De l'ordonnancement déterministe à l'ordannancement distribué sous incertitudes

Samia Ourari

Thèse

Présentée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vue de l'obtention du grade de
Doctorat de l'Université Paul Sabatier de Toulouse

Spécialité : **Systèmes Industriels**

par

Samia OURARI
Ingénieur de l'ENP d'Alger et Magister de l'EMP d'Alger

DE L'ORDONNANCEMENT DÉTERMINISTE À L'ORDONNANCEMENT DISTRIBUÉ SOUS INCERTITUDES

---

soutenue le 28 Janvier 2011 devant le jury :

Rapporteurs :	Jean-Charles BILLAUT	Professeur de l'Université de Tours
	Aziz MOUKRIM	Professeur de l'Université de Technologie de Compiègne
Examinateurs :	Michel COMBACAU	Professeur de l'Université Paul Sabatier
	Eric SANLAVILLE	Professeur de l'Université du Havre
Directeurs de thèse :	Cyril BRIAND	Maître de Conférences-HDR de l'UPS, Toulouse

---

Avant-propos

Le travail présenté dans ce mémoire a été effectué, au Laboratoire d'Analyse et d'Architecture des Systèmes du C.N.R.S (Toulouse), au sein de l'équipe Modélisation, Optimisation et Gestion Intégrée de Systèmes d'Activités, et au Centre de Développement des Technologie Avancées (Alger) au sein de l'équipe Systèmes Robotisés de Production. A ce titre, je tiens à remercier Pierre Lopez, Chargé de Recherche au C.N.R.S, de m'avoir accueilli au sein du groupe MOGISA, et Monsieur Brahim Bouzouia, Directeur de recherche au CDTA, et responsable de l'équipe Systèmes Robotisés de Production à laquelle j'appartiens pour sa confiance.

Ce travail doit énormément à mon Directeur de thèse, C. Briand qui m'a encadré pour la réalisation des travaux. Qu'il trouve ici l'expression de ma profonde gratitude et ma haute reconnaissance pour la collaboration active et fructueuse que nous avons eu. Grâce à sa patience, sa disponibilité pendant mes séjours au LAAS ou bien à travers ses appels téléphonique du LAAS au CDTA, ainsi qu'à ses encouragements, ce mémoire à pu être concrétisé et mené à terme.

Je tiens à remercier Messieurs, Michel COMBACAU, Professeur à l'Université Paul Sabatier, pour avoir accepté de présider le jury de ma soutenance, Jean-Charles BILLAUT, Professeur à l'Université de Tours et Aziz MOUKRIM, Professeur à l'université de Technologie de Compiègne pour avoir rapporté cette thèse et participé au jury de ma soutenance, ainsi que Eric SANLAVILLE, Professeur à l'Université du Havre, pour avoir accepté de participer à mon jury de thèse.

Mes remerciement vont également aux collègues et amis du CDTA, et ceux du groupe MOGISA, pour leurs sympathies et encouragements. Et pour finir, je ne remercierai jamais assez mes proches et amis pour leurs aides et soutien moral.

Table des matières

Table des figures

Liste des tableaux

INTRODUCTION

L'ordonnancement consiste à organiser dans le temps l'exécution d'un ensemble de tâches au moyen d'un ensemble de ressources tout en satisfaisant un ensemble de contraintes et/ou en optimisant un ou plusieurs objectifs. C'est une fonction qui joue un rôle essentiel et peut être rencontrée dans de nombreux domaines : les systèmes industriels de production (gestion de production, gestion de projets), les systèmes informatiques (ordonnancement de processus), les systèmes administratifs (gestion hospitalière), etc. Dans le cas où les ressources sont disjonctives (machines), un problème d'ordonnancement est un problème d'optimisation combinatoire dont la solution caractérise pour chaque ressource, une séquence de tâches. Les travaux scientifiques concernant les problèmes d'ordonnancement et les méthodes de résolution foisonnent à l'heure actuelle dans la littérature.

De très nombreuses études se sont portées sur la résolution de problèmes déterministes. Les auteurs considèrent dans ce cas que les données du problème sont parfaitement connues, et proposent des méthodes pour générer un ordonnancement unique satisfaisant les contraintes du système et ayant des performances optimales ou proches des optimums. Ces méthodes sont alors, ou bien exactes (plus ou moins efficaces selon la difficulté du problème), ou bien approchées (avec ou sans garantie de performance). Notons, toutefois, que ces problèmes demeurent ardus dans la plupart des cas et le développement de nouvelles méthodes d'ordonnancement déterministes est toujours d'intérêt dans le domaine théorique.

Aujourd'hui, l'hypothèse du déterminisme des problèmes d'ordonnancement est souvent jugée trop restrictive. En effet, en pratique, les données du problème ne sont pas toujours connues à l'avance. De plus, de nouvelles informations apparaissent durant l'exécution du programme prévisionnel susceptibles de rendre sa mise en œuvre caduque. Ce constat sur la réalité de l'environnement d'application de l'ordonnancement, à la fois non déterministe et incertain, a poussé plusieurs chercheurs à poser la problématique de l'ordonnancement avec gestion des incertitudes afin que les travaux théoriques en ordonnancement puissent être destinés à être mis en œuvre dans la pratique. De manière générale, cela a conduit l'émergence d'une nouvelle thématique de recherche appelée *ordonnancement robuste*.

