout aussi le [$n/2/F$, préemption/$C_{\max}$] [GONZ 78]. Johnson énonçait, en 1954, que si l'ordre trouvé par **J** sur les machines 1&2 et 2&3 était le même alors il était optimal sur 1&2&3. Burns et Rooker, en 1976, viendront d'une part tempérer ce jugement et montreront d'autre part que si **J** fournit le même ordonnancement sur 1&2, 2&3, 1&3 alors cet ordonnancement est optimal sur 1&2&3. De très nombreux travaux se sont déroulés depuis quarante ans relatifs aux $n/3/F$, conditions de dominance/$C_{\max}$ dûs notamment à Szwarc, Nabeshima... Des synthèses ont été effectuées par Achugbue et Chin, [ACHU 82], par Monma et Rinnooy Kan [MONM 83].

En ce qui concerne l'implémentation de **J**, nous connaissons deux algorithmes classiques **J1** et **J2** de complexités respectives $O(n^2)$ et O (nlogn) [CARL 88 *a*]. La dernière proposition en O (nlogn) est due à Kusiak [KUSI 86]. On notera qu'on ne sait toujours pas représenter de façon synthétique l'ensemble des solutions optimales lorsqu'elles sont multiples. Rappelons que Page, en 1961, fournit une analogie entre **J** et une fonction de classement par valeurs croissantes d'un indicateur de profil de pente $f(i)=A/\min(p_{i,1}, p_{i,2})$ où $A=1$ si $p_{i,2} \leq p_{i,1}$ et $A=-1$ sinon, ainsi qu'une autre analogie pour **J'** avec $f(i)=A/\min_{j=1,2}(p_{i,j}+p_{i,j+1})$ où $A=1$ si $p_{i,3} \leq p_{i,1}$ et $A=-1$ sinon.

4.1.1.3. L'influence de S. M. Johnson sur la résolution de [$n/m/F$, $*/C_{\max}$]

Depuis, des propositions d'algorithmes pour résoudre le problème de base à m machines paraissent régulièrement dans la littérature spécialisée.

Diverses variantes intéressent aussi les chercheurs; elles consistent à enrichir les conditions d'application de **J** ou **J'** sur les [$n/3/F$, $*/C_{max}$], à prendre en compte par exemple une succession sans attente des opérations ou des temps de transport inter-opérations ou des chevauchements d'opérations ou l'existence de temps de montage ou de démontage d'outils à chaque opération (dépendant ou non de la succession des travaux) ou des contraintes de précédence entre les travaux ou enfin l'existence de machines dominantes. D'autres critères furent aussi étudiés comme par exemple le retard maximum.

Nous avons effectué une étude de nombreux travaux où l'influence des idées de S. M. Johnson est manifeste [PROU 92]. Y figure une volumineuse bibliographie dont nous ne citerons ici que quelques éléments. Ceci nous permet d'avancer quelques propositions et réflexions pour la résolution du problème d'ordonnancement statique sur m machines, de n travaux à gamme identique, avec temps de montage ($S_{i,j}$) et de démontage ($R_{i,j}$) d'outils indépendants de l'ordre (no sequence dependent) : le [$n/m/F$, S_{nsd}, R_{nsd}/C_{max}] [PROU 87, 91 *a*].

Pour notre part, nous considérons que beaucoup d'algorithmes exacts ou heuristiques de résolution du $n/m/F$, contraintes diverses/C_{max} relèvent de quatre catégories (hormis les Procédures par Séparation et Evaluation complètes ou tronquées) :

– application de l'algorithme **J** (justifiée de façon optimale ou heuristique, par mise en évidence d'un indicateur du type **Dn** (S) ou par extrapolation du comportement de **J'**),

– génération d'un indicateur de profil de pente des temps opératoires (c'est une extrapolation du comportement de **J**),

– association de ces deux premiers aspects,

– permutation de sous-ensembles de travaux (y compris couplage).

Cette dernière catégorie nous intéresse moins ici. Il faut signaler toutefois qu'elle vient souvent en complément à l'une des trois premières.

Nous laisserons de côté les approches par la programmation en nombres entiers ou par les PSE où, parfois pourtant, **J** est présent comme dans le calcul des bornes de Nabeshima (1967) ou Han ([HAN 92]).

Nous illustrons les trois premières catégories par quelques exemples :

Sule et Huang, en 1983, prennent en compte à la fois les temps de montage, les temps opératoires et les temps de démontage d'outils. Les temps de montage et de démontage sont indépendants de l'ordre. La résolution est exacte pour [$n/2/F$, S_{nsd}, R_{nsd}/C_{max}] et approchée pour trois machines.

Les auteurs démontrent que, pour 2 machines, la durée de l'ordonnancement optimal est :

$$C_{\max} = C_{n,2} = \max_{1 \leqq u \leqq n} \left[\sum_{i=1}^{u} (S_{i,1} - S_{i,2} + p_{i,1}) - \sum_{i=1}^{u-1} (p_{i,2} + R_{i,2} - R_{i,1}) \right] + \sum_{i=1}^{n} (S_{i,2} + p_{i,2} + R_{i,2})$$

Cette expression ressemble fortement, à une constante près, à l'expression **Dn** (**S**) étudiée dans la démonstration de l'optimalité de **J**. La conséquence est immédiate : il suffit d'appliquer **J** à deux machines fictives bien particulières dont les pseudo-temps d'exécution seront respectivement

$$p'_{i,1} = S_{i,1} - S_{i,2} + p_{i,1} \qquad \text{et} \qquad p'_{i,2} = p_{i,2} + R_{i,2} - R_{i,1}.$$

Szwarc, en 1983, développa une heuristique en suivant une voie similaire pour l'étude des [$n/m/F$, décalage temporel/$C_{\max}$], c'est-à-dire dans le cas où on considère un délai minimal $a_{i,k}$ à respecter entre la fin de l'opération k–1 et le début de l'opération k, pour un travail i. Alors

$$C_{i,k} = \text{Max}\ (C_{i,k-1} + a_{i,k},\ C_{i-1,k}) + p_{i,k}.$$

On remarquera que ce modèle englobe aussi des contraintes du type : temps minimum $d_{i,k}$ à respecter entre les dates de début d'un travail i sur une machine et sur la suivante k, ou temps minimum $d'_{i,k}$ à respecter entre les dates de fin d'un travail sur une machine et sur la suivante, ou bien les deux à la fois : $a_{i,k} = \text{Max}\,(d_{i,k} - p_{i,k-1},\ d'_{i,k} - p_{i,k})$.

Nabeshima et Maruyama, la même année, résolvent de façon optimale le [$n/2/F$, S_{nsd}, R_{nsd}, décalages temporels minimaux entre les débuts et les fins d'opérations, temps de transport minimaux à respecter entre les deux opérations/$C_{\max}$] : ils construisent un problème artificiel à deux machines sur lequel on applique **J** en posant pour tout travail i :

$$p'_{i,1} = S_{i,1} - S_{i,2} + \text{Max}\,(d_{i,2},\ p_{i,1} + t_i,\ p_{i,1} + d'_{i,2} - p_{i,2})$$

et

$$p'_{i,2} = -p_{i,1} + p_{i,2} - R_{i,1} + R_{i,2} + \text{Max}\,(d_{i,2},\ p_{i,1} + t_i, p_{i,1} + d'_{i,2} - p_{i,2}).$$

En extrapolant les travaux de Page au [$n/m/F/C_{\max}$], Gupta propose en 1971 une heuristique utilisant pour tout travail i :

$$f(i) = \text{signe}\ \{p_{i,1} - p_{i,m}\} / \min_{1 \leqq j \leqq m-1} \{p_{i,j} + p_{i,j+1}\}$$

où signe $\{x\}$ est égal à -1 si $x<0$ et égal à 1 sinon.

Bonney et Gundry, en 1976, bâtissent un algorithme approché en particularisant, par une mesure simple, les profils de chaque travail lorsqu'on les considère individuellement et que leurs opérations se succèdent sans attente. Ainsi ils définissent, pour chaque travail, deux indicateurs de profil de pente en calculant les régressions linéaires, d'une part entre les points représentatifs des dates de début de chaque opération, d'autre part entre les points représentatifs des dates de fin de chaque opération. Ils se souviennent eux aussi, comme Palmer, du comportement de l'algorithme **J**, mais aussi de sa règle. Nous présentons ci-dessous cet algorithme, appelé Bonney2, pour une résolution heuristique du $[n/m/F/C_{\max}]$:

début
Lire les données
Calculer pour chaque travail i les indices de pente de début (« start-slope$_i$ ») et construire le vecteur des pseudo-temps d'exécution sur une machine fictive 1 par $p'_{i,1}$=« start-slope$_i$ »
Calculer pour chaque travail i les indices de pente de fin (« end-slope$_i$ ») et construire le vecteur des pseudo-temps d'exécution sur une machine fictive 2 par $p'_{i,2}$=« end-slope$_i$ »
Appliquer l'algorithme **J-inverse** :
a) Trouver la valeur *maximale* des $p'_{i,j}$ pour l'ensemble des travaux restant à ordonnancer. (Initialement les n travaux sont concernés).
b) S'il s'agit de $p'_{k,1}$ placer le travail k en début de l'ordonnancement restant; s'il s'agit de $p'_{k,2}$ placer le travail k en fin de l'ordonnancement restant.
c) Supprimer le travail k de l'ensemble des travaux encore à placer et aller en *a*).
fin

Campbell, Dudek et Smith, en 1970, fournissent une solution approchée (algorithme CDS) au problème $n/m/F/C_{\max}$, en étendant le comportement de **J'**. En fait ils créent m–1 sous problèmes fictifs à deux machines, résolvent chacun d'entre eux grâce à **J** et conservent le meilleur ordonnancement parmi les m–1, comme solution du problème originel. Cette heuristique se révèla extrêmement fiable dans les années qui suivirent.

Dans un contexte d'atelier de réparation organisé en flow-shop où de nombreux $p_{i,j}$ sont nuls, Jhaveri et Foote, [JHAV 91], annoncent la supériorité de l'algorithme de Petrov (1966), basé sur **J** et modifié par Radharaman, [RADH 86], par rapport à CDS et à la procédure de type quatre de Nawaz, Enscore et Ham ([NAWA 83]).

Les performances des heuristiques, où les machines sont remplacées par deux machines fictives et où des temps d'exécution fictifs sont calculés à partir des temps opératoires originaux, furent étudiées par Rock et Schmidt [ROCK 82].

Proust *et al.*, en 1987, dans une synthèse de CDS et des travaux de Sule et Huang, ont proposé un algorithme approché pour la résolution du $[n/m/F, S_{\text{nsd}}, R_{\text{nsd}}/C_{\max}]$: les pseudo-temps opératoires, pour chaque sous-

problème k, sont respectivement

$$p'^{k}_{i,1} = \sum_{j=1}^{k} p_{i,j} - S_{i,m-k+1} + S_{i,1} \quad \text{et} \quad p'^{k}_{i,2} = \sum_{j=m-k+1}^{m} p_{i,j} + R_{i,m} - R_{i,k}$$

Pour l'instant, cette approche fournit de meilleurs résultats que celle qui tente de générer des indicateurs de profil de pente ou que d'autres qui adoptent une démarche de séparation et d'évaluation basée sur des considérations de borne inférieure du $C_{\max}$, telles que les heuristiques de Szwarc (1983) ou de Townsend (1977). Han améliora ces résultats dans le cas du [$n/m/F$, S_{nsd}, R_{nsd}, avec ou sans machines goulets/$C_{\max}$] en 1992.

On ne peut terminer ce survol sans rappeler l'existence de l'algorithme optimal de Mitten, proposé en 1959, pour la résolution du [$n/2/F$, décalages temporels/$C_{\max}$] (cas particulier du modèle de Szwarc évoqué ci-dessus) : on applique **J** sur un problème fictif à deux machines où pour chaque travail i :

$$p'_{i,1} = p_{i,1} + \text{Max}\left(d_{i,2} - p_{i,1},\ d'_{i,2} - p_{i,2}\right)$$

et

$$p'_{i,2} = p_{i,2} + \text{Max}\left(d_{i,2} - p_{i,1},\ d'_{i,2} - p_{i,2}\right)$$

et l'on utilise un indicateur de type **Dn** (**S**). Il permit à Sidney et Monma, en 1979, de résoudre de façon optimale le [$n/2/F$, contraintes de précédence de type série-parallèle/$C_{\max}$], et McMahon ([McMA 92]) utilise ce dernier pour la résolution optimale du [$n/2/F$, contraintes de précédence de type quelconque/$C_{\max}$] dans une approche de type PSE; cette approche se révèlerait plus efficace que celle de Hariri et Potts (1984).

En 1990, Baker effectue une présentation sous différents angles de l'algorithme de Mitten et en déduit une approche optimale en deux phases pour l'ordonnancement de groupes de travaux où il existe des temps de réglage lors de la mise en fabrication de chaque groupe [BAKE 90].

Signalons enfin que **J** est bien souvent sous-jacent dans les algorithmes de Leisten ([LEIS 90]) proposés pour fournir de bonnes solutions au [$n/m/F$, capacités de stockage limitées entre les machines/$C_{\max}$].

4.1.1.4. Flow-shop sans attente

On impose, dans un flow-shop sans attente, que les opérations s'effectuent sans attente, c'est-à-dire, que la pièce usinée i passe de la machine j à la machine j+1 sans temps d'immobilisation. Le problème consiste à minimiser le temps total d'usinage. Dans la figure 11, on a représenté une solution réalisable avec m=3 et n=3.

Figure 11.

Le temps est dans le sens du défilement horizontal et les trois bandes représentent les trois machines. L'ordre de passage choisi ici est (J_1, J_2, J_3) et la quantité à minimiser est la somme des temps morts sur la première machine augmentée de la durée totale du travail placé en dernière position (le $n/m/F$, « no-waiting »/C_{max}). Ce problème peut être exprimé sous la forme d'un problème de Voyageur de Commerce ou TSP (Travelling Salesman Problem) [GILM 64]. Rappelons brièvement l'énoncé du TSP : étant donné un graphe connexe sans boucle dont les arcs sont valués par un coût, il s'agit de chercher un circuit Hamiltonien (encore appelé tour) de coût minimal. Dans le cas des problèmes de routage ou de tournées, la matrice des valuations est positive et souvent symétrique. Ce problème est l'un des problèmes d'optimisation combinatoire NP-difficile les plus étudiés depuis fort longtemps. On peut citer, comme référence historique, le problème du tour du cavalier sur l'échiquier qui a été étudié notamment par Euler en 1759 et par Vandermonde en 1771. L'ouvrage le plus complet sur ce sujet est le livre de [LAWL 86].

Les méthodes de résolution de TSP sont de deux types :

– *méthodes exactes :*

A l'instar de bon nombre de problèmes d'optimisation combinatoire, le TSP peut se mettre sous forme d'un programme linéaire en nombres entiers, mais le nombre exponentiel des contraintes de tour rend la résolution très difficile. Pour une formulation complète, *voir* par exemple [LAWL 86]. En 1986, M. Padberg et G. Rinaldi [PADB 86] ont présenté un algorithme basé sur des méthodes polyédrales associées à une procédure du type « branch

and cut ». Cette méthode a permis de résoudre un problème à 532 villes et, plus récemment, à 4×532 villes. Cependant, il faut savoir que cette méthode est lourde à mettre en œuvre et nécessite une très bonne initialisation. De plus, la procédure n'atteint pas toujours l'optimum si on limite son temps d'exécution.

– *des méthodes heuristiques :*

Comme nous venons de le voir, une méthode heuristique fournissant une solution sous-optimale est indispensable, même dans le cas où l'on veut atteindre l'optimum. Il en existe deux grandes familles : les heuristiques par construction progressive et les heuristiques d'amélioration par voisinage.

L'heuristique par construction progressive la plus connue est l'heuristique des plus proches voisins qui donne, en général, des résultats très moyens; elle est donc réservée soit à l'initialisation d'autres techniques, soit aux cas peu exigeants. Une heuristique itérative de type glouton est la méthode de l'élastique due à B. Angeniol *et al.* [ANGE 87]. On se place dans le plan euclidien, les nœuds sont donc des points du plan et la matrice des valuations est celle des distances euclidiennes entre les nœuds. L'élastique est une ligne polygonale fermée dont les sommets sont proches des nœuds du parcours. Le principe est d'étirer l'élastique passant déjà près de p nœuds de façon à ce qu'il se trouve maintenant près de $p+1$ nœuds parmi les n donnés. A la fin de l'algorithme, après un certain nombre d'itérations de cette procédure, on obtient un tour de coût proche de l'optimum. Associée à un recuit simulé cette méthode donne d'assez bons résultats. Les heuristiques du type améliorations successives sont les plus utilisées et sont, en général, assez efficaces (*voir* 1.2.2).

Le meilleur exemple est la procédure OPT-k de Lin et Kerninghan [LIN 71]; celle-ci consiste à effectuer des échanges dans l'ordre de parcours du tour. La complexité (expérimentale) est de l'ordre de $O(n^{2,2})$. Le problème des minima locaux est central dans ce genre de méthode et il y a plusieurs moyens de le contourner. On peut faire varier le tour initial, ce qui permet d'explorer plusieurs minima locaux. On peut aussi utiliser plusieurs procédures différentes et les faire fonctionner en cascade, ou encore bloquer certains arcs (méthodes tabou).

Nous noterons enfin que les réseaux neuronaux, qui s'apparentent aux méthodes par « améliorations successives », sont également appliqués à ce problème. On pourra trouver de nombreux exemples de ces techniques d'optimisation combinatoire dans les actes de Neuro Nîmes de 1989 à 1992.

4.1.1.5. Perspectives

Pour l'ensemble des problèmes de type flow-shop une exploration des travaux où l'on cherche une borne inférieure du critère reste à faire. On connaît l'importance de ces bornes inférieures pour l'aide à la mise au point de bonnes heuristiques.

D'autres problèmes de cette famille nécessiteraient une plus grande attention comme le [$n/m/F$, $d_i>0$, S_{nsd}, R_{nsd}/T_{max}] ou le [$n/m/F$, S_{nsd}, R_{nsd}, succession sans attente/C_{max}] ou le [$n/m/F$, S_{nsd}, R_{nsd}, contraintes de précédence de type série-parallèle/C_{max}] ou le [$n/m/F$, S_{sd}, succession sans attente, $d_i>0/T_{max}$] vu son intérêt industriel dans l'agro-alimentaire en particulier.

Des études fines sur ces sujets nous paraissent fondamentales même s'il est vraisemblable que, dans la réalité des ateliers, les algorithmes résultants ne seront que des briques de base de logiciels d'aide à la décision interactifs. En effet on peut constater [PROU 89] que beaucoup de problèmes réels, s'ils sont globalement de type flow-shop, présentent bien souvent, étage par étage, des structures locales de type « machines à exemplaires multiples » (*cf.* 3.2).

Enfin la mise au point d'heuristiques robustes nécessite que l'on dispose d'un logiciel ouvert offrant la définition des plans d'expérience, la possibilité de programmation des algorithmes dans le langage de son choix, la visualisation et les tests de validation des résultats. Il faut alors intégrer gestionnaire d'écrans, éditeur, compilateurs, logiciels de statistique et d'analyse de données, grapheur... Pour cette intégration nous disposons maintenant de ce « logiciel de logiciels » [PROU 91 *b*].

4.1.2. *Le job-shop*

Les premiers résultats théoriques concernant le job-shop datent des années cinquante avec la résolution du problème à 2 machines et d'un cas particulier du problème à 3 machines. On trouve sa définition formelle dans l'ouvrage « Industrial Scheduling » de Muth et Thompson, paru en 1963, où trois exemples de job-shop, considérés depuis comme données tests sont proposés. Les deux premiers ont été résolus dans les années soixante-dix. En revanche, il a fallu attendre les années quatre-vingts pour le troisième, l'optimalité de la meilleure solution connue, due à Lageweg, n'ayant été prouvée qu'en 1986 [CARL 89 *b*]. Aussi, depuis vingt-cinq ans, les recherches sur ce problème sont nombreuses (*cf.* par exemple [PINS 88]).

Le job-shop est NP-difficile au sens fort. Aussi, la plupart des méthodes proposées pour sa résolution sont-elles du type procédures arborescentes

basées sur une modélisation par un graphe disjonctif. Néanmoins, d'autres modélisations ont été proposées, parmi lesquelles la programmation linéaire en nombres entiers, la génération d'ordonnancements actifs, les approches géométriques. On a aussi utilisé récemment les approches polyédrales, les méthodes tabou et de recuit simulé. Contrairement au cas particulier du flow-shop, le job-shop se prête d'autre part mal à des caractérisations analytiques. Nous nous restreignons à la présentation de deux familles de résultats.

4.1.2.1. Minimisation de la durée d'un job-shop

Dans le problème de base de type job-shop, n travaux doivent être exécutés sur m machines, sous des hypothèses quasi-identiques à celles du flow-shop. La seule différence – et elle est de taille ! – est que maintenant les séquences opératoires relatives aux différents travaux peuvent être distinctes.

Nous ne considérerons, dans le cadre de ce paragraphe, que le critère « durée totale ». Les séquencements des opérations sur les différentes machines doivent être déterminés dans le but de minimiser cette durée.

Deux cas particuliers fondamentaux du problème de job-shop sont résolus polynomialement : le cas de 2 machines et le cas de 2 travaux.

Job-Shop à 2 machines

Cet algorithme, dû à Jackson, constitue une extension de celui de Johnson pour la résolution d'un problème de flow-shop n'utilisant que deux machines $[n/2/F/C_{\max}]$. Il traite de plus le cas où certains travaux peuvent n'avoir à passer que sur une des deux machines. Notant $M1$ et $M2$ les deux machines, on peut alors partitionner l'ensemble des travaux en 4 classes : $O1$: ensemble des travaux ne passant que sur $M1$, $O2$: ensemble de ceux ne passant que sur $M2$, $O12$: ensemble des travaux passant sur $M1$ puis sur $M2$, $O21$: ensemble de ceux passant sur $M2$ puis sur $M1$.

On montre alors qu'un ordonnancement optimal peut être déterminé en adoptant les séquencements $O12$-$O1$-$O21$ sur $M1$ et $O21$-$O2$-$O12$ sur $M2$, où les travaux de $O12$ et $O21$ sont ordonnancés par la règle de Johnson.

Job-Shop à 2 travaux

Historiquement, c'est en 1955 qu'Ackers et Friedman montrèrent que, pour le problème à 2 travaux, une condition nécessaire et suffisante pour qu'un séquencement réalisable soit optimal est que ce séquencement ne génère aucun temps mort « machine ».

En 1956, Ackers proposa une solution graphique du problème en exprimant tout ordonnancement réalisable comme un chemin dans un rectangle du

plan (xOy), puis Szwarc, en 1960, présenta une méthode de programmation dynamique pour déterminer un chemin optimal.

Des études théoriques sur la base de cette approche graphique ont été également réalisées en 1963 par Hardgrave et Nemhauser pour le problème général. En 1973, Shankar et Turksen proposèrent une décomposition de ce même problème en C_n^2 problèmes à deux travaux, formalisant ces derniers à l'aide d'un système d'équations booléennes qui permet de restreindre la détermination d'un ordonnancement optimal à celle d'ordonnancements dominants. Plus récemment, Brucker, reprenant cette approche graphique, proposa un algorithme de complexité O (n_1 n_2 log (n_1 n_2)) (n_1 et n_2 désignant respectivement le nombre d'opérations des travaux J_1 et J_2) permettant la résolution du [$2/n/J/C_{\max}$]. Comme Ackers, il montre que le problème peut être formalisé comme la recherche d'un plus court chemin dans le plan (xOy) sur lequel sont disposés des obstacles infranchissables, mais contournables, figurant l'exécution simultanée et impossible des 2 travaux sur la même machine. Sur chacun des axes figure la gamme d'un travail. Nous reproduisons sur la figure 12 un exemple proposé dans la littérature pour lequel les travaux J_1 et J_2 ont respectivement n_1=4 et n_2=3 opérations.

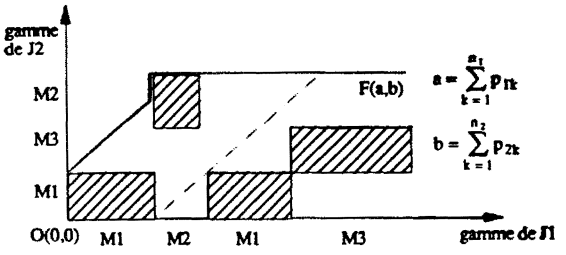


Figure 12.

Un ordonnancement réalisable correspond alors à un chemin de O à F et possède les propriétés suivantes : 1) les segments constitutifs de ce chemin sont soit parallèles à l'un des axes (cas où un seul des deux travaux est exécuté) soit diagonaux (cas où les deux travaux sont exécutés en parallèle) et 2) les obstacles sont infranchissables, une machine ne pouvant exécuter plus d'une opération à la fois et la préemption étant interdite; en contrepartie, ils sont contournables.

Brucker propose alors la transformation de ce problème en la recherche d'un plus court chemin dans un réseau approprié. Il est alors résolu en temps

O (plogp) si p est le nombre d'obstacles en utilisant une structure d'arbre équilibré. Des travaux visant à l'extension de ce principe au problème général sont en cours.

Résolution exacte du job-shop général

Les méthodes ayant donné les résultats les plus probants jusqu'à maintenant sont celles utilisant une modélisation par graphe disjonctif. On peut attribuer à Roy et Balas les premières applications de ce type de modélisation aux problèmes de job-shop.

Le principe en est le suivant : deux opérations utilisant une même ressource ne peuvent être exécutées simultanément et sont alors dites en disjonction. Le problème peut alors être modélisé par un *graphe disjonctif* $G'=(G, D)$ où :

– $G=(X, U)$ est le graphe correspondant à la partie « conjonctive » du graphe G' où X est l'ensemble des opérations des travaux à ordonnancer, (on adjoint une source ° et un puits noté *) et U est un ensemble d'arcs tel que si (i, k) et (i, j) sont deux opérations consécutives dans la séquence opératoire relative à un travail i, alors l'arc $\{(i, k), (i, j)\} \in U$.

– D est un ensemble de disjonctions : si (i, k) et (l, j) sont deux opérations en disjonction, on leur associe la paire d'arcs « disjonctifs »

$$[(i, k), (l, j)] = [\{(i, k), (l, j)\}; \{(l, j), (i, k)\}]$$

La figure 13 montre un exemple de graphe disjonctif associé à un problème de job-shop comportant deux travaux et deux machines.

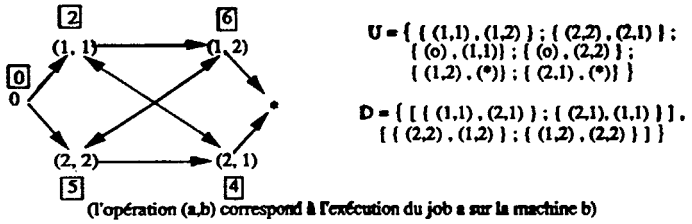


Figure 13.

En notant $p_{i,j}$ la durée de la tâche représentée par le sommet (i, j), nous pouvons, à partir de cette modélisation, définir *un ordonnancement* sur le graphe disjonctif G' comme *un ensemble de potentiels* $T=\{t_{i,k}/(i, k) \in X\}$ tel que :

– les contraintes conjonctives soient satisfaites :

$$\forall \{(i, k), (i, j)\} \in U, \qquad t_{i,j} - t_{i,k} \geqq p_{i,k},$$

– les contraintes disjonctives soient satisfaites :

$$\forall [(i, k), (j, k)] \in D, \qquad t_{j,k} - t_{i,k} \geqq p_{i,k} \qquad \text{ou} \qquad t_{i,k} - t_{j,k} \geqq p_{j,k}.$$

Pour construire un ordonnancement, il est nécessaire *d'arbitrer* » les contraintes disjonctives, c'est-à-dire de choisir, pour chaque disjonction $[(i, k), (j, k)]$ si (i, k) passera avant ou après (j, k) sur la machine correspondante, de manière à déterminer un séquencement sur chacune des machines.

Un arbitrage A est un ensemble d'arcs disjonctifs tel que si

$$\{(i, k), (j, k)\} \in A,$$

alors $\{(j, k), (i, k)\} \notin A$, l'appartenance de l'arc $\{(i, k), (i, k)\}$ à A imposant l'exécution de la tâche (i, k) avant la tâche (j, k). On associe à tout arbitrage A le graphe conjonctif G_A formé des arcs de U et des arcs de A. Un arbitrage A est dit complet si toutes les disjonctions de D sont arbitrées. Un arbitrage A est dit compatible si le graphe conjonctif associé est sans circuit.

Un ordonnancement correspond à un arbitrage complet compatible. Sa durée est alors égale à la valeur d'un plus long chemin dans le graphe conjonctif associé. Un arbitrage complet compatible pour le problème de la figure 13 est $\{\{(1, 1), (2, 1)\}, \{(2, 2), (1, 2)\}\}$. La durée de l'ordonnancement correspondant est 11.

Utilisant cette formalisation, Bertier et Roy proposèrent une méthode de résolution de type PSEP utilisant un principe de pénalités pour le choix de la disjonction à arbitrer.

Soit $[(i, k), (j, k)]$ une disjonction non arbitrée. $F_{i,k}$ et $F_{j,k}$ désignant les dates au plus tard associées aux tâches (i, k) et (j, k) dans le graphe conjonctif courant, les quantités $\beta_{i,j}=\max(0, t_{i,k}+p_{i,k}-F_{j,k})$ et $\beta_{j,i}=\max(0, t_{j,k}+p_{j,k}-F_{i,k})$ correspondent aux augmentations d'un plus long chemin du graphe, suivant qu'on choisisse d'exécuter (i, k) avant (j, k) ou (j, k) avant (i, k). Comme de toute manière un des deux choix devra être fait, ils associent à la disjonction $[(i, k), (j, k)]$ la pénalité $k_{i,j}=\min(\beta_{i,j}, \beta_{j,i})$ correspondant à l'augmentation minimale d'un chemin critique, et parallèlement la quantité $r_{i,j}=\beta_{i,j}-\beta_{j,i}$ correspondant au « regret » obtenu quand on ne choisit pas l'arc $\{(i, k), (j, k)\}$ correspondant au $\min(\beta_{i,j}, \beta_{j,i})$. La disjonction à séparer au nœud courant sera donc celle de $r_{i,j}$ maximal et, en cas d'égalité, celle de $k_{i,j}$ maximal.