Dans cette étude, la contribution à la résolution des problèmes d'ordonnancement a

évolué selon deux axes, leur point commun étant l'utilisation d'un théorème de dominance démontré dans les années quatre-vingt, baptisé théorème des pyramides. Ce théorème, formulé initialement pour limiter la complexité algorithmique liée à la recherche d'une solution admissible pour un problème d'ordonnancement à une machine avec fenêtres d'exécution, a été aussi exploité dans un travail antérieur dans un contexte de robustesse, pour évaluer en temps polynomial les performances de l'ensemble dominant qu'il caractérise.

Le premier axe abordé dans cette thèse concerne la résolution exacte de problèmes d'ordonnancement à une machine pour lesquels nous proposons de nouvelles formulations de programmation linéaire en nombres entiers. Le second axe concerne l'étude du problème d'ordonnancement de type job shop dans un contexte perturbé : en supposant que chaque ressource, assimilée à un centre de décision, dispose d'une autonomie décisionnelle, nous proposons une nouvelle approche d'ordonnancement distribuée, dont la solution globale résulte de la concaténation d'ordonnancements locaux, eux-mêmes élaborés par des négociations successives entre des couples d'acteurs.

Le manuscrit est organisé comme suit :

La **Partie I** décrit quelques concepts nécessaires à la compréhension de la suite du manuscrit. Nous commençons par présenter brièvement les problèmes d'ordonnancement et donner un aperçu des approches classiques de résolution. Ensuite, nous discutons les limites des hypothèses classiques en ordonnancement, ou en d'autres termes les limites des ordonnancements déterministes et centralisés et montrons l'intérêt, des approches de résolution robustes et de la distribution de la fonction ordonnancement.

La seconde **partie II** s'intéresse à l'étude du problème d'ordonnancement à une machine. Dans un premier temps, la notion de condition nécessaire et suffisante d'admissibilité énoncée par le théorème des pyramides est rappelée et présentée, pour ensuite énoncer de nouvelles conditions analytiques et numériques pour la recherche de la solution la plus dominante. Nous montrons aussi sous quelles conditions, le théorème des pyramides est dominant vis-à-vis du critère de minimisation du nombre de travaux en retard. Dans un second temps, à partir des résultats énoncés et démontrés, et considérant tour à tour les critères d'admissibilité, puis la minimisation du retard algébrique maximal et enfin la minimisation du nombre de travaux en retard, nous proposons de nouvelles formulations mathématiques sous forme de programmes linéaires en nombres entiers. Des expérimentations sont également présentées pour l'évaluation des performances des modèles qui mettent en évidence leur efficacité.

La dernière **partie III** s'intéresse à un problème à plusieurs ressources en contexte perturbé. Nous admettons d'une part, que la recherche d'une solution optimale ou presque optimale, souvent coûteuse, ne constitue pas un objectif réaliste, en raison des aléas qui

viennent changer la structure du problème lors de sa mise en œuvre, et que d'autre part, il est possible de tirer profit d'une organisation distribuée de la fonction ordonnancement, en exploitant l'autonomie de décision dont peuvent disposer les différentes ressources. De là, une approche nouvelle d'ordonnancement coopératif permettant à la fois, d'anticiper et de réagir hors-ligne aux perturbations et, d'intégrer une dimension distribuée dans sa résolution, est proposée. C'est une approche qui consiste à déterminer, à la place d'un ordonnancement de référence, un ensemble de solutions flexibles au niveau de chaque centre de décision et à faire appel à la coopération entres les différents acteurs pour l'ordonnancement global du système de fabrication.

Dans cette partie, un aperçu des méthodes traitant de l'ordonnancement sous incertitudes est donné dans un premier chapitre, puis, une méthode d'ordonnancement robuste à une machine avec apport de flexibilité séquentielle est rappelée dans un second chapitre. La problématique d'ordonnancement multi-ressources est abordée dans le chapitre suivant, puis une approche heuristique de résolution, permettant de construire de façon distribuée et coopérative la solution d'ordonnancement est décrite dans le dernier chapitre.

Première partie

Notion préliminaires et contexte

Chapitre 1

Présentation générale des problèmes d'ordonnancement

1.1 Introduction aux problèmes d'ordonnancement

Ce chapitre constitue un rappel de quelques notions de base relatives aux problèmes d'ordonnancement. Il rappelle aussi quelques concepts sur la théorie de la complexité, et donne un aperçu des approches classiques de résolution des problèmes d'ordonnancement.

1.1.1 Définition

On trouve dans la littérature plusieurs ouvrages consacrés à l'ordonnancement, nous citons [Baker 74], [French 82], [Carlier & Chrétienne 88], [Mortan & Pintico 93], [Chrétienne *et al.* 95], [Chu & Proth 96], [Esquirol & Lopez 99], [Lopez & Roubelat 01] et [Pinedo 08].