Balas proposa également en 1969 une méthode arborescente pour le problème disjonctif. Cette méthode est basée sur les deux résultats suivants :

– soit G un graphe conjonctif associé à un arbitrage complet compatible, et soit c un chemin critique dans G. Tout graphe conjonctif G' obtenu à partir de G par inversion d'un arc disjonctif de c est encore réalisable.

– soit G un graphe conjonctif de chemin critique de valeur v. Dans tout graphe conjonctif G' de chemin critique de valeur $v'<v$, au moins un arc disjonctif d'un des chemins critiques de G est inversé.

Balas propose alors la procédure suivante d'énumération implicite de graphes réalisables. Soit G_A un graphe conjonctif réalisable associé à l'arbitrage complet compatible A, et c un chemin critique dans G_A. Si c ne contient aucun arc disjonctif de A alors, en vertu des résultats précédents, aucun graphe $G_{A'}$ correspondant à un ordonnancement de durée inférieure ne peut être généré par inversion d'un arc disjonctif de A dans A'. Dans le cas contraire, l'inversion d'un arc disjonctif de A dans A' génère un graphe conjonctif lui-même réalisable.

Balas définit ainsi pour chacune des disjonctions de c potentiellement inversibles une pénalité basée sur une évaluation par défaut du chemin critique associé à l'inversion, et permettant de guider le branchement à chaque nœud de l'arbre de recherche. Cette procédure génère alors une arborescence de réseaux réalisables, conduisant à un ordonnancement optimal. Pratiquement, cette méthode, pionnière dans le domaine avec celle de Bertier et Roy, donne des résultats assez éloignés des optima pour les « benchmarks » de Muth et Thompson, bien que des arborescences atteignant 200 000 nœuds aient été développés.

Carlier, en 1975 [CARL 75, 78], puis Carlier et Pinson à partir de 1986 [CARL 89 *b*, 90], reprirent et améliorèrent cette approche de modélisation pour les problèmes disjonctifs, introduisant de plus la notion d'arbitrages triviaux. Leurs méthodes sont basées sur une relaxation du problème de job-shop à m problèmes à une machine. Ils montrent que si un ensemble de conditions sont satisfaites pour un sous-ensemble de tâches utilisant la même machine, alors l'une de ces tâches peut être positionnée par rapport à toutes les autres dans tout séquencement réalisable, permettant ainsi l'arbitrage « trivial » des contraintes disjonctives associées (*voir* 3.1.3). Ils utilisèrent cet outil dans plusieurs procédures arborescentes de type PSES dont le schéma de séparation est basé, pour la dernière, sur l'arbitrage de sous-ensembles de tâches dits critiques, *i. e.* d'évaluation par défaut locale maximale. Cette évaluation est, quant à elle, basée sur celle des problèmes à une machine associés. Ces méthodes fournissent des résultats très satisfaisants pour des problèmes de taille relativement importante, et ont de plus résolu de façon

optimale le célèbre problème à 10 travaux/10 machines proposé par Muth et Thompson en 1963. Notons enfin que Brucker [BRUC 91] et son équipe travaillent maintenant sur une procédure arborescente utilisant le concept d'arbitrage trivial et un schéma de séparation inspiré de celui proposé par Grabowski [GRAB 82] pour le problème de flow-shop.

Résolution approchée du job-shop

De nombreuses tentatives de résolution du cas général par un algorithme de liste utilisant des règles de priorité ont été proposées depuis 1950. Le principe général de ces heuristiques par construction est le suivant : on ordonnance à chaque instant t – où une machine et au moins une tâche sont disponibles – la tâche de priorité maximale conformément à la règle de priorité retenue. Parmi les différentes règles de priorités testées, on peut citer **FIFO** : sélection de l'opération disponible le plus tôt, **SPT** : sélection de l'opération de plus petit temps opératoire, **LPT** : sélection de l'opération de plus grand temps opératoire, **MTR** : sélection de l'opération ayant le plus grand nombre de tâches restant à exécuter dans sa séquence opératoire et **MWKR** : sélection de l'opération correspondant à la plus grande quantité de travail restant à exécuter.

Parmi les travaux réalisés dans ce sens, on peut citer Lawrence (*cf.* [PINS 88]), et une synthèse comparative de Adams, Balas et Zawack, montrant que le dénominateur commun de ces heuristiques est leur absence de stabilité, c'est-à-dire la fluctuation de leur performance en fonction des jeux de données testés.

Plus récemment, ces derniers proposèrent une méthode de résolution approchée de ce même problème, « The Shifting Bottleneck Procedure » [ADAM 88], dont les résultats sont très satisfaisants même sur des problèmes de taille importante. Le principe de cette méthode est le suivant : partant du problème global, ils ordonnancent localement les machines une à une, chacune de manière optimale (en utilisant la méthode arborescente proposée par Carlier en 1982). Ils propagent pour chaque nouveau séquencement les contraintes résultantes à l'aide du graphe conjonctif associé au problème global. L'ordre dans lequel les machines seront séquencées dépend d'une évaluation des goulets d'étranglement associés à chacune d'entre elles. Chaque fois qu'une nouvelle machine est ordonnancée, ils réoptimisent le séquencement de toute machine déjà ordonnancée dont l'ordre induit peut être amélioré. Dans la méthode arborescente associée, chaque nœud correspond donc à un sous-ensemble de machines ordonnancées. Dans le but de limiter la taille de l'arborescence générée, une fonction de pénalité, calculée en

chaque nœud, mesurant sa déviation par rapport au goulet d'étranglement, et pondérée en fonction de son niveau dans l'arbre de recherche, est utilisée.

4.1.2.2. Minimisation de la somme des retards d'un job-shop

La méthode présentée ici est une méthode par décomposition qui utilise la technologie de groupe.

Décomposition d'un atelier en îlots de fabrication

L'objectif de la décomposition spatiale d'un atelier est de regrouper les machines en îlots de fabrication et les produits à fabriquer en familles de telle sorte que :

– il y ait autant d'îlots de fabrication que de familles de produits,

– les îlots et les familles de produits soient mis en correspondance biunivoque,

– tout produit d'une famille soit traité essentiellement sur les machines de l'îlot associé et rarement sur celles d'autres îlots.

C'est un problème de classification croisée que l'on peut présenter brièvement de la façon suivante : étant donnée une matrice A, dont l'élément $a_{i,j}$ vaut 1 si le produit i utilise la machine j et 0 sinon, il s'agit de permuter les lignes et les colonnes de la matrice A de telle sorte que les lignes et les colonnes des mêmes familles soient regroupées; les blocs diagonaux correspondent alors aux machines et aux produits de la même famille, le critère à maximiser est égal à la somme du nombre de 1 des blocs diagonaux et du nombre de 0 à l'extérieur des blocs diagonaux. Cette décomposition est illustrée par la figure 14.

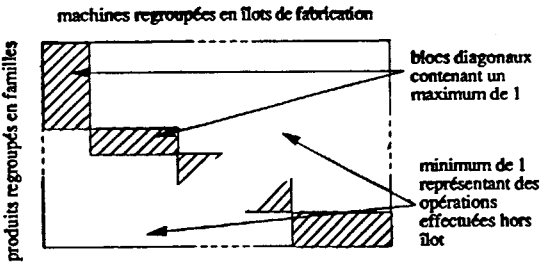


Figure 14.

Plusieurs algorithmes approchées ont été développés pour minimiser ce critère, en particulier ([GARC 85, 86]).

Utilisation des îlots de fabrication pour ordonnancer

Plusieurs méthodes ont été proposées dans [PORT 88], en particulier une méthode dite de décomposition spatiale et une méthode dite de décomposition spatiale et temporelle.

Les deux méthodes demandent une décomposition *a priori* de l'atelier en îlots de fabrication à partir du carnet de commandes correspondant aux produits à ordonnancer (ordonnancement de type job-shop non périodique). Elles ont toutes deux été conçues dans l'optique de minimiser la somme des retards des travaux par rapport aux dates échues, mais on pourrait construire des méthodes analogues pour d'autres critères. Dans les deux méthodes, on remplace un produit dont la gamme de fabrication utilise des machines de plusieurs îlots par une suite de sous-produits auxquels on attribue des caractéristiques identiques à celles des produits (en particulier, une date de disponibilité et une date échue, calculées artificiellement en partageant la marge du produit entre les différents sous-produits qui le composent).

Dans la méthode de décomposition spatiale, on ordonnance une première fois indépendamment dans chaque îlot les produits et les sous-produits correspondants; on regarde produit par produit si l'ordonnancement proposé est réalisable (dans le cas contraire, il existe un chevauchement entre deux sous-produits, la suite de la gamme dans l'îlot suivant commençant avant la fin des opérations du même produit programmées en retard dans l'îlot précédent); on modifie de manière plus ou moins importante les dates échues d'un ou plusieurs sous-produits associés aux produits concernés par un chevauchement (selon les variantes de la méthode); on réordonnance les îlots concernés; on itère sur les trois dernières phases jusqu'à convergence de la méthode.

Dans la méthode de décomposition spatiale et temporelle, on ordonnance à chaque itération uniquement des sous-produits « accessibles » d'un des îlots (c'est-à-dire les produits ou les sous-produits dont le début de gamme est déjà ordonnancé sur un autre îlot), en figeant les opérations terminées avant l'instant t, en introduisant les sous-produits accessibles disponibles avant l'instant $t+\Delta$ et en les ordonnançant de manière optimale conjointement avec les opérations non terminées à l'instant t en ignorant les autres sous-produits. Les chevauchements sont impossibles car on impose des dates de fin impérative aux sous-produits pour lesquels les opérations suivantes sont déjà ordonnancées sur un autre îlot. Les choix de t et de $t+\Delta$ sont faits de telle sorte que l'on ne construise que des ordonnancements actifs, c'est-à-dire qu'il n'existe pas d'intervalle où une machine est disponible et où une

opération pourrait être avancée sans déplacer d'autres opérations [BAKE 74]. D'une itération à l'autre, on tourne sur les différents îlots en sautant ceux qui entre t et $t+\Delta$ n'ont pas de nouveaux produits accessibles à introduire, puis t progresse à $t+\Delta$...

Ces deux méthodes par décomposition fournissent des solutions approchées. La seconde méthode consomme moins de temps pour des performances moyennes comparables. On ne peut apprécier la qualité réelle de la solution trouvée que sur des petits exemples pour lesquels il est possible d'obtenir une solution optimale (procédure arborescente). Pour ces petits exemples, l'écart à l'optimum est relativement peu important pour des gains en temps de calcul considérables. Les méthodes par décomposition sont d'autant meilleures que la décomposition en îlots laissent moins de liens résiduels (ou encore qu'il y a peu de produits décomposés en sous-produits). Une série d'expériences a montré un gain d'environ 30 % sur la valeur moyenne du critère somme des retards en faveur des méthodes de décomposition comparée à la meilleure solution obtenue par une série d'heuristiques utilisant des règles de priorité ([CHU 92 *a*]).

4.1.3. *L'open-shop*

C'est un modèle d'atelier moins contraint que le job-shop et le flow-shop, car l'ordre d'exécution des tâches d'un travail y est libre. Comme précédemment, m machines sont disponibles pour traiter n travaux. Chaque travail comprend m tâches. La tâche j du travail i doit être traitée par la machine j, en une durée $p_{i,j}$. Si une machine est inutile pour un travail i, on convient alors que $p_{i,j}=0$.

L'open-shop modélise des problèmes courants, comme l'organisation d'examens médicaux dans un hôpital ou la planification d'un atelier de réparation automobile. Dans ces deux exemples, les travaux sont respectivement les patients et les voitures, les tâches sont les examens médicaux et les réparations à effectuer, les machines étant les salles ou appareils d'examen (radio, endoscopie, ...) et les postes de travail (graissage, lavage, ...).

Des problèmes d'open-shop à des échelles de temps de l'ordre de la milliseconde surviennent aussi en télécommunications, notamment dans les systèmes par satellite où les transferts sont réalisés par paquets [CARL 88 *b*].

On peut remarquer que le problème de l'open-shop peut être interprété comme un problème à tâches indépendantes dans lequel chaque tâche nécessite simultanément deux ressources : le travail et la machine. Ces deux ressources introduisent chacune des contraintes disjonctives. Il en résulte une

certaine symétrie entre les travaux et les machines, comme cela apparaît dans l'exemple ci-après (interruption des tâches).

4.1.3.1. Variantes et extensions

Compte tenu de la symétrie entre travail et machine qui résulte de l'absence d'ordre entre les tâches d'un même travail, l'interruption d'une tâche, lorsqu'elle est possible, peut être utilisée soit pour la préemption du travail, soit pour la préemption de la machine, soit pour les deux à la fois. L'interdiction de la préemption du travail peut être justifiée par des coûts de transport trop élevés entre machines. L'interdiction de la préemption de la machine peut être justifiée par des coûts de reconfiguration de la machine trop élevés pour passer d'un type de travail à un autre.

Lorsque les deux types de préemption sont autorisés les problèmes d'open-shop sont plus faciles. Par contre dès que la préemption est totalement ou partiellement interdite ces problèmes sont nettement plus difficiles (*voir* 4.1.3.2). Certains d'entre eux, dits sans attente, obligent à traiter sans interruption un travail, une fois démarré : de tels cas sont fréquents dans le traitement à chaud des métaux.

4.1.3.2. Complexité et algorithmes exacts pour l'open-shop

La théorie de la complexité a permis de montrer que la plupart des variantes de l'open-shop sont NP-difficiles. Nous insistons donc sur ces résultats de complexité, puisqu'il n'existe que peu de cas où l'on dispose d'algorithmes polynomiaux.

La minimisation de la durée totale est le cas le plus courant, étudié par Gonzalez et Sahni dès 1976 [GONZ 76]. Ces auteurs ont montré que le problème de minimisation de $C_{\max}$ est NP-difficile, dans le cas non préemptif, à partir de m=3 [pour m=2, ils ont proposé un algorithme en $O(n)$].

Dans le cas préemptif sans restriction, ils ont développé un algorithme en $O(r^2)$ (r étant le nombre d'opérations), basé sur des calculs successifs de couplages dans un graphe biparti décrivant les relations travaux-machines. Cet algorithme est optimal car il trouve un ordonnancement de durée égale à la borne inférieure B suivante, dans laquelle les deux termes caractérisent respectivement le plus long travail et la machine la plus chargée :

$$B = \text{Max}\left(\max_{i=1,n} \sum_{j=1}^{m} p_{i,j},\ \max_{j=1,m} \sum_{i=1}^{n} p_{i,j} \right).$$

Comme l'ont montré Cho et Sahni, le problème préemptif devient NP-difficile à partir de m=3, lorsque seule la préemption de la machine est autorisée c'est-à-dire si toute tâche d'un travail doit être complètement terminée avant d'en démarrer une autre [CHO 81]. Ces mêmes auteurs ont montré que le problème d'existence d'un ordonnancement admissible en présence de dates de disponibilité ou échues est NP-difficile dans le cas non préemptif, même si m=2. Dans le cas préemptif, ils proposent un algorithme polynomial, assez lourd car basé sur un programme linéaire. Deux cas particuliers sont cependant résolus plus rapidement : en O (n) si m=2, et en O (n^3+m^4) si les dates échues ne prennent que deux valeurs.

La formulation de Cho et Sahni en terme de programme linéaire s'applique aussi à la minimisation du retard maximal, toujours dans le cas préemptif. Lawler *et al.*, en 1981, ont proposé un algorithme de complexité linéaire pour m=2. Ils ont aussi prouvé que minimiser le nombre de travaux en retard est par contre NP-difficile, même si m=2.