Plusieurs définitions de l'ordonnancement y sont données, nous reprenons celle proposée dans [Carlier & Chrétienne 88] :

> *" Ordonnancer c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution."*

Le terme réalisation a un sens large. En effet, selon le champ d'application de l'ordonnancement considéré, la réalisation peut concerner, en production, les opérations à effectuer sur des machines ; en informatique parallèle, les morceaux de programmes à exécuter sur des processeurs ; ou en gestion de projets, les activités (ou actions) à accomplir nécessitant une ou plusieurs ressources.

Une opération est une activité élémentaire, reliée à d'autres opérations par des contraintes temporelles et nécessitant pour sa réalisation l'utilisation d'une ou plusieurs ressources. Un travail est constitué de plusieurs opérations. On appelle gamme d'un travail ou gamme opératoire, la succession des opérations qui le constituent, ou la succession des machines qui

leur sont associées. Les contraintes et les critères du problème concernent les opérations, leurs localisations temporelles, et les moyens nécessaires à leur réalisation.

Résoudre un problème d'ordonnancement, c'est choisir pour chaque opération une date de début, de telle sorte que les contraintes du problème soient respectées (la solution est alors dite admissible ou réalisable) et qu'un ou plusieurs critères donnés soient optimisés. Un ordonnancement est la solution au problème d'ordonnancement.

1.1.2 Éléments fondamentaux d'un problème d'ordonnancement

Un problème d'ordonnancement est défini par l'ensemble des opérations à réaliser, les machines disponibles pour les exécuter, les contraintes relatives aux opérations et aux machines ainsi que les fonctions objectifs à optimiser. Généralement, nous considérons un problème $V = \{J_1, J_2, ..., J_n\}$ de n travaux à ordonnancer sur m machines $\{m_1, m_2, ..., m_m\}$. Si nous utilisons l'indice j pour désigner le travail et l'indice i pour désigner la machine, alors les principales notations des données relatives à ce problème sont :

- *La durée opératoire* (processing time en anglais) $p_{i,j}$: c'est le temps d'exécution du travail j sur la machine i. Il sera noté p_j si le travail doit être exécuté sur une seule machine.

- *La date de disponibilité* (release date en anglais) r_j : c'est la date d'arrivée du travail j dans le système, elle représente aussi sa date de début au plus tôt.

- *La date d'échéance* (due date en anglais) d_j : c'est la date de fin d'exécution préférentielle du travail j (date de livraison promise de j pour le client) ; lorsque cette date d'échéance doit être respectée, on parlera de date d'échéance stricte (deadline en anglais) notée, $\tilde{d}_j$.

- *Le poids* (weight en anglais) w_j : c'est un facteur de priorité qui dénote l'importance du travail j relativement aux autres.

Plusieurs notations ont été introduites dans la littérature pour spécifier un problème d'ordonnancement. Un schéma de classification proposé par [Graham *et al.* 79], puis repris par [Blazewicz *et al.* 96], structure les données d'un problème d'ordonnancement sur la base d'une notation à trois champs distincts : $\alpha|\beta|\gamma$. Le champs α décrit l'environnement machine, β précise les contraintes liées aux opérations, et le champs γ décrit le ou les critères à optimiser.

Les environnements machines les plus courants décomposent le champ α en deux sous champs α_1 et α_2. Le premier indique la nature du problème (job shop, flow shop, etc.)

alors que le second précise le nombre de machines ou d'ensembles de machines (pools de ressources).

Les différents environnements machines possibles, spécifiés dans le champs α_1, sont :

- *Ressource unique* : c'est le cas simple où une seule ressource disjonctive (ne pouvant traiter qu'une opération à la fois) exécute les tâches. Dans ce cas, le champ $\alpha_1 = \varnothing$ ou est absent.

- *Machines parallèles* : il existe m machines parallèles. Chaque opération doit être exécutée sur une machine à sélectionner dans l'ensemble des m machines parallèles. Les machines peuvent être alors identiques, uniformes ou indépendantes. Le champs α_1 prend ainsi respectivement les valeurs P, Q ou R.

- *Les ateliers à cheminement unique* ou "flow shop" : il existe m machines en série. Tous les travaux ont la même gamme opératoire, et l'ordre de visite des travaux est le même pour chaque machine. Le champ $\alpha_1 = F$.

- *Les ateliers à cheminement multiples* ou "job shop" : c'est un cas plus général que le flow shop, où chaque travail à sa propre gamme opératoire. Le champ $\alpha_1 = J$.

- *Les ateliers à cheminements libre* "ou open shop" : il existe m machines. L'ordre de passage sur les machines est libre pour tous les travaux, i.e., il n'existe aucune relation de précédence entre les opérations d'un même travail. Le champ $\alpha_1 = O$.

La Figure suivante (issue de [Billaut 99]) présente une classification des problèmes d'ordonnancement en fonction de la nature des ressources et des gammes des travaux à réaliser.