La minimisation préemptive de la durée moyenne est NP-difficile pour m non fixé, même si les $p_{i,j}$ ne prennent que deux valeurs 0 et 1 (Gonzalez 82). Liu et Bulfin, en 1985, ont affiné ce résultat au cas m=3, et au cas m=2 avec dates échues.

L'introduction de ressources supplémentaires complique l'open-shop dans le cas non préemptif. Cependant, pour 2 machines, des tâches de durée unitaire, et des ressources renouvelables, on peut minimiser la durée totale en O ($n^{2,5}$) par une méthode de couplage [BLAZ 83].

Le cas préemptif est traitable par la « méthode des deux phases », qui s'applique aussi aux problèmes à machines parallèles [SLOW 78]. Cette méthode peut prendre en compte des contraintes de ressources très variées, minimiser des fonctions économiques incorporant des coûts sur les ressources, etc.

4.1.3.3. Méthodes approchées

La plupart des variantes de l'open-shop étant NP-difficiles, elles relèvent de méthodes approchées dès que les problèmes à résoudre en pratique sont un peu gros. Ces heuristiques appartiennent à deux principaux groupes : les méthodes par construction progressive, et les techniques de couplage.

Dans les premières, un ordonnancement est construit itérativement, en considérant à chaque étape la première machine libre et en lui affectant la tâche disponible de plus grande priorité. Les priorités peuvent être calculées une fois pour toutes, ou ajustées dynamiquement en cours d'algorithme. Ces

méthodes sont très flexibles et peuvent traiter facilement des contraintes de ressources. Elles s'appliquent avec succès aux télécommunications par satellite [PRIN 88], où l'on a pu vérifier *a posteriori* qu'elles donnent en moyenne des résultats à 1 ou 2 % de l'optimum.

Les techniques de couplage de Camerini consistent à construire des « tranches » successives d'ordonnancement. Une tranche consiste en un ensemble de m tâches pouvant s'exécuter simultanément; elle se calcule par recherche d'un couplage max-min ou de poids maximal dans le graphe biparti de participation des machines aux travaux (un arc représentant une tâche). L'ordonnancement final peut souvent être amélioré par tassement ou insertion de tâches dans les « trous » d'inactivité des machines, quand cela ne crée pas de conflits avec d'autres tâches.

4.2. Problèmes cumulatifs : modélisation. Analyse sous contraintes

Dans cette partie, nous faisons les hypothèses traditionnelles sur l'ordonnancement de projet : les tâches sont non préemptives, les ressources sont renouvelables, les modèles sont déterministes et la fonction économique est la durée totale ou le respect des dates échues. Pour des modèles plus généraux ou récents, nous renvoyons le lecteur intéressé aux ouvrages tels [DEME 90], [BLAZ 86], [TALB 78] et [SLOW 89] ainsi qu'aux numéros spéciaux d'*Eur. J. Oper. Res.* liés aux congrès « Project Management and Scheduling » ([TAVA 90] [CARL 92]). Dans les références citées, les auteurs considèrent des modèles déterministes et stochastiques. Pour le cas déterministe, ils relâchent les hypothèses usuelles en considérant différentes catégories de ressources (renouvelables, consommables et doublement limitées), en analysant des modes alternatifs d'exécution des tâches (par exemple, la durée d'une tâche est fonction de la quantité de ressources qui lui est allouée), en introduisant plusieurs critères. Pour le cas stochastique, on trouvera les concepts de base et des modèles traités à l'aide de processus stochastiques.

Les problèmes d'ordonnancement d'atelier présentés précédemment se caractérisent par des contraintes de ressources de type disjonctif particulièrement fortes liées à l'utilisation de ressources renouvelables (machines, opérateurs, ...). On considère ici le cas plus général où chaque tâche nécessite plusieurs unités de certaines ressources renouvelables disponibles en quantité limitée. L'accent est mis sur la modélisation des conflits pour l'utilisation de ces ressources, le cas disjonctif apparaissant comme un cas particulier. L'exploitation de ce modèle, pour la caractérisation des ordonnancements

respectant à la fois les contraintes sur le temps et les ressources, est ensuite présentée.

4.2.1. *Le problème cumulatif général : modélisation des contraintes de ressources*

On suppose d'une part que chaque tâche O_i est caractérisée par sa durée p_i, par l'ensemble K_i des ressources qu'elle utilise et par les quantités q_i^k de ressource k ($k \in K_i$) requises tout au long de sa réalisation et d'autre part que, pour chaque ressource k, Q^k unités sont disponibles à tout instant.

Les contraintes résultant de la limitation de la quantité instantanée de ressource k disponible, sont souvent appelées contraintes cumulatives car elles impliquent à tout instant la limitation de la quantité cumulée de ressource utilisée par les tâches en cours de réalisation. La modélisation de ces contraintes peut s'appuyer utilement sur le concept d'*ensemble critique* de tâches [BELL 82]. Un ensemble critique de tâches I_c est un ensemble minimal de tâches dont la réalisation simultanée nécessite une quantité d'une ressource supérieure à la quantité disponible. Pour qu'un ordonnancement respecte l'ensemble des contraintes cumulatives, il faut et il suffit que deux tâches soient ordonnées au sein de chaque ensemble critique. La résolution du conflit associé à un ensemble critique peut être représentée par un ensemble non conjonctif d'inégalités de potentiels (reliées entre elles par un *OU*) :

$$H(I_c) = \{t_j - t_i \geqq p_i / (i, j) \in I_c \star I_c,\ i \neq j\}$$

Les ensembles $H(I_c)$ représentent l'ensemble des contraintes cumulatives. Cette représentation permet de ramener la prise en compte des ressources à un problème de séquencement modélisé par un graphe potentiels-tâches non conjonctif [ROY 70], ce qui permet de s'appuyer sur certains résultats liés à la résolution du problème central (§ 2.1) [ERSC 79]; une illustration pour trois exemples d'ensembles critiques est donnée sur la figure 15. La construction d'une séquence satisfaisant les contraintes de ressources consiste à choisir une inégalité de potentiels dans chaque ensemble $H(I_c)$, c'est-à-dire à ordonner deux tâches dans chaque ensemble I_c pour résoudre le conflit associé.

Dans le cas général, la recherche des ensembles critiques peut s'avérer complexe (*voir* par exemple [ESQU 87]). Des informations sur les potentiels limites associés à chaque tâche peuvent réduire cette recherche en la limitant aux ensembles critiques de tâches effectivement en conflit dans le temps [ERSC 79]. Dans le cas particulier où q_i^k=1 et où Q^k=m (*voir* 3.2.1), tout ensemble de m+1 tâches utilisant la ressource k constitue un ensemble critique

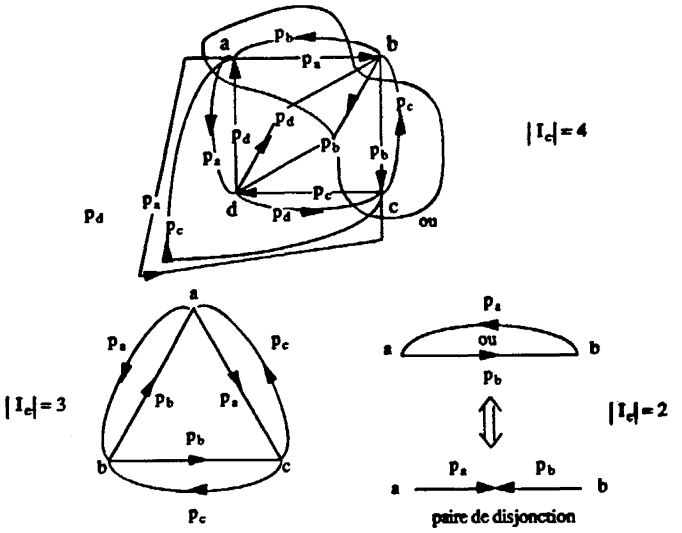


Figure 15. — Exemples d'ensembles critiques et de groupes d'arcs non conjonctifs associés.

de tâches. Ainsi dans le cas où $q_i^k=Q^k=1$, H (I_c) est une paire de disjonction modélisant une contrainte disjonctive (*voir* 4.1.2.1).

On peut noter que dans le cas où il existe des contraintes disjonctives et cumulatives, il est généralement intéressant de traiter en priorité les contraintes disjonctives qui sont les plus structurantes pour l'ordonnancement. Dans tous les cas, le *OU* qui apparaît dans le graphe potentiels-tâches introduit dans le problème un caractère fortement combinatoire qui nécessite, sauf cas particuliers, de faire appel à des méthodes d'énumération implicite ou à des procédures heuristiques.

4.2.2. *Analyse de problèmes d'ordonnancement sous contraintes de temps et de ressources*

4.2.2.1. Présentation de la problématique

Indépendamment de toute fonction économique, il peut être intéressant d'analyser un problème d'ordonnancement défini exclusivement en termes de contraintes. Les contraintes pouvant être prises en compte sont les suivantes : contraintes potentielles entre tâches, contraintes de dates limites (début au plus tôt, fin au plus tard) associées à certaines tâches, contraintes

cumulatives associées aux ressources. Une telle analyse peut conduire à étudier l'existence d'ordonnancements admissibles, à caractériser l'ensemble des ordonnancements admissibles ou encore à générer un ou plusieurs ordonnancements admissibles.

Le problème important est celui de la caractérisation des ordonnancements admissibles. La caractérisation peut en effet aider aussi bien à étudier l'existence de solutions (cohérence des caractéristiques), qu'à générer des solutions admissibles (satisfaction des caractéristiques). La caractérisation des ordonnancements admissibles tend à expliciter les degrés de libertés disponibles pour l'ordonnancement compte tenu des contraintes prises en compte. Notons qu'en l'absence de contraintes de ressources, les notions de marges et de dates de début au plus tôt et au plus tard constituent une caractérisation des ordonnancements admissibles en présence de contraintes de temps alloué (la notion de chemin critique permet de trouver la contrainte maximale admissible).

Les informations issues d'une telle *analyse sous contraintes* peuvent être utilisées comme aide à la décision pour l'ordonnancement dynamique de tâches ou peuvent aider à générer des ordonnancements respectant les contraintes et prenant en compte éventuellement d'autres connaissances sur le problème. Cette approche a donné lieu à un ensemble de travaux ([ERSC 76 *a*, 76 *b*, 79, 80], [ESQU 87], [THOM 80] et [LEGA 89]).

4.2.2.2. Principes généraux de l'analyse sous contraintes

L'analyse sous contraintes peut s'appuyer sur la représentation du problème d'ordonnancement par un graphe potentiels-tâches non conjonctif G^0. Les groupes d'arcs non conjonctifs représentent les conflits associés aux ensembles critiques de tâches (*cf.* 4.2.1) issus des contraintes de ressources. Les conditions d'admissibilité (caractéristiques des ordonnancements admissibles) sont obtenues en interdisant les circuits de longueur strictement positive sur le graphe. Ces circuits peuvent être recherchés en s'appuyant sur les bornes inférieures et supérieures des potentiels obtenues en considérant la partie conjonctive du graphe. L'interdiction de ces circuits se traduit par l'adjonction de nouveaux groupes d'arcs non conjonctifs qui expriment des conditions nécessaires de séquencement entre tâches, résultant de l'interaction entre les contraintes temporelles et les contraintes de ressources. Certains de ces groupes d'arcs (qui peuvent se réduire à un seul arc) permettent d'affiner les bornes des potentiels (dates limites associées aux tâches) qui constituent une caractérisation temporelle des ordonnancements admissibles.

Il est possible d'itérer des transformations du graphe et d'affiner ainsi alternativement la *caractérisation séquentielle et temporelle* des ordonnancements admissibles. On peut montrer [ERSC 79] que cette procédure converge vers un graphe limite G^* dans lequel tout circuit de longueur strictement positive est interdit et qui caractérise donc l'ensemble (supposé non vide) des ordonnancements admissibles. Les bornes des potentiels représentent les caractéristiques temporelles alors que les groupes d'arcs non conjonctifs ainsi que les arcs de la partie conjonctive issus de tels groupes représentent les caractéristiques séquentielles (conditions sur la résolution des conflits d'utilisation des ressources). Ces caractéristiques constituent des conditions nécessaires et suffisantes d'admissibilité.

La recherche de G^* se heurte dans le cas général à la barrière de la complexité, même si, pour des problèmes très contraints, l'interaction temps/séquence permet des résultats spectaculaires [ERSC 76 *b*]. Néanmoins, si on tronque la procédure de caractérisation, on obtient des conditions nécessaires d'admissibilité qui constituent des informations utiles pour l'ordonnancement. Cette procédure d'analyse a donné lieu à des résultats et à des développements spécifiques dans le cas où les contraintes de ressources sont de type disjonctif [ERSC 76 *b*, 80]. Elle a été mise en œuvre sous la forme d'un processus d'inférence utilisant des règles de séquencement et d'actualisation de dates limites permettant d'affiner alternativement les caractéristiques séquentielles et temporelles des ordonnancements admissibles [ESQU 87]. Cette approche est détaillée ci-après.

4.2.2.3. Règles pour l'analyse sous contraintes

Les données du problème sont celles définies en 4.2.1.

Trois types de contraintes sont prises en compte : les contraintes de ressource (chaque ressource k est disponible en quantité limitée Q^k), les contraintes de cohérence technologique représentables par une conjonction d'inégalités de potentiels portant sur les dates de début des tâches et les contraintes de dates limites (les travaux doivent être réalisés dans une fenêtre temporelle définie par une date de début au plus tôt et une date de fin au plus tard). On suppose ainsi que pour toute opération O_i on connaît une borne inférieure de la date de début au plus tôt S_i et une borne supérieure de la date de fin au plus tard F_i. Ces bornes peuvent être obtenues initialement en propageant les dates limites des travaux à travers le graphe conjonctif des contraintes technologiques.

On suppose par ailleurs que l'on connaît les ensembles critiques de tâches (*cf.* 4.2.1). L'analyse sous contraintes du problème peut s'effectuer à travers des règles de séquencement et d'actualisation des dates limites S_i, F_i qui permettent d'affiner la caractérisation des ordonnancements admissibles. Dès qu'une date limite est actualisée, il convient de propager cette actualisation à travers le graphe conjonctif des contraintes de potentiels. Le processus d'analyse s'arrête lorsqu'il n'est plus possible de déduire de nouveaux faits (séquentiels ou temporels) par application d'une règle, ou lorsqu'une incohérence apparaît, mettant ainsi en évidence l'absence d'ordonnancement admissible. Les principales règles sont présentées ci-après.

Règles élémentaires dans le cas disjonctif

Soit $\{O_i, O_j\}$ un ensemble critique d'opérations (utilisant par exemple la même machine). On peut établir la *règle de séquencement* suivante :

Règle 1 : Si $(F_i - S_j) < p_i + p_j$ *Alors* O_i précède O_j.

Réciproquement, si une condition essentielle de précédence telle que « O_i précède O_j » a été établie, on peut chercher à appliquer les règles d'actualisation suivantes :

Règle 2 a : Si O_i précède O_j et $S_i + p_i > S_j$ *Alors* S_j peut être actualisée à la valeur $S_i + p_i$.

Règle 2 b : Si O_i précède O_j et $F_j - p_j < F_i$ *Alors* F_i peut être actualisée à la valeur $F_j - p_j$.

Règles générales dans le cas disjonctif

On peut généraliser les règles précédentes à une clique de disjonctions. Soit Ω un ensemble d'opérations tel que toute paire d'opérations constitue un ensemble critique (clique de disjonctions) et soit O_i une opération de Ω. Les règles de séquencement suivantes généralisent le cas précédent ($\cup$ représente le *OU* logique) :

Règle 3 a : Si $\max_{O_j \in \Omega, O_j \neq O_i} (F_j) - S_i < \sum_{\mu \in \Omega} p_\mu$ *Alors* $\bigcup_{O_j \in \Omega, O_j \neq O_i} (O_j$ précède $O_i)$.

Règle 3 b : Si $F_i - \min_{O_j \in \Omega, O_j \neq O_i} (S_j) < \sum_{\mu \in \Omega} p_\mu$ *Alors* $\bigcup_{O_j \in \Omega, O_j \neq O_i} (O_i$ précède $O_j)$.

Les conditions de séquencement issues de l'application des règles 3 *a* et 3 *b* signifient qu'une opération au moins parmi celles de l'ensemble $\{\Omega - O_i\}$ doit précéder (ou être précédée par) l'opération O_i. Pour simplifier, ces conditions seront notées :

$$\cup\{\Omega - O_i\} \text{ précède } O_i \qquad \text{et} \qquad O_i \text{ précède } \cup\{\Omega - O_i\}.$$

Lorsque de telles conditions sont établies, on peut appliquer les règles d'actualisations suivantes :

Règle 4 a : *Si* $\cup\{\Omega-O_i\}$ précède O_i *et* $\min_{O_j\in\{\Omega-O_i\}} (S_j + p_j) > S_i$

Alors S_i peut être actualisée à la valeur $\min_{O_j\in\{\Omega-O_i\}} (S_j+p_j)$.

Règle 4 b : *Si* O_i précède $\cup\{\Omega-O_i\}$ *et* $\max_{O_j\in\{\Omega-O_i\}} (F_j - p_j) < F_i$

Alors F_i peut être actualisée à la valeur $\max_{O_j\in\{\Omega-O_i\}} (F_j-p_j)$.

Règles dans le cas cumulatif

On considère à présent le cas où deux tâches ne sont pas forcément ordonnées, c'est-à-dire qu'il est possible de réaliser plusieurs tâches simultanément. Soit Γ un ensemble critique d'opérations tel que $|\Gamma|>2$ et soit O_i une opération appartenant à Γ. Les règles de séquencement suivantes peuvent être établies :

Règle 5 a : *Si* $\forall (O_j, O_k) \in \{\Gamma-O_i\}^2$ avec $O_j\neq O_k$, on a $F_k-S_j<p_k+p_j$

et *Si* $\forall O_j \in \{\Gamma-O_i\}$, on a $F_j-S_i<p_j+p_i$ *Alors* $\cup\{\Gamma-O_i\}$ précède O_i.

Règle 5 b : *Si* $\forall (O_j, O_k) \in \{\Gamma-O_i\}^2$ avec $O_j\neq O_k$, on a $F_k-S_j<p_k+p_j$

et *Si* $\forall O_j \in \{\Gamma-O_i\}$, on a $F_i-S_j<p_i+p_j$ *alors* O_i précède $\cup\{\Gamma-O_i\}$.

Lorsque de telles conditions de séquencement ont été obtenues, on peut chercher à appliquer les règles d'actualisation 4 *a* et 4 *b* (cf. *Règles générales*).

Exemple : Soient $\Omega=\{O_1, O_4\}$ et $\Gamma=\{O_2, O_3, O_4\}$ deux ensembles critiques d'opérations dont les caractéristiques sont données dans le tableau ci-dessous.

O_i	S_i	F_i	p_i
O_1	8	15	4
O_2	0	13	6
O_3	3	13	6
O_4	3	13	6

R 1 : $(13-8<6+4)\Rightarrow O_4$ précède O_1.

R 2 *a* : O_4 précède O_1 et $3+6>8\Rightarrow S_1$ est actualisée à 9.

R 2 *b* : O_4 précède O_1 et 15–4<13⇒F_4 est actualisée à 11.

R 5 *b* : (13–3<6+6 et 13–3<6+6) et (13–3<6+6 et 13–3<6+6)⇒O_2 précède (O_3 ou O_4).

R 4 *b* : O_2 précède (O_3 ou O_4) et max (13–6, 13–6)<13⇒F_2 est actualisée à 7.

Ces actualisations étant enregistrées, on peut alors déclencher la règle 5 *a* :

R 5 *a* : (11–0<6+6 et 7–3<6+6) et (11–3<6+6 et 7–3<6+6)⇒(O_2 ou O_4) précède O_3.

R 4 *a* : (O_2 ou O_4) précède O_3 et min (0+6, 3+6)>3 alors S_3 est actualisée à 6.

Le tableau suivant donne les caractéristiques des tâches après actualisations :

O_i	S_i	F_i	p_i
O_1	9	15	4
O_2	0	7	6
O_3	6	13	6
O_4	3	11	6

Exceptées ces informations numériques, l'analyse déduit également des caractéristiques séquentielles entre les tâches : O_4 précède O_1; de plus par traitement logique de « O_2 précède (O_3 ou O_4) » et « (O_2 ou O_4) précède O_3 », on obtient la contrainte de succession simple : O_2 précède O_3, nouvelle relation qui n'entraîne toutefois pas d'actualisation supplémentaire.

4.2.2.4. Autres développements

Le processus d'analyse sous contraintes peut être étendu et enrichi en utilisant le concept d'énergie qui lie la consommation du temps et des ressources [ERSC 91]. Une autre approximation de l'analyse sous contraintes peut consister à rechercher des conditions suffisantes d'admissibilité en caractérisant complètement un sous-ensemble particulier d'ordonnancements admissibles ([THOM 80], [LEGA 89]). Le choix de l'approximation (conditions suffisantes ou conditions nécessaires présentées en 4.2.2.3) peut dépendre du taux de contraintes du problème.

Enfin, un concept de *dominance* vis-à-vis de l'admissibilité peut être défini : un ordonnancement en domine un autre si son admissibilité implique celle de l'autre, pour un corps d'hypothèses donné. Pour certains problèmes à contraintes disjonctives et pour un corps d'hypothèses relativement faible

(ordre relatif des dates limites connu), une caractérisation intéressante des ordonnancements dominants a pu être établie ([ERSC 83], [COUZ 79] et [FONT 80]). Lorsqu'il existe des ordonnancements admissibles, les ordonnancements dominants sont ceux qui satisfont le plus largement les contraintes (flexibilité maximale). Les caractéristiques des ordonnancements dominants peuvent être également utilisées pour améliorer les performances de procédures arborescentes d'optimisation vis-à-vis de certains critères ([FONT 80]).

5. PROBLÈMES D'ORDONNANCEMENT CYCLIQUES

Les problèmes d'ordonnancement cycliques ont de nombreuses applications tant informatiques (calculs cycliques sur architectures parallèles) que dans le domaine de la production (job-shops cycliques) ou de la planification (affectation périodique de ressources). Jusqu'à maintenant, la plupart des travaux qui leur sont consacrés sont très proches de leurs applications respectives et concernent surtout des heuristiques permettant une résolution approchée rapide. Cependant, de plus en plus d'auteurs s'intéressent à l'étude théorique des problèmes sous-jacents, avec toutefois un partage regrettable des chercheurs selon le domaine d'application d'origine de leur problème. Nous tentons, dans cet article, un début de synthèse qui demande à être poursuivie à l'avenir. Nous présentons tout d'abord un exemple montrant une application informatique de ces problèmes. Puis nous évoquons les principaux modèles généraux et leurs propriétés. Enfin, nous présentons des résultats récents sur les problèmes avec ressources issus du problème central cyclique.

5.1. Un exemple de problème cyclique avec ressources

Nous présentons comme exemple une application informatique des problèmes cycliques : la microprogrammation de boucles vectorielles sur des super-calculateurs. Les architectures correspondantes comportent des modules spécialisés pour le calcul vectoriel, qui sont des pipelines microprogrammables. Le module vectoriel du CRAY2, par exemple, peut être vu comme l'indique la figure 16.

Il comporte quatre opérateurs spécialisés parallèles qui communiquent *via* un banc de huit registres : un additionneur, un multiplieur, une unité de lecture et une unité d'écriture sur un banc mémoire (BM). Chaque opérateur est caractérisé par sa durée (d) et sa latence (l) qui mesure l'intervalle de temps minimum entre deux activations successives. Le contrôle est assuré au

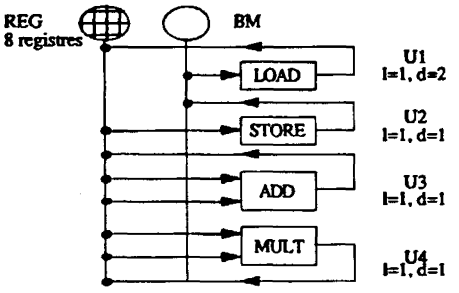


Figure 16.

moyen d'un microprogramme dont chaque instruction gère le fonctionnement de la machine durant une unité de temps.

Sur une telle architecture, on cherche à effectuer la boucle vectorielle suivante :

Pour I=1 à N faire
$A(I+2)=(A(I)+B(I)) \star B(I)$
fin faire

Le calcul d'une itération i est décomposé en 5 tâches :

Tâche $T1$, effectuée par $U1$: lire $A(i)$;

Tâche $T2$, effectuée par $U1$: lire $B(i)$;

Tâche $T3$, effectuée par $U3$: additionner $A(i)$ et $B(i)$;

Tâche $T4$, effectuée par $U4$: multiplier le résultat de $T3$ par $B(i)$;

Tâche $T5$, effectuée par $U2$: écrire le résultat de $T4$ sur $A(i+2)$.

La sémantique du calcul induit des contraintes qui sont celles d'un problème central cyclique, et qui sont représentées dans le graphe potentiel généralisé de la figure 17. On rappelle que chaque arc porte deux valuations (l, h) : l représente la durée de la tâche origine de l'arc, et h la hauteur de la contrainte correspondante.

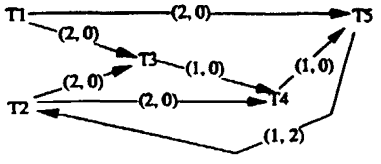


Figure 17.

Lorsque le nombre d'itérations de la boucle est grand, on le suppose infini et on cherche à ordonnancer les occurrences des tâches génériques de façon à maximiser le débit de la machine, c'est-à-dire le nombre moyen d'itérations calculées par unité de temps. Il faut alors veiller à ce que les opérateurs soient utilisés sans conflits, et déterminer une affectation périodique des huit registres de la machine pour stocker les résultats intermédiaires des occurrences de tâches.

5.2. Modèles et résultats généraux

Quatre modèles généraux, qui induisent des techniques de résolution particulières ont été proposés pour les problèmes cycliques.

Réseaux de Petri temporisés

Le premier d'entre eux est le modèle des réseaux de Petri temporisés (*cf.* § 1.1) [CARL 89 *a*, 89 *c*]. Chaque transition modélise une tâche générique; les contraintes de précédence internes, externes et les contraintes de ressource sont représentées par les places et les arcs qui les relient aux transitions. La figure 18 représente le réseau de Petri associé à l'exemple précédent. Lorsque le réseau de Petri est un graphe d'événements, le problème d'ordonnancement associé est un énoncé du problème central cyclique (*cf.* 2.3).

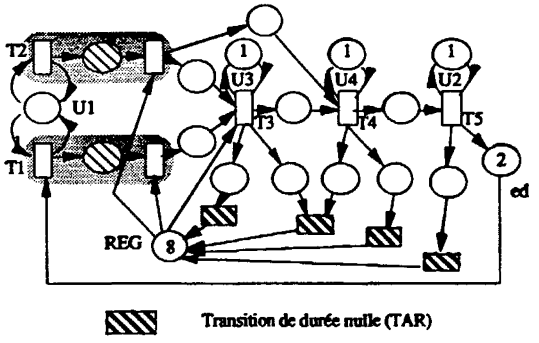


Figure 18.

Carlier et Chretienne ont consacré une étude générale aux ordonnancements infinis d'un réseau de Petri temporisé quelconque. Ils ont montré que lorsqu'une séquence périodique de franchissement de transitions est fixée, il existe un ordonnancement au plus tôt qui est K-périodique et se calcule polynomialement. De plus ces ordonnancements sont dominants. Cela permet la construction d'un graphe d'état, appelé graphe au plus tôt du réseau, dont les chemins correspondent aux ordonnancements au plus tôt des séquences de

franchissement. Le problème de la maximisation du débit est alors exprimé comme un programme linéaire dont les contraintes sont construites à partir de l'ensemble des circuits élémentaires du graphe au plus tôt.

Ce modèle est très général et permet de représenter un nombre important de contraintes dans un formalisme unifié, ce qui s'avère être très important pour effectuer des simulations ou pour piloter un atelier de production. Toutefois, sa généralité même est une faiblesse pour ce qui concerne la recherche de techniques efficaces de résolution. Dans la lignée des travaux mentionnés ci-dessus, les recherches menées actuellement, que nous détaillons plus loin, consistent essentiellement à ajouter des contraintes de ressources particulières à un énoncé de problème central cyclique. On utilise ensuite les propriétés des ordonnancements basées sur l'étude des graphes bivalués pour déterminer de bons algorithmes.

Dioïdes

Une étude algébrique poussée du dioïde (R^+, max, +) a permis une nouvelle approche de la résolution du problème central cyclique [COHE 85]. A titre de comparaison, la période d'exécution des tâches, qui est associée aux circuits critiques du graphe d'événement, correspond dans ce modèle aux valeurs propres de certaines matrices. Les dioïdes ont l'avantage d'offrir un cadre rigoureux général pour l'étude des problèmes cycliques. Des travaux récents de Gaubert ont permis de caractériser les solutions du problème de minimisation du nombre de jetons nécessaires dans certaines places d'un graphe d'événements pour réaliser un débit donné, ce qui s'apparente au problème de minimisation des encours dans un atelier de production. Cependant, ce modèle semble jusqu'à présent difficile à utiliser en présence de contraintes de ressource disjonctives.

Modèle à distances cycliques limitées

Plus récemment, deux auteurs [SERA 89] ont proposé un modèle général pour des problèmes cycliques un peu différents de ceux induits par les modèles précédents. Il s'agit, à partir d'un ensemble de tâches génériques, de déterminer un ordonnancement l-périodique : les suites des dates d'exécution d'une même tâche forment une progression arithmétique de période T donnée. De ce fait dans toute période de temps de longueur T, une et une seule instance de chaque tâche générique est calculée. Les contraintes considérées sont des contraintes de localisation dans le temps et des contraintes de ressources très générales. Les contraintes de localisation ont une forme particulière : pour deux tâches i et j, on impose que, dans toute période de longueur T, la différence entre les dates de début des instances des tâches i

et j appartiennent, modulo T, à un intervalle de temps donné. Il faut remarquer qu'une telle contrainte porte sur des instances de tâches indépendamment des itérations auxquelles elles appartiennent. Ce ne sont donc pas des contraintes de précédence au même titre que celles du problème central cyclique.

Les auteurs, après avoir établi la NP-difficulté du problème comportant uniquement des contraintes de localisation temporelle, proposent un algorithme fondé sur une recherche exhaustive à l'aide d'une arborescence.