Le second champ β correspond aux contraintes liées aux opérations. Il peut inclure plusieurs sous-champs. Les principales valeurs de ces sous-champs sont données ci-dessous :

- *Dates de disponibilité* "r_j" : quand cette valeur apparaît dans le champs β cela veut dire que les dates de disponibilité des tâches sont distinctes (toutes les tâches commencent leur exécution après cette date). Si au contraire, toutes les tâches sont disponibles à l'instant initial, alors la valeur r_j est absente du champs β.

- *La préemption* "*pmtn*" : la préemption signifie qu'une machine peut interrompre à tout moment l'exécution d'une opération pour en exécuter une autre. Ainsi, quand la valeur *pmtn* apparaît dans le champs β cela implique que la préemption est autorisée.

- *La précédence* "*prec*" : on dit que deux travaux j_1 et j_2 sont liés par une contrainte de précédence si j_2 ne peut commencer son exécution avant la fin d'exécution de j_1,

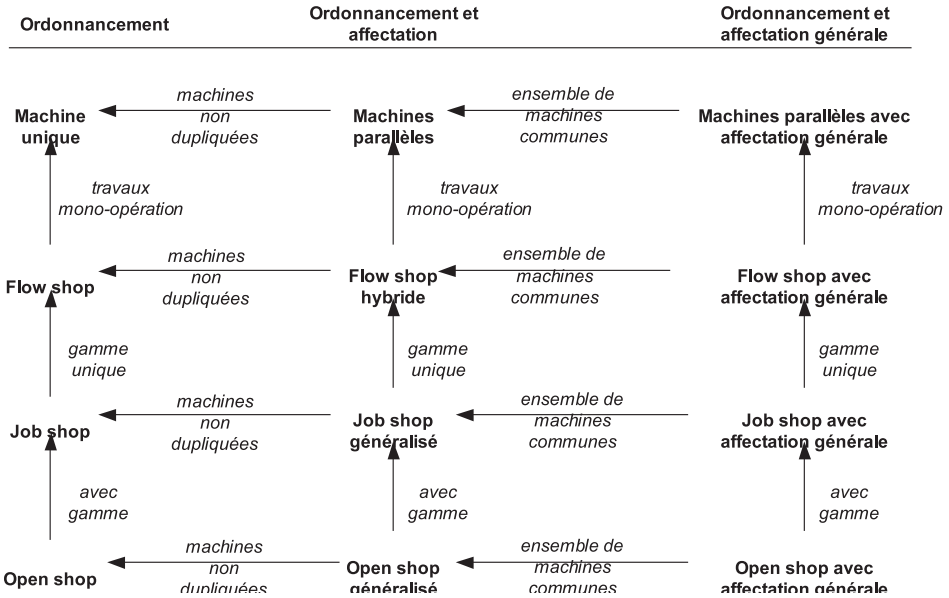

FIG. 1.1: Une typologie des problèmes d'ordonnancement (d'après [Billaut 99])

et on note $j_1 \prec j_2$. Si la valeur *prec* n'apparaît pas dans le champ β, cela signifie que les travaux ne sont pas soumis aux contraintes de pécédence.

- *No-wait* "nwt" : les opérations de chaque travail doivent se succéder sans attente devant la machine.

- *Durées opératoires égales* "$p_j = p$". Si la valeur $p_j = p$ apparaît dans le champ β cela veut dire que toutes les opérations ont la même durée opératoire.

Dans le dernier champ γ apparaît le critère d'optimisation. Le plus souvent, le critère est exprimé en fonction des variables de décision associées à chaque tâche j, soit :

- la date de fin d'exécution du travail j sur la dernière machine : C_j ;

- la durée de séjour du travail j dans l'atelier (Flow-time). Son expression est donnée par $F_j = C_j - r_j$;

- le retard algébrique (lateness) qui dénote l'écart par rapport au délai souhaité avec avance favorable et retard défavorable. Il est exprimé par $L_j = C_j - d_j$;

- le retard vrai (tardiness) donné par l'expression, toujours positive, $T_j = max(0, C_j - d_j) = \max(0, L_j)$;

- l'indicateur de retard (unit penalty) définit par $U_j = \begin{cases} 0 & \text{si } C_j \geq d_j \\ 1 & \text{sinon} \end{cases}$
- l'avance (earliness) $E_i = \max(0, d_j - C_j) = -\min(0, L_j)$.

Les indicateurs de performance usuelles sont alors :

- la durée totale de l'ordonnancement (makespan) $C_{max} = \max_j C_j$;
- le plus grand retard algébrique $L_{max} = \max_j L_j$;
- la plus grande durée de séjour $F_{max} = \max_j F_j$;
- le plus grand retard vrai $T_{max} = \max_j T_j$;
- le nombre de tâches en retard $N_T = \sum_{j=1}^{n} U_j$;
- le retard moyen $\overline{T} = \frac{1}{n} \sum_{j=1}^{n} T_j$;
- la durée moyenne de séjour $\overline{F} = \frac{1}{n} \sum_{j=1}^{n} F_j$.