Le lien de ce modèle avec celui des réseaux de Petri temporisés n'est pas immédiat. Cependant, il semble que les contraintes du problème central cyclique se représentent comme des contraintes de ressource dans ce modèle, alors que certaines contraintes de ressource représentées dans un réseau de Petri temporisé, comme par exemple les contraintes disjonctives, peuvent être modélisées par des contraintes de localisation temporelle.

Equations de récurrence

Le problème central cyclique peut être vu comme celui de la parallélisation d'un système d'équations de récurrence uniforme dont le domaine de variation des indices est de dimension 1. Or les systèmes d'équations de récurrence en dimension n quelconque ont fait l'objet d'un certain nombre de travaux relatifs notamment à la synthèse d'architectures systoliques destinées à résoudre un système particulier. Rappelons qu'un tel système est caractérisé par des variables $V_1(p), \ldots, V_k(p)$ où p appartient à un domaine fini D de Z^n. Pour tout n-uplet p de D, les variables sont liées par des équations de la forme :

$V_i(p)=f\,(V_1(g_1(p)), \ldots, V_k(g_k(p)))$ où les g_i sont des fonctions de D dans D.

Une telle équation est dite équation de récurrence uniforme lorsque les fonctions g_i sont des translations simples dans D : $g_i(p)=p-r_i$, où r_i est un vecteur. On l'appelle équation de récurrence linéaire lorsque les fonctions g_i sont affines : $g_i(p)=a_ip-b_i$ où a_i est une constante entière et b_i un vecteur.

La base de ces travaux est un article fondamental [KARP 67] qui détermine un ordonnancement linéaire par morceaux qui majore l'ordonnancement au plus tôt pour une équation. Cet ordonnancement est obtenu en déterminant les sommets d'un polyèdre en dimension n. Cet article établit de plus des conditions de cohérence d'un système d'équations de récurrence uniforme.

Par la suite, les auteurs se sont intéressés à la caractérisation des ordonnancements linéaires ou quasi-linéaires pour des systèmes d'équations de récurrence uniforme et de récurrence linéaire [QUIN 89]. Là aussi, des conditions nécessaires et suffisantes d'existence ont été établies. D'autre part,

en dimension quelconque, Saouter a établi que le problème de la cohérence d'un système d'équations avec des récurrences linéaires est un problème indécidable [SAOU 90].

En dimension 1, en généralisant les travaux sur les graphes d'événements, Munier a caractérisé le comportement asymptotique de l'ordonnancement au plus tôt pour un système d'équations de récurrences linéaires. La K-périodicité de cet ordonnancement est ainsi établie, et un algorithme de calcul des fréquences optimales basé sur une décomposition du graphe en composantes dites unitaires et une « expansion » de ces composantes est proposé [MUNI 91 *b*].

5.3. Problèmes avec gamme unitaire

Un petit nombre d'études ont porté sur les problèmes cycliques à gamme unitaire. Il s'agit, considérant un problème central cyclique, de planifier les tâches sur des machines identiques.

Cytron, partant d'une application informatique, s'est intéressé au problème avec une infinité de processeurs mais en imposant que toutes les tâches d'une même itération soient effectuées par la même machine [CYTR 84]. Il s'agit alors de déterminer un ordre total sur les tâches génériques et un décalage constant minimal entre les itérations de façon à respecter les contraintes de précédence externes. Il a établi la NP-difficulté du problème, et déterminé des heuristiques inspirées par l'algorithme de *HU*, fondées sur une décomposition par niveau du graphe des contraintes. Ce problème a récemment été repris par Munshi et Simons [MUNS 90] qui ont généralisé ce type d'heuristiques lorsque le nombre de processeurs est limité.

Concernant le problème à m processeurs, Aiken et Nicolau ont proposé des heuristiques dont le principe est le suivant : on déroule les itérations, c'est-à-dire que les tâches génériques sont dupliquées un certain nombre de fois et l'on détermine une liste de priorités basée sur l'exécution au plus tôt sans contraintes de ressources. L'ordonnancement sur m machines de ces tâches dupliquées est alors obtenu à l'aide d'un algorithme de liste. Le nombre de duplications doit être suffisamment important pour atteindre un régime périodique, avec au besoin quelques modifications locales [AIKE 88]. Il est donc difficile de déterminer si cet algorithme est polynomial, dans la mesure où même lorsque le nombre de processeurs est infini, le problème posé est celui (non résolu) de la durée du régime transitoire de l'ordonnancement au plus tôt. Sur ce problème, on peut citer GAO qui a établi que cette durée est polynomiale dans un cas très particulier de graphe d'événement (dont tous les circuits élémentaires sont de hauteur 1) [GAO 91].

Munier a montré la NP-difficulté du problème, et établi un certain nombre de sous-cas polynomiaux selon la structure du graphe des contraintes de précédence. Elle a déterminé en particulier des algorithmes efficaces lorsque ce graphe est réduit à un circuit, ou encore lorsqu'il n'y a pas de contraintes externes [MUNI 91 *a*]. Dans la lignée de ces travaux, Hanen et Munier ont étudié la structure de l'ensemble des solutions du problème général. Elles ont montré la dominance des ordonnancements pour lesquels les tâches sont affectées circulairement aux machines. Des heuristiques fondées sur la résolution de plusieurs instances du problème central cyclique ont été proposées [HANE 92].

Enfin, Gasperoni et Schwiegelshohn [GASP 91] ont montré que l'on pouvait associer à un problème cyclique sur m machines un problème non cyclique dont la solution fournit un ordonnancement réalisable du problème cyclique. En utilisant un algorithme de liste sur ce problème, on peut donc construire une solution approchée en temps polynomial O (n^3 logn). De plus, grâce à la borne connue sur les performances des algorithmes de liste pour les problèmes d'ordonnancement à m machines, la distance relative de cette solution à l'optimum a pu être évaluée dans le pire des cas.

On peut remarquer que jusqu'à présent il n'a pas été proposé de méthode exacte de résolution pour ce problème.

5.4. Problèmes avec gamme générale

Les problèmes cycliques à gamme générale ont été étudiés indépendamment selon deux principales directions : les ordonnancements de pipelines et les versions cycliques de problèmes d'atelier (flow-shop et job-shop). Cependant les problèmes sous-jacents sont très voisins et mériteraient d'être abordés en commun.

5.4.1. *Problèmes de pipelines*

Les pipelines sont des machines composées de processeurs parallèles spécialisés appelés unités fonctionnelles qui échangent des données au travers de registres plus ou moins partagés. Leur contrôle est assuré au moyen d'un microprogramme, dont chaque instruction gère l'activation des unités fonctionnelles ainsi que les chemins de données pendant une unité de temps. Le module vectoriel du CRAY2 présenté plus haut en est un exemple. L'utilisation de tels pipelines pour exécuter des instructions de haut niveau pose des problèmes d'ordonnancement complexes. Le cas des calculs cycliques (boucles vectorielles avec ou sans récurrences) mérite une

attention particulière puisqu'ils forment environ 80 % du temps de calcul des programmes scientifiques.

Comme dans notre exemple, le calcul d'une itération de la boucle est associé à un ensemble de tâches génériques, exécutables par les processeurs. Du fait de leur spécialisation, l'allocation des tâches aux processeurs est fixée. Aux transferts de données entre les tâches sont associées des contraintes de précédence de même nature que celles du problème central cyclique. Cependant, la présence d'un nombre limité de registres pour assurer ces transferts de données induit des contraintes de ressource qui sont rarement prises en compte dans les approches classiques.

Tables de réservation

Le problème de la maximisation du débit de pipelines a tout d'abord été traité [KOGG 81] pour les pipelines classiques et les boucles vectorielles non récurrentes (à itérations indépendantes), c'est-à-dire sans contraintes externes dans le problème central cyclique sous-jacent. Une table de réservation, déterminée au moyen d'heuristiques, représente l'allocation des ressources, unité par unité, pendant la durée de l'ordonnancement générique. La recherche de circuits dans un graphe d'état fournit un ordonnancement périodique, optimal pour cette table. Cette approche ne permet notamment pas de représenter le partage de ressources de même type, comme les registres d'un même banc. Les solutions obtenues peuvent en conséquence être éloignées de l'optimum. Ce type de technique a été ensuite généralisé pour tenir compte de dépendances entre les itérations et de contraintes de ressource cumulatives plus complexes [HANE 90].

Il est à noter que la complexité de ce problème est toujours inconnue. Certains auteurs ont cherché à déterminer un ordonnancement générique optimal respectant un cycle de lancement donné. Les algorithmes d'insertion de délai [PATE 76] fournissent une solution exacte en un temps polynomial dans un cadre très restreint : les tâches sont de durée unitaire et ni les registres ni les précédences externes ne sont pris en compte.

Etude générale

Par ailleurs, il a été montré que les ordonnancements cycliques fondés sur un ordonnancement générique fixé ne sont pas nécessairement dominants lorsque toutes les contraintes sont prises en compte. L'étude du problème global a été effectuée grâce à une modélisation en termes de réseaux de Petri temporisés. Le principal résultat en est un algorithme exact (non polynomial) fournissant un ordonnancement K-périodique optimal, fondé sur la recherche de circuits critiques dans un graphe d'état [HANE 89].

Ordonnancements périodiques

Beaucoup d'auteurs se sont intéressés aux ordonnancements périodiques, constitués d'un d'ordonnancement générique répété à des intervalles de temps constants. La plupart proposent des heuristiques [LAM 87], [EISE 88] qui s'inspirent des techniques d'insertion de délai citées plus haut : pour une valeur du débit fixée, de la forme $1/\alpha$ avec α entier, on utilise une liste de priorités pour construire un ordonnancement des tâches génériques de telle sorte que la répétition de cet ordonnancement toutes les α unités de temps n'engendre pas de conflit. La liste de priorité est déterminée par la position relative des tâches dans le corps de boucle, et correspond en général à une date au plus tôt dans un sous-problème sans contraintes de ressource. Si l'on échoue on recommence avec un débit $1/(\alpha+1)$.

Récemment il a été proposé une étude du problème [HANE 91] qui établit sa NP-difficulté même lorsque le graphe est réduit à un circuit. Cet article montre qu'un arbitrage des disjonctions consiste dans ce cas à définir les hauteurs d'arcs de disjonction sur le graphe des contraintes. Il est donc associé à une instance d'un problème central cyclique. Des techniques permettant d'obtenir des bornes de ces hauteurs ainsi qu'une première méthode exacte par séparation et évaluation ont été proposées. Une nouvelle heuristique permet d'initialiser la méthode arborescente.

Cependant dans toutes ces techniques les registres sont toujours supposés en nombre suffisant, ce qui peut amener à des solutions non réalisables.

5.4.2. *Problèmes d'atelier*

Dans un domaine d'application différent, les problèmes d'atelier, certains auteurs se sont intéressés à des problèmes de flow-shop cycliques. Matsuo a traité des problèmes à deux machines sans attente en se ramenant pour l'un d'entre eux à un sous-problème polynomial du voyageur de commerce [MATS 88].

Roundy s'est penché sur le problème d'un seul job cyclique en cherchant non seulement à maximiser le débit, mais aussi à minimiser la durée d'une occurrence du job, ce qui correspond, en termes informatiques, à la durée d'une itération [ROUN 88]. Il a établi la NP-difficulté de son problème et a étudié la structure des solutions réalisables. Il a montré que certaines solutions pouvaient être regroupées en un nœud d'un treillis, chaque nœud étant associé à un graphe partiel du graphe des contraintes. Il en déduit un algorithme exact, qui consiste à parcourir le treillis au moyen de transformations locales du

graphe. L'idée de base de ces transformations est d'échanger deux tâches effectuées successivement (et de manière périodique) sur la même machine.

On peut aussi mentionner les problèmes de traitement de surface. On considère une ligne composée de cuves contenant des produits chimiques différents. Au dessus de la ligne se trouve un rail sur lequel se déplacent un ou plusieurs robots. Une gamme consiste ici à plonger un objet grâce au(x) robot(s) dans une suite donnée de bains. Les temps d'égouttage et de déplacement du robot sont donnés. On donne des fourchettes pour les temps de trempage. Il s'agit alors de produire de grandes séries de produits selon la même gamme. Il s'agit d'un problème NP-difficile. Les techniques de résolution pour la recherche d'un ordonnancement périodique d'un robot sont principalement basées sur la programmation linéaire mixte. Toutefois, Lei a utilisé un autre modèle qui lui permet de proposer une méthode arborescente basée sur l'évaluation de fenêtres de temps pour les dates d'activation du robot [LEI 89]. Cette méthode peut être étendue pour la recherche d'ordonnancements K-périodiques avec K fixé. Elle a aussi proposé une heuristique pour le cas de plusieurs robots lorsqu'une zone d'influence est définie pour chacun d'entre eux.

5.5. Perspectives

Ce tour d'horizon des problèmes cycliques montre que leurs applications pratiques sont variées. La complexité de ces problèmes reste en partie ouverte. S'il est vraisemblable que la plupart des problèmes cycliques comportant des contraintes de ressource sont NP-difficiles, on peut noter que contrairement aux problèmes classiques ce n'est pas toujours le cas.

La plupart des approches qui ont été proposées pour résoudre ces problèmes sont, soit de nature heuristique, soit utilisent des recherches dans des graphes d'état dont le nombre de sommets ne permet pas de traiter de grands problèmes. Il est certain qu'à l'avenir, des approches énumératives efficaces du type méthode arborescente ou programmation dynamique devront être définies. D'autre part une étude de l'efficacité numérique et dans le pire des cas des heuristiques proposées reste à faire.

De nouveaux résultats devraient pouvoir naître de la confrontation et la synthèse des différents domaines d'application jusqu'à présent très cloisonnés.

CONCLUSION

Toutes les activités humaines amènent à planifier des tâches en leur allouant dans le temps des ressources. Ceci explique que depuis 1954 et les travaux de Jackson, de Johnson et Smith, les publications sur les problèmes d'ordonnancement foisonnent dans des revues très variées. Par exemple, dans cet article, nous avons considéré des problèmes d'ateliers se posant dans l'industrie, des problèmes d'ordonnancement de paquets liés aux télécommunications par satellite, des problèmes d'allocation de processeurs rencontrés en informatique. Mais on pourrait multiplier les domaines en considérant les emplois du temps d'une administration, la planification des tâches d'une usine chimique ou alimentaire, les roulements d'agents dans les transports...

Pourtant, à la lecture de la littérature et malgré la diversité des applications, on reconnaît des problèmes de même nature. Ce qui les caractérise, c'est leur aspect combinatoire. On peut les aborder avec une méthodologie générale, mais il faut les traiter par des algorithmes particuliers nécessitant une expertise de leur structure. Dans certains cas, on ne peut guère espérer mieux qu'une aide interactive à leur traitement si les contraintes sont trop nombreuses et mal formulées (c'est souvent le cas des emplois du temps). Mais, fréquemment, il faudra mettre au point des heuristiques ou des méthodes exactes et cela nécessite de connaître la théorie classique de l'ordonnancement, c'est pourquoi nous avons rappelé la méthode PERT, le problème central cyclique et les principaux résultats concernant les problèmes à une ressource et à plusieurs ressources.

Dans ce survey, nous avons également mis l'accent sur des modèles plus récents comme les réseaux de Petri temporisés utiles pour les simulations et des problèmes nouveaux comme les problèmes cycliques ou distribués. C'était sans doute arbitraire et lié à nos préoccupations personnelles dues à nos passés de chercheurs. En effet, il y a de nombreuses directions nouvelles concernant des modèles ou techniques utiles qui auraient mérité un développement spécifique. Puisqu'il fallait choisir, nous renvoyons le lecteur intéressé à la bibliographie. Citons d'autres modèles de tâches [SLOW 89] ou de ressources [BLAZ 86], le PERT-coût [ELMA 92], [FOLD 92], les problèmes multicritères [HHOO 92], la complexité des problèmes à une ressource [BLAZ 83], la relaxation lagrangienne [VELD 91], les algorithmes d'analyse numérique sur les machines parallèles [QUIN 89], le stochastique [DEMP 82]...

Cette discussion montre s'il en était besoin la vivacité du domaine mais n'oublions pas que le sel des problèmes d'ordonnancement, c'est la perspective d'un enrichissement permanent de la recherche par les applications nouvelles.

Présentation du groupe GOTHA

Jacques CARLIER	Professeur à l'Université de Technologie de Compiègne, Laboratoire HEUDIASYC (C.N.R.S.)
Philippe CHRETIENNE	Professeur à l'Université P.-et-M.-Curie, Institut Blaise Pascal, Laboratoire L.I.T.P. (C.N.R.S.)
Jacques ERSCHLER	Professeur à l'I.N.S.A. de Toulouse, Laboratoire L.A.A.S. (C.N.R.S.)
Claire HANEN	Maître de Conférences à l'Université P.-et-M. Curie, Institut Blaise Pascal, Laboratoire L.I.T.P. (C.N.R.S.)
Pierre LOPEZ	Attaché Temporaire d'Enseignement et de Recherche à l'E.N.S.E.E.I.H.T.-I.N.P. de Toulouse, Laboratoire L.A.A.S. (C.N.R.S.)
Alix MUNIER	Attaché Temporaire d'Enseignement et de Recherche à l'Université de Paris-XI - Orsay. Laboratoire L.R.I. (C.N.R.S.)
Eric PINSON	Maître de Conférences à l'Institut de Mathématique Appliquée d'Angers, Laboratoire C.R.E.A.M.
Marie-Claude PORTMANN	Professeur à l'Ecole des Mines de Nancy, Projet S.A.G.E.P. (I.N.R.I.A.-Lorraine)
Christian PRINS	Maître de Conférences à l'Institut de Mathématique Appliquée d'Angers, Laboratoire C.R.E.A.M.
Christian PROUST	Maître de Conférences à l'Université de Tours, Laboratoire d'Informatique, EIII.
Pierre VILLON	Enseignant-Chercheur à l'Université de Technologie de Compiègne, Laboratoire HEUDIASYC (C.N.R.S.)

BIBLIOGRAPHIE

[ACHU 82] J. O. Achugbue et F. Y. Chin, Complexity and Solution of Some Three-Stage Flow-Shop Scheduling Problem, *Math. Opns. Res.*, 1982, *7*, p. 532-544.

[ADAM 88] J. Adams, E. Balas et D. Zawack, The Shifting Bottleneck Procedure for Job Shop Scheduling, *Management Sci.*, 1988, *34*, p. 391-401.

[AIKE 88] A. Aiken et A. Nicolau, Optimal Loop Parallelization, *Proc. of the SIGPLAN'88 Conf. on Prog. Language Design an Implementation*, Atlanta, U.S.A., june 1988, p. 308-317.

[ANGE 87] B. Angeniol, G. De La Croix Vaubois et J.-Y. Le Texier, Self Organizing Features Maps on the Travelling Salesman Problem, *Rapport Interne Thomson C.S.F./D.S.E.*, 1987.

[AXSA 84] S. Axsater et H. Jonsson, Aggregation and Dissaggregation in Hierarchical Production Planning, *EJOR*, 1984, *17*, p. 338-350.

[BAKE 74] K. R. Baker, *Introduction to Sequencing and Scheduling*, John Wiley, 1974.

[BAKE 90] K. R. Baker, Scheduling Groups of Jobs in the Two-Machine Flow Shop, *Math. Comput. Modelling*, 1990, *13*, 3, p. 29-36.

[BELL 82] R. Bellman, A. O. Esogbue et I. Nabeshima, *Mathematical Aspects of Scheduling and Applications*, Pergamon Press, Oxford, England, 1982.

[BIEG 90] J. E. Biegel et J. J. Davern, Genetic Algorithms and Job Shop Scheduling, *Comput. ind. Engng.*, 1990, *19*, 1/4, p. 81-91.

[BITR 82] G. R. Bitran, E. A. Haas et A. C. Hax, Hierarchical Production Planning: a Two Stage System, *Opns. Res.*, 1982, *30*, p. 232-251.

[BRUC 91] P. Brucker, B. Jurish et B. Sievers, A Branch and Bound Algorithm for the Job Shop Scheduling Problem, *Disc. Appl. Math.* (à paraître).

[BLAZ 83] J. Blazewicz, J. K. Lenstra et A. H. G. Rinnooy Kan, Scheduling Subject to Resource Constraints: Classification and Complexity, *Disc. Appl. Math.*, 1983, *5*, p. 11-24.

[BLAZ 86] J. Blazewicz, W. Cellary, R. Slowinski et J. Weglarz, Scheduling Under Resource Constraints: Deterministic Models, *Annals of Opns. Res.*, J. C. Baltzer AG, Basel, 1986.

[CARL 75] J. Carlier, Disjonctions dans les ordonnancements, *RAIRO-Oper. Res.*, juin 1975, *2*, p. 83-100.

[CARL 78] J. Carlier, Ordonnancements à Contraintes Disjonctives, *RAIRO-Oper. Res.*, novembre 1978, *12*, p. 333-351.

[CARL 82 *a*] J. Carlier et A. H. G. Rinnooy Kan, Financing and Scheduling, *Opns. Res. Letters*, avril 1982, *1*, 2, p. 52-55.

[CARL 82 *b*] J. Carlier et P. Chretienne, Les Problèmes d'ordonnancement : un domaine très ouvert, *RAIRO-Oper. Res.*, août 1982, p. 175-217.

[CARL 82 *c*] J. Carlier, The One Machine Problem, *EJOR*, 1982, p. 42-47.

[CARL 84] J. Carlier, P. Chretienne et C. Girault, Modelling Scheduling Problems with Timed Petri Nets, Advances Studies in Petri Nets, *Lecture Notes in Comput. Sci.*, Springer Verlag, september 1984.

[CARL 88 *a*] J. Carlier et P. Chretienne, *Les problèmes d'ordonnancements : modélisation, complexité, algorithmes*, Masson, Paris, février 1988.

[CARL 88 *b*] J. Carlier et C. Prins, Optimisation des plans de trame dans le système AMRT/CNC d'EUTELSAT, *Annales des Télécomm.*, 1988, *43*, 9-10, p. 506-521.

[CARL 89 *a*] J. Carlier et P. Chretienne, Timed Petri Nets Schedules, Advances in Petri Nets 1988, *Lecture Notes in Comput. Sci.*, Springer Verlag, january 1989, p. 62-84.

[CARL 89 *b*] J. Carlier et E. Pinson, A Branch and Bound Method for Solving the Job Shop Problem, *Management Sci.*, february 1989, *35*, 2, p. 164-176.

[CARL 89 *c*] J. Carlier, Scheduling under Financial Constraints, Advances in Project Scheduling (R. Slowinsky et J. Weglarz eds.), Elsevier science, april 1989, p. 187-224.

[CARL 90] J. Carlier et E. Pinson, The Use of the Jackson Preemptive Schedule for Solving the Job Shop Problem, *Annals of Opns. Res.*, 1990 (à paraître).

[CARL 92] J. Carlier et L. Tavares éd., Project Management and Scheduling 2, *EJOR*, Special Issue (à paraître).

[CHO 81] Y. Cho et S. Sahni, Preemptive Scheduling of Independent Jobs with Release and Due Dates on Open, Flow and Job Shops, *Opns Res.*, 1981, *29*, p. 511-522.

[CHRE 83] P. Chretienne, Les réseaux de Petri temporisés, *Thèse d'état*, Université Paris-VI, 1983.

[CHRE 85] P. Chretienne, Analyse des régimes transitoire et permanent d'un graphe d'événements temporisé, *TSI*, 1985, *4*, 1, p. 127-142.

[CHRE 89] P. Chretienne, A Polynomial Algorithm to Optimally Schedule Tasks on a Virtual Distributed System Under Tree-Like Precedence Constraints, *EJOR*, 1989, *43*, p. 225-230.

[CHRE 90 *a*] P. Chretienne, Task Scheduling with Interprocessor Communication Delays, *EJOR*, 1992, *57*, p. 348-354.

[CHRE 90 *b*] P. Chretienne, Complexity of Tree Scheduling with Interprocessor Communication Delays, *Rapport M.A.S.I. n° 90.5*, février 1990.

[CHRE 91] P. Chretienne, The Basic Cyclic Scheduling Problem with Deadlines, *Disc. Appl. Math.*, 1991, *30*, p. 109-123

[CHRY 91] G. Chryssolouris, K. Dicke et M. Lee, An Approach to Short Interval Scheduling for Discrete Parts Manufacturing, *Int. J. Comput. Integrated Manufacturing*, 1991, *4*, 3, p. 157-168.

[CHU 89] C. Chu et M. C. Portmann, Minimisation de la somme des retards pour les problèmes d'ordonnancement à une machine, *Rapport de Recherche INRIA n° 1023*, avril 1989.

[CHU 90] C. Chu, Nouvelles approches analytiques et concept de mémoire artificielle pour divers problèmes d'ordonnancement, *Thèse d'Université*, Université de Metz, septembre 1990.

[CHU 92 *a*] C. Chu, M. C. Portmann et J. M. P. Proth, A Splitting-up Approach to Simplify Job-Shop Scheduling Problems, *Int. J. Prod. Res.*, 1992, *30*, 4, p. 859-870.

[CHU 92 *b*] C. Chu et M. C. Portmann, Some New Efficient Methods to Solve the $n/1/r_i/\sum T_i$ Scheduling Problems, *EJOR* (à paraître).

[COFF 76] E. G. Coffman Jr éd., *Computer and Job-Shop Scheduling*, J. Wiley & Son, New York, 299 p., 1976.

[COHE 85] G. COHEN, D. DUBOIS, J. P. QUADRAT et M. VIOT, A Linear System Theoretic View of Discrete Event Processes and its Use for Performance Evaluation in Manufacturing, *IEEE. Trans. on Automatic Control*, *30*, 3, p. 210-220.

[COLI 90] J. Y. COLIN et P. CHRETIENNE, C.P.M. Scheduling with Small Interprocessor Communication Times, *Opns. Res.*, 1990.

[CONW 67] R. W. CONWAY, W. L. MAXWELL et L. W. MILLER, *Theory of Scheduling*, Addison-Wesley Publishing Company, 294 p., 1967.

[COUZ 79] C. COUZINET-MERCE, Etude de l'existence de solutions pour certains problèmes d'ordonnancement, *Thèse de Docteur-Ingénieur*, Université Paul Sabatier, Toulouse, 1979.

[CYTR 84] R. CYTRON, Compile-Time Scheduling and Optimization for Asynchronous Machines, *PHD Thesis* Univ. of Illinois, Urbana-Champaign, 1984.

[DANN 77] D. G. DANNENBRING, A Evaluation of Flow-Shop Sequencing Heuristics, *Management Sci.*, 1977, *23*, 11, p. 1174-1182.

[DEME 90] E. DEMEULEMEESTER et W. HERROELEN, A Branch and Bound Procedure for the Multiple Constrained Resource Project Scheduling Problem, *EURO WG-PMS*, Compiègne, 1990.

[DEMP 82] M. A. H. DEMPSTER, J. K. LENSTRA et A. H. G. RINNOOY KAN, *Deterministic and Stochastic Scheduling*, Reidel Dordrecht, 1982.

[DOGR 79] A. DOGRAMACI et J. SURKIS, Evaluation of a Heuristic for Scheduling Independent Jobs on Parallel Identical Processors, *Management Sci.*, 1979, *25*, 12, p. 1208-1216.

[DU 90] J. DU et J. Y. T. LEUNG, Minimizing Total Tardiness on One Machine is N.P.-Hard, *Math. Opns. Res.*, 1990, *15*, p. 483-495.

[EISE 88] C. EISENBEIS, Optimization of Horizontal Microcode Generation for Loop Structures, *Proc. of the 1988 ACM Int. Conf. on Super-Computing*, St Malo, France, juillet 1988, p. 453-465.

[ELMA 92] S. E. ELMAGHRABY, Optimal Time-Cost Trade-off via Dynamic Programming, Project Management and Scheduling 2, *EJOR*, Special Issue (à paraître).

[ERSC 76 *a*] J. ERSCHLER, Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement, *Thèse d'état*, Université Paul-Sabatier, Toulouse, 1976.

[ERSC 76 *b*] J. ERSCHLER, F. ROUBELLAT et J. P. VERNHES, Finding Some Essential Characteristics of the Feasible Solutions for a Scheduling Problem, *Opns. Res.*, 1976, *24*, p. 774-784.

[ERSC 79] J. ERSCHLER, G. FONTAN et F. ROUBELLAT, Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités, *RAIRO-Oper. Res.*, 1979, *13*, 4, p. 363-378.

[ERSC 80] J. ERSCHLER, F. ROUBELLAT et J. P. VERNHES, Characterizing the Set of Feasible Sequences for *n* Jobs to be Carried out on a Single Machine, *EJOR*, 1980, *4*, p. 189-194.

[ERSC 83] J. ERSCHLER, G. FONTAN, C. MERCE et F. ROUBELLAT, A New Dominance Concept in Scheduling *n* Jobs on a Single Machine with Ready Times and Due Dates, *Opns. Res.*, 1983, *31*, 1, p. 114-127.

[ERSC 85] J. ERSCHLER, G. FONTAN et C. MERCE, Consistency of the Dissaggregation Process in Hierarchical Planning, *Opns. Res.*, 1985, *34*, p. 464-469.

[ERSC 91] J. Erschler, P. Lopez et C. Thuriot, Raisonnement temporel sous contraintes de ressource et problèmes d'ordonnancement, *Revue d'intelligence artificielle*, 1991, *5*, p. 7-32.

[ESQU 87] P. Esquirol, Règles et processus d'inférence pour l'aide à l'ordonnancement de tâches en présence de contraintes, *Thèse de Doctorat*, Université Paul-Sabatier, Toulouse, 1987.

[FALK 91] E. Falkenauer et S. Bouffouix, A Genetic Algorithm for Job Shop, *Proc. of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, avril 1991, p. 824-829.

[FISH 73] M. L. Fisher, Optimal Solution for Scheduling Problems Using Lagrange Multipliers, part 1, *Opns. Res.*, 1973, *21*, p. 1114-1127.

[FOLD 92] S. Foldes et F. Soumis, Generalizations of PERT Using the Network Model, Project Management and Scheduling 2, *EJOR*, Special Issue (à paraître).

[FONT 80] G. Fontan, Notion de dominance et son application à l'étude de certains problèmes d'ordonnancement, *Thèse d'état*, Université Paul Sabatier, Toulouse, 1980.

[FREN 82] S. French, *Sequencing and Scheduling: an Introduction to the Mathematics of the Job Shop*, Horwood, Chichester, 1982.

[GAO 91] G. R. Gao, Y. B. Wong et Qi Ning, A Petri Net Model for Fine Grain Loop Scheduling, *Proc. of the 91 ACM-SIGPLAN conf. on Prog. Lang. Design. and Impl.*, Toronto, juin 1991, p. 204-218.