À partir de ces mesures, des critères d'appréciation ou d'évaluation des solutions du problème d'ordonnancement sont construits. Pour celles que nous avons présentées, l'objectif consiste dans tous les cas à minimiser ces mesures de performance. On remarque que tous ces critères sont basés sur l'inconnue C_j et des fenêtres d'exécution $[r_j, d_j]$ associées aux opérations.

On note que les critères présentés ci-dessus sont tous réguliers. On dit qu'un critère, fonction des variables C_j, est *régulier* si sa valeur décroît lorsqu'au moins un des C_j décroît, les autres restant fixes (on cherche à exécuter les tâches au plus tôt).

1.1.3 Représentation à l'aide d'un graphe potentiels-tâches

Une manière de modéliser les contraintes d'un problème d'ordonnancement d'atelier est d'utiliser un *graphe potentiels tâches* [Roy & Sussmann 64]. Ce graphe noté $G(X, U, F)$ est un graphe orienté. X représente l'ensemble des nœuds correspondant aux opérations à ordonnancer. Cet ensemble contient parfois deux opérations fictives appelées *source* et *puit* representant respectivement le début et la fin de l'ordonnancement. U est l'ensemble des arcs qui représente les contraintes de précédence entre les opérations d'une même tâche. Chaque arc $u = (i, j) \in U$ de longueur $\delta_{i,j}$, appelé *arc conjonctif*, correspond à une inégalité de potentiels entre les deux nœuds i et j ($t_j - t_i \geq \delta_{i,j}$). Le potentiel t_i correspond généralement à la date de début de l'opération i, et $\delta_{i,j}$, à la valeur de la durée opératoire de l'opération origine i. F contient l'ensemble des arcs *disjonctifs* non orientés (arêtes). Ces arcs non-conjonctifs connectent mutuellement des opérations non ordonnées nécessitant l'utilisation de la même machine. On note qu'un arc disjonctif peut être représenté par deux arcs orientés de sens opposé.

Trouver une solution admissible à ce problème, i.e. trouver un ordre de passage des

opérations sur les ressources, revient à orienter les arcs disjonctifs sans créer de circuits. Le plus long chemin dans le graphe résultant définit alors la durée totale de l'ordonnancement.

À titre d'illustration, considérons un problème job-shop à 3 machines M_1, M_2 et M_3 et 4 travaux J_1, J_2, J_3 et J_4. Tous les travaux sont disponibles à l'instant zéro. Les données concernant les gammes et les durées opératoires sont données dans le Tableau 1.1.

Tâche J_i	1	1	1	2	2	3	3	3	4	4
Opération i	1	2	3	4	5	6	7	8	9	10
p_i	7	10	5	3	15	30	3	2	6	3
M_i	M_1	M_2	M_3	M_2	M_1	M_3	M_1	M_2	M_3	M_1

TAB. 1.1: Exemple d'un problème de job-shop à 3 machines

Le graphe disjonctif représentant ce problème est donné par la Figure 1.2

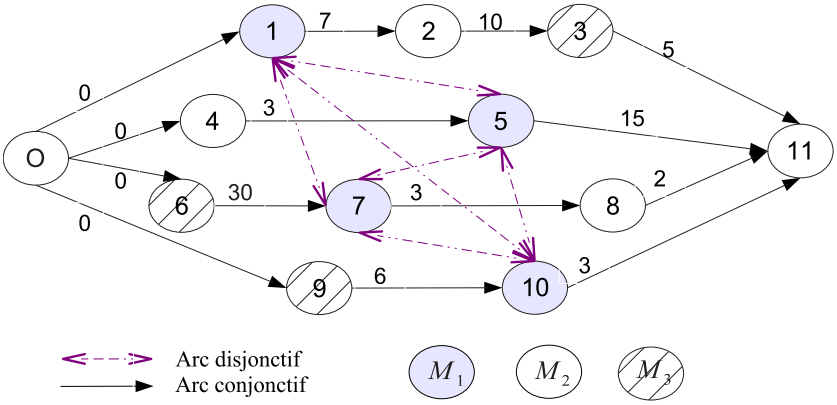

FIG. 1.2: Représentation graphique du problème du tableau 1.1

Seuls les arcs disjonctifs connectant les opérations qui requièrent la machine M_1 sont représentés sur la Figure 1.2. Une solution spécifiée par les ordres de passage des différentes opérations sur les machines, par exemple $M_1(1, 5, 10, 7)$, $M_2(4, 2, 8)$ et $M_3(9, 6, 3)$ voir Figure 1.3, correspond à une solution minimisant le makespan C_{max}.

La durée totale de l'ordonnancement C_{max} est ici égale à 41.

1.1.4 Classes d'ordonnancement remarquables

La solution d'un problème d'ordonnancement est par abus de language appelée ordonnancement. Ce dernier est défini, comme évoqué précédemment, par la valeur, pour chaque tâche j, de sa date de fin C_j ou, de manière équivalente, par sa date de début $t_j = C_j - p_j$,

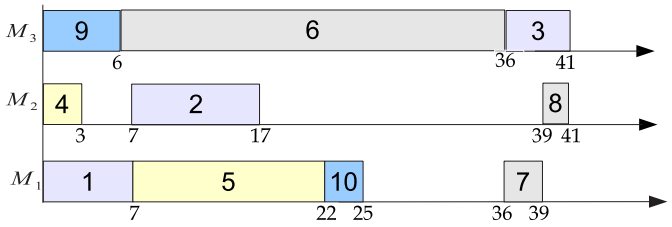

FIG. 1.3: Solution du problème du tableau 1.3

ainsi que par la valeur du critère de la solution considérée.

Fréquemment, on peut restreindre l'espace de recherche des solutions à un sous ensemble d'ordonnancements particuliers. On s'intéresse en particulier aux sous ensembles dominants. On dit qu'un sous ensemble de solutions est *dominant* pour l'optimisation d'un critère donné, s'il contient au moins un optimum pour ce critère. On dit qu'un sous-ensemble est *dominant* relativement à l'admissibilité lorsque le sous ensemble contient au moins une solution admissible, s'il en existe. Rappelons qu'un ordonnancement est dit *admissible* s'il respecte toutes les contraintes du problème.

Trois classes d'ordonnancement sont généralement considérées dans la littérature : les ordonnancements actifs, semi-actifs et sans délai.

- Un ordonnancement est dit *actif* si aucune tâche ne peut être démarrée plus tôt sans décaler vers la droite une autre tâche.

- Dans un ordonnancement *semi-actif* (ou calé au plus tôt), aucune tâche ne peut être exécutée plus tôt sans changer l'ordre d'exécution des autres tâches sur les ressources. Notons qu'un ordonnancement actif est semi-actif.

- Un ordonnancement est dit *sans délai* si aucune ressource n'est laissée inactive alors qu'elle pourrait commencer l'exécution d'une autre opération.

Les relations entre les différentes classes d'ordonnancement sont illustrées par le diagramme de *Venn* donnée sur la Figure 1.4.

Pour l'optimisation d'un critère régulier, il est suffisant de considérer l'ensemble des ordonnancements semi-actifs ; ces derniers dominent l'ensemble de tous les ordonnancements. Cependant leur nombre important amène à considérer souvent les hypothèses plus restrictives des ordonnancements actifs pour améliorer la recherche de solutions optimales. l'ensemble des ordonnancements actifs est le plus petit ensemble dominant [Esquirol & Lopez 99]. Notons également que lorsqu'on considère la recherche d'ordonnancements admissibles, des conditions de dominance permettant de limiter l'espace de recherche de solutions peuvent

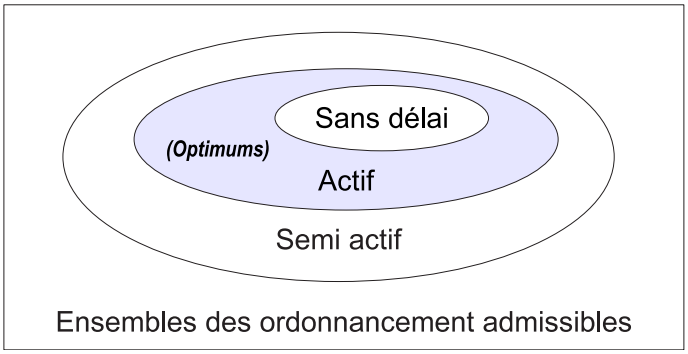

FIG. 1.4: Diagramme de *Venn*

être mises en évidence. Nous reviendrons en détail sur cette notion dans le Chapitre 3 de la partie II.

1.2 Quelques notions sur la complexité des problèmes et des algorithmes

La théorie de la complexité a pour but d'apporter des informations sur la difficulté théorique d'un problème à résoudre. Elle permet de classer "du point de vue mathématique" les problèmes selon leur difficulté [Esswein 03].

La difficulté d'un problème est mesurée par la complexité du meilleur algorithme résolvant le problème de façon optimale. La performance d'un algorithme est évaluée sur la base du temps de calcul (nombre d'opérations élémentaires $T(n)$ nécessaires à l'exécution de l'algorithme pour une instance de taille n) et de la taille mémoire (taille nécessaire pour stocker les différentes structures de données) requise pour résoudre le problème. On dit souvent que le temps d'exécution dépend de la taille de l'instance du problème. La complexité algorithmique est un concept fondamental qui permet de mesurer les performances d'un algorithme et de le comparer avec d'autres algorithmes réalisant les mêmes fonctionnalités.

Un algorithme est dit de complexité polynomiale $O(p(n))$ si son temps d'exécution est borné par une fonction polynomiale p en la taille de l'instance n. Si $p(n)$ est bornée supérieurement par un polynôme d'ordre k, on dira alors que l'algorithme est polynomial d'ordre k, i.e. $T(n) = O(n^k)$. Si par contre il n'existe pas de polynôme $p(n)$ tel que $T(n) = O(p(n))$, par exemple si $T(n) = 2^n$, alors l'algorithme est dit de complexité exponentielle.