[GARC 85] H. Garcia et J. M. Proth, Group Technology in Production Management: the Short Horizon Planning Level, *J. Appl. Stoc. Model and Data Analysis*, 1985, *1*, p. 25-34.

[GARC 86] H. Garcia et J. M. Proth, A New Cross-Decomposition Algorithm: the GPM. Comparison with the Bond Energy Method, *Control and Cybernetics*, 1986, *15*, 2, p. 155-165.

[GARE 79] M. R. Garey et D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[GASP 91] F. Gasperoni et U. Schwiegelsohn, Efficient Algorithms for Cyclic Scheduling, *IBM Tech. Rep. RC 17068*, août 1991.

[GILM 64] P. C. Gilmore et R. E. Gomory, Sequencing a One-State Variable Machine: a Solvable Case of the Travelling Salesman Problem, *Opns. Res.*, 1964, *12*, p. 655-679.

[GLOV 75] F. Glover, Surrogate Constraint Duality in Mathematical Programming, *Opns. Res.*, 1975, *23*, p. 434-451.

[GLOV 87] F. Glover, Tabu Search Methods in Artificial Intelligence and Operations Research, ORSA, *Artificial Intelligence Newletter*, 1987, *1*.

[GOLD 89] D. E. Golderg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[GONZ 76] T. Gonzalez et S. Sahni, Open-Shop Scheduling to Minimize Finish Time, *J.A.C.M.*, 1976, *23*, 4, p. 665-679.

[GONZ 78] T. Gonzalez et S. Sahni, Flowshop and Jobshop Schedules: Complexity and Approximation, *Opns. Res.*, 1978, *26*, 1, p. 36-52.

[GRAB 82] J. Grabowski, A New Algorithm of Solving the Flow Shop Problem, G. Fleichtinger et P. Kall ed., *Opns. Res. in Progress*, Reidel, Dordrecht, 1982, p. 57-75.

[GRAH 79] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA et A. H. G. RINNOOY KAN, Otimisation and Approximation in Deterministic Sequencing and Scheduling: a Survey, *Annals Disc. Math.*, 1979, *5*, p. 287-326.

[HAN 92] W. HAN, Algorithmes de résolution exacte et heuristique pour les problèmes d'ordonnancement en flowshop, *Thèse de Doctorat*, École Centrale Paris, France, 156 p., 1992.

[HANE 89] C. HANEN, Microprogramming Using Timed Petri Nets, Advances in Petri Nets, 1989, Springer Verlag.

[HANE 90] C. HANEN, Les tables de réservation numériques : un outil de résolution de certains problèmes d'ordonnancement cycliques, *RAIRO-Oper. Res.*, avril 1990.

[HANE 91] C. HANEN, Study of a NP-Hard Scheduling Problem: the Recurrent Job-Shop, *EJOR* (à paraître).

[HANE 92] C. HANEN et A. MUNIER, Cyclic Scheduling on Parallel processors: Problem Structure and Heuristic, *Rapport CRI*, juin 1992.

[HERZ 90] A. HERTZ et D. DE WERRA, The Tabu Search Metaheuristic: How to Used It, *Ann. of Math. and Artificial Intelligence*, 1990, *1*, p. 111-121.

[HHOO 92] H. HOOGEVEN, Single-Machine Bicriteria Scheduling, *Doctoral thesis*, CWI, Amsterdam, 1992.

[HWA 89] J. J. HWANG, Y. C. CHOW, F. O. ANGER et C. H. LEE, Scheduling Precedence Graphs in Systems with Interprocessor Communication Times, *SIAM J. Comput.*, avril 1989, *18*, p. 244-257.

[JACK 55] J. R. JACKSON, Scheduling a Production Line to Minimize Maximum Tardiness, *Research Report 43, Management Science Research Project*, University of California, Los Angeles, 1955.

[JHAV 91] S. C. JHAVERI et B. L. FOOTE, A Heuristic Algorithm to Schedule Work in the Repair Industry, *Int. J. Prod. Res.*, 1991, *29*, 12, p. 2393-2405.

[JOHN 54] S. M. JOHNSON, Optimal Two- and Three- Stage Production Schedules with Setup Times Included, *Nav. Res. Log. Q.*, 1954, *1*, 1, p. 61-68.

[KARP 67] R. M. KARP, R. E. MILLER et S. WINOGRAD, The Organisation of Computations for Uniform Recurrence Equations, *J. of the A.C.M.*, 1967, *14*, 3, p. 563-590.

[KIRK 82] S. KIRKPATRICK, C. D. GELATT et M. P. VECCHI, Optimization by Simulated Annealing, *Res. Rep. R.C. 9335*, IBM Thomas J. Watson, Center Yorktown, NY, 1982.

[KOGG 81] P. M. KOGGE, *The Architecture of Pipelined Computers*, New York, McGraw Hill, 1981.

[KUSI 85 *a*] A. KUSIAK, A. VANNELLI et K. R. KUMAR, Grouping Problem in Scheduling Flexible Manufacturing Systems, *Robotica*, 1985, *3*, p. 245-252.

[KUSI 85 *b*] A. KUSIAK, The Part Families Problem in Flexible Manufacturing Systems, *Ann. of Opns. Res.*, 1985, *3*, p. 279-300.

[KUSI 86] A. KUSIAK, Efficient Implementation of Johnson's Scheduling Algorithm, *IIE Trans.*, 1986, *18*, p. 215-216.

[KUSI 88] A. KUSIAK et M. CHEN, Expert Systems for Planning and Scheduling Manufacturing Systems, *EJOR*, 1988, *34*, p. 113-130.

[LAM 87] M. LAM, A Systolic Array Optimizing Compiler, *PhD. Dissertation*, Carnegie Mellon University, 1987.

[LAWL 86] E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN et D. B. SHMOYS, *The Travelling Salesman Problem*, John Wiley, 1986.

[LEGA 89] A. LE GALL, Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'atelier, *Thèse de Doctorat*, Université Paul-Sabatier, Toulouse, 1989.

[LEI 89] L. LEI et T. J. WANG, On the Optimal Cyclic Schedules of Single Hoist Electroplating Processes, *GSM Working paper n° 89-06*, Rutgers University, New-Jersey, 1989.

[LEIS 90] R. LEISTEN, Flowshop Problems with Limited Buffer Storage, *Int. J. Prod. Res.*, 1990, *28*, 11, p. 2085-2100.

[LENS 77] J. K. LENSTRA, Sequencing by Enumerative Methods, *Math. Cent. tracts 69*, Centre for Mathematical and Computer Science, Amsterdam, 1977.

[LIN 71] S. LIN et B. W. KERNIGHAN, An Effective Heuristic Algorithm for the Travelling Salesman Problem, *Opns. Res.*, 1971, *21*, p. 498-516.

[McMA 71] G. B. MCMAHON, A Study of Algorithms for Industrial Scheduling Problem, *Ph. D. Thesis*, University of South Wales.

[McMA 92] G. B. MCMAHON, The Two-Machine Maximum Flow-Time Problem with Arbitrary Precedence Relations, *EJOR* (à paraître).

[MATS 88] H. MATSUO, *Cyclic Sequencing Problems in the Two-Machine Permutation Flow Shop: Complexity, Worst Case and Average Case Analysis*, Graduate School of Business, University of Texas, Austin, january 1988.

[MEGU 88] S. MEGUELATI, Méthodes de classification pour la constitution d'îlots de fabrication et l'ordonnancement, *Thèse de Doctorat de l'I.N.S.A.*, Toulouse, 1985.

[MONM 83] C. L. MONMA et A. H. G. RINNOOY KAN, A Concise Survey of Efficiently Solvable Special Cases of the Permutation Flow-Shop Problem, *RAIRO-Oper. Res.*, 1983, *17*, 2, p. 105-109.

[MUNI 91 *a*] A. MUNIER, Résolution d'un problème d'ordonnancement cyclique à itérations indépendantes et contraintes de ressource, *RAIRO-Oper. Res.*, 1991, *25*, 2, p. 161-182.

[MUNI 91 *b*] A. MUNIER, Contribution à l'étude des problèmes d'ordonnancement cycliques, *Thèse de l'Université Paris-VI*, février 1991.

[MUNS 90] A. MUNSHI et B. SIMONS, Scheduling Loops on Processors: Algorithms and Complexity, *SIAM J. of Comput.*, 1990, *19*, p. 728-741.

[NABE 73] I. NABESHIMA, *Algorithms and Reliable Heuristics Programs for Multi-Projects Scheduling with Resource Constraints and Related Parallel Scheduling*, University of Electrocommunication, Chofu, Tokyo, Japon, 1973.

[NAWA 83] M. NAWAZ, E. ENSCORE et I. HAM, A Heuristic Algorithm for the *m* Machine, *n* Job Flowshop Sequence Problem, *Omega*, 1983, *11*, 1, p. 91-95.

[PADB 86] M. PADBERG et G. RINALDI, Optimization of a 532-City Symmetric Travelling Salesman Problem, AFCET, *Journées du 20e anniversaire du groupe combinatoire A.F.C.E.T./I.N.R.I.A.*, décembre 1986.

[PARK 77] R. G. PARKER, R. H. DEANE et R. A. HOLMES, On the Case of a Vehicle Routing Algorithm for the Parallel Processor Problem with Sequence Dependent Changeover Costs, *AIIE Trans.*, 1977, *9*, 2, p. 155-160.

[PATE 76] J. H. PATEL et E. S. DAVIDSON, Improving the Throughput of a Pipeline by Insertion of Delays, *IEEE 3rd Ann. Symp. on Computers Architecture*, january 1976.

[PICO 91] C. PICOULEAU, Two New NP-Complete Scheduling Problems with Communication Delays and Unlimited Number of Processor, *Rapport MASI n° 91-24*, 1991.

[PINS 88] E. PINSON, Le problème de Job Shop, *Thèse de Doctorat* de l'Université Paris-VI, 1988.

[PORT 88] M. C. PORTMANN, Méthodes de décomposition spatiales et temporelles en ordonnancement de la production, *RAIRO-APII*, 1988, *22*, 5, p. 439-451.

[POTT 87] C. N. POTTS et L. N. VAN WASSENHOVE, Dynamic Programming and Decomposition Approaches for the Single Machine Total Tardiness Problem, *EJOR*, 1987, *32*, p. 405-414.

[PRIN 88] C. PRINS, Problèmes d'optimisation de ressources dans les systèmes de télécommunications par satellite utilisant l'A.M.R.T. (Accès Multiple à Répartition dans le Temps), *Thèse de Doctorat* de l'Université de Paris-VI, juin 1988.

[PROU 87] C. PROUST, M. DROGOU, J.-M. FOUCHER et E. FOUCHEYRAND, Une heuristique pour le problème d'ordonnancement statique de type *n/m/*flowshop avec prise en compte des temps de montage et démontage d'outils, *2e Conférence Internationale Systèmes de Production, I.N.R.I.A.*, Paris, 1987, p. 125-141.

[PROU 89] C. PROUST, J.-C. BILLAUT, A. LE SAUX et A.-M. SALAUN, Un logiciel de planification pour outil à deux étages avec changement de fabrication, 1989, *Actes du Colloque International « Logistique »*, A.F.C.E.T., Paris, p. 125-132.

[PROU 91 *a*] C. PROUST, J. N. D. GUPTA et V. DESCHAMPS, Flowshop Scheduling with Setup, Processing and Removal Times Separated, *Int. J. Prod. Res.*, 1991, *29*, 3, p. 479-493.

[PROU 91 *b*] C. PROUST, A. FERREIRA et J. BIJAOUI, Le paradoxe de l'Interprogrammation : la diversité des logiciels au service d'une intégration efficace, 1991, *Actes du 3e Congrès International de Génie Industriel, GGI*, Tours, p. 1157-1166.

[PROU 92] C. PROUST, Influence des idées de S. M. Johnson sur la résolution de problèmes d'ordonnancement de type *n/m/F*, contraintes diverses/C_{max}, 1992, *Rapport Interne, Laboratoire d'Informatique/E3i*, Université de Tours, 150 p.

[QUIN 89] P. QUINTON et Y. ROBERT, *Algorithmes et architectures systoliques*, Etudes et Recherches en Informatique, Masson, 1989.

[RADH 86] R. RADHARAMAN, A Heuristic Algorithm for Group Scheduling, *Proc. of the Int. Industrial Engineering Conference*, Institute of Industrial Engineers, 1986, p. 229-236

[RINN 76] A. H. G. RINNOOY KAN, *Machine Sequencing Problems: Classification, Complexity and Computation*, Nijhoff, The Hague, 1976.

[ROCK 82] H. ROCK et G. SCHMIDT, *Machine Agregation Heuristics in Shop Scheduling*, 1982, Bericht 82-11, Fachbereich 20, Mathematich Technische Universitat, Berlin.

[ROUN 88] R. ROUNDY, *Cyclic Schedules for Job Shops with Identical Jobs*, School of Operation Research and Industrial Engineering, Cornell University, T.R. n° 766, juillet 1988.

[ROY 70] B. ROY, *Algèbre moderne et théorie des graphes*, tome 2, Dunod, Paris, 1970.

[SAOU 90] Y. Saouter et P. Quinton, Computability of Recurrence Equations, *Rapport de recherche I.N.R.I.A. n° 1203,* 1990.

[SERA 89] P. Serafini et W. Ukovich, A Mathematical Model for Periodic Scheduling Problems, *S.I.A.M. J. of Disc. Math.,* 1989, *2*, 4.

[SLOW 78] R. Slowinski, Scheduling Preemptable Tasks on Unrelated Processors with Additionnal Ressources to Minimize Schedule Lenght, *Lect. Notes on Comput. Sci.,* Springer-Verlag, 1978, p. 536-547.

[SLOW 82] R. Slowinski, Multiobjective Network Scheduling with Efficient Use of Renewable and Non-Renewable Resources, *EJOR,* 1982, *7*, 3, p. 265-273.

[SLOW 89] R. Slowinski et J. Weglarz éd., *Advances in Project Scheduling*, Elsevier Science, Amsterdam, avril 1989.

[SMIT 56] W. E. Smith, Various Optimizers for Single-Stage Production, *Nav. Res. Log. Quart.,* 1956, *3*, p. 59-66.

[SRIN 71] V. Srinivasan, A Hybrid Algorithm for the One-Machine Sequencing Problem to Minimize Total Tardiness, *Nav. Res. Log. Quart.,* 1971, *18*, 3, p. 317-327.

[SUMI 87] R. T. Sumichrast et J. R. Baker, Scheduling Parallel Processors: an Integer Linear Programming Based Heuristic for Minimizing Setup Time, *Int. J. Prod. Res.*, 1987, *25*, 5, p. 761-771.

[TALB 78] F. B. Talbot et J. H. Patterson, An Efficient Integer Programming with Network Cuts for Solving Resources Constrained Scheduling Problems, *Management Sci.*, 1978, *245*, p. 1163-1174.

[TAVA 90] L. Tavares et J. Weglarz éd., Project Management and Scheduling, *EJOR*, Special Issue 1, novembre 1990.

[THOM 80] V. Thomas Aide à la décision pour l'ordonnancement d'atelier en temps réel, *Thèse de Doctorat*, Université Paul-Sabatier, Toulouse, 1980.

[VELD 91] S. L. Van de Velde, Machine Scheduling and Lagrangian Relaxation, *Doctoral Thesis*, CWI, Amsterdam, 1991.