Les problèmes d'ordonnancement correspondent soit à des problèmes de décision, soit à des problème d'optimisation. Les problèmes de décision sont des problème pour lesquels on cherche à répondre par "oui" ou "non" à une question donnée. Un problème d'optimisation

est un problème pour lequel on cherche à déterminer une solution qui optimise un critère. Les problèmes d'optimisation étant les plus couramment étudiés, il est possible d'associer à tout problème d'optimisation un problème de décision. Par exemple, le problème de décision associé au problème d'optimisation $F2|r_j|C_{max}$ est : existe-t-il un ordonnancement tel que $C_{max} \leq b$? b étant une valeur connue.

Dans la théorie de la complexité, la résolution d'un problème d'existence s'effectue en deux phases : la recherche d'une solution et la vérification de la solution. On définit alors deux types de classes, la classe $\mathcal{P}$ et la classe $\mathcal{NP}$.

Un problème de décision appartient à la classe $\mathcal{P}$ si et seulement si il existe un algorithme polynomial permettant de le résoudre (la recherche et la vérification de la solution réalisable peut se faire en un temps polynomial). Il est considéré comme étant un problème "facile". Tous les problèmes de décision n'appartiennent pas forcément à la classe P. Ainsi, un problème appartient à la classe $\mathcal{NP}$ si et seulement il existe un algorithme non déterministe polynomial permettant de le résoudre (la vérification de la solution peut se réaliser en temps polynomial indépendamment du temps nécessaire à la recherche de la solution proprement dite).

Il est clair que $\mathcal{P} \subseteq \mathcal{NP}$. La théorie de la complexité ne permet pas encore d'affirmer que ces deux classes sont différentes, une conjecture est que $\mathcal{P} \neq \mathcal{NP}$ et cela signifie que certains problèmes "difficiles" n'ont pu être résolus jusqu'à présent par des algorithmes polynomiaux. Ces problèmes appartiennent à une autre sous-classe de problèmes NP : la classe des problèmes $\mathcal{NP}-$*complets*. Un problème de décision est dit $\mathcal{NP}-$*complet*, s'il appartient à $\mathcal{NP}$ et si tous les problèmes de $\mathcal{NP}$ se réduisent polynomialement en ce problème. On remarque que si on parvient à définir un algorithme polynomial pour un problème $\mathcal{NP}$-*complet* alors $\mathcal{P} = \mathcal{NP}$. Pour les problèmes d'optimisation, un problème est dit $\mathcal{NP}$-*difficile* si et seulement si le problème de décision qui lui est associé est $\mathcal{NP}$-*complet*. La classe des problèmes $\mathcal{NP}$-*complets* contient les problèmes les plus difficiles à résoudre dans $\mathcal{NP}$.

Plusieurs méthodes sont proposées dans la littérature pour résoudre les problèmes d'ordonnancement, selon leur classe de complexité. On trouve des algorithmes polynomiaux qui résolvent les problèmes "faciles". Par contre, deux alternatives sont possibles pour le cas des problèmes $\mathcal{NP}$-*difficiles* :

- les algorithmes exactes de type exponentiel qui permettent de garantir l'optimalité de la solution du problème, mais ne peuvent traiter que les problèmes de taille réduite en un temps de calcul raisonnable ;

- les algorithmes polynomiaux (heuristiques) qui permettent de donner des solutions

approchées mais sans garantie d'optimalité du problème, en un temps de calcul raisonnable.

La Figure 1.5 présente un arbre de réduction des critères d'optimisation en ordonnancement. Il s'interprète de la manière suivante. Pour un environnement d'atelier donné et pour des contraintes et un critère donnés, si un problème est $\mathcal{NP}$-*difficile*, tous ses successeurs dans l'arbre le sont également.

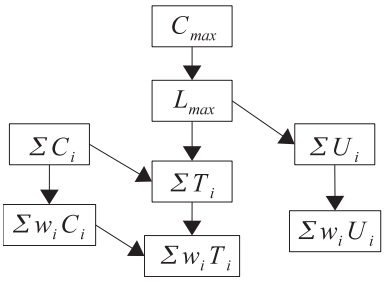

FIG. 1.5: Arbre de réduction des problèmes selon les critères

1.3 Aperçu des approches classique de résolution

Nombreuses sont les méthodes proposées dans la littérature pour l'ordonnancement de la fabrication dans les ateliers. Elles sont généralement classées selon deux dimensions : La première est basée sur la connaissance de la disponibilité des travaux qui peut être statique ou dynamique alors que la seconde, s'appuie sur le dynamisme de la méthode d'ordonnancement qui peut être prévisionnelle (off-line) ou temps réel (real-time). Une schématisation des modèles d'ordonnancement selon la nature de l'univers d'ordonnancement considéré est donnée par la Figure 1.6 [Noronha & Sarma 91].

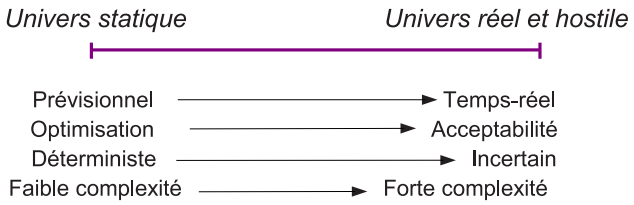

FIG. 1.6: Les modèles d'ordonnancement

Les méthodes d'ordonnancement prévisionnelles considèrent uniquement les problèmes d'ordonnancement statiques et utilisent des algorithmes pour définir le meilleur ordonnancement sans tenir compte des perturbations qui peuvent intervenir. Les problèmes d'ordonnancement prévisionnel sont des problèmes d'optimisation combinatoire. À côté des

algorithmes classiques exacts de résolution, des algorithmes approchés ont été développés. Un algorithme exact permet de garantir l'optimalité de la solution obtenue, à la différence d'un algorithme approché qui tente de s'en approcher le plus possible.

1.3.1 Les approches exactes

- *La programmation linéaire (PL)* regroupe l'ensemble des problèmes d'optimisation qui consistent à maximiser (ou minimiser) une fonction *objectif linéaire* de variables soumises à un ensemble de contraintes exprimées sous forme *d'équations ou d'inéquations linéaires*. On note que l'ensemble des solutions admissibles (qui respectent les différentes contraintes du problème) forment dans ce cas une enveloppe convexe nommé polytope et que l'ensemble des sommets du polytope constitue l'ensemble dominant de solutions parmi toutes les solutions admissibles.
  L'algorithme du simplexe développé en 1947 par Dantzig permet de résoudre la plupart des problèmes de la PL. Ce dernier étant très efficace la plupart du temps, il existe cependant des exemples pour lesquels il est montré que l'algorithme du simplexe n'est pas polynomial [Klee & Minty 72]. Des algorithmes polynomiaux de résolution de PL ont été développés par la suite. Khachian propose en 1979 une méthode appelée *méthode ellippsoïde* jugée lente pour des problèmes réels, mais montre toutefois que les problèmes de programmation linéaire sont polynomiaux [Khachian 79]. Une seconde méthode de *points intérieurs* basée sur le principe de géométrie projective et de programmation linéaire est proposée dans [Karmarkar 84]. Malgré que cette dernière méthode soit efficace pour les problèmes de grande taille, le simplexe demeure toutefois une méthode avantageuse servant de base pour d'autres approches de résolution, e.g. "les approches de décomposition".

- *La programmation linéaire en nombres entiers* : Lorsqu'un problème d'optimisation peut être modélisé comme un programme linéaire dont les variables sont contraintes à être entières alors le programme est dit linéaire en nombre entiers (PLNE). Quand certaines variables seulement sont contraintes à être entières, on parle alors de programme linéaire en variables mixtes (PLNE mixte).
  La PLNE est souvent difficile à résoudre du fait que l'espace de recherche n'est plus convexe mais discret. Il existe principalement deux approches de recherche de la solution optimale entière des PLNE : Les méthodes arborescentes par *séparation et évaluation* (branch and bound ou le branch and cut) et les *méthodes de coupes*. La méthode par séparation et évaluation sera décrite plus bas. Les méthodes de coupes ont pour but de trouver le plus petit polytope contenant toutes les solutions entières du PLNE. En pratique, ces méthodes consistent à itérer le processus suivant jusqu'à obtenir une solution entière : résoudre la relaxation linéaire du problème (relâcher les contraintes d'intégrité), puis ajouter des contraintes supplémentaires au problème

(les *coupes*) pour réduire l'enveloppe des solutions sans éliminer des solutions entières admissibles. La difficulté principale de cette méthode réside dans le choix des coupes efficaces pour une convergence rapide de l'algorithme. Dans [Johnson *et al.* 00], les auteurs mettent en avant les méthodologies les plus utilisées ces dernières années pour la resolution des problèmes pratiques de taille importante en PLNE-mixte.

- *La méthode par séparation et évaluation (PSE)* : C'est une méthode de recherche arborescente qui énumère à travers des nœuds des sous-ensembles de solutions. Le principe est d'éliminer de l'arbre de recherche les branches prouvées non-optimales afin d'éviter l'énumération exhaustive des solutions. Pour chaque sous-ensemble, une borne inférieure et une borne supérieure du critère sont calculées. Si à un nœud donné, la borne inférieure est supérieure à la meilleure borne supérieure trouvée jusqu'ici (pour un problème de minimisation), alors il est certain qu'il n'existe pas de solutions meilleure dans ce sous-ensemble issu du nœud (le branchement s'arrête au niveau du nœud). Les méthodes de recherche arborescente sont généralement des méthodes exponentielles.

- *Relaxation lagrangienne* : C'est une méthodologie générale qui permet d'obtenir des bornes inférieures de bonne qualité pour certains problèmes d'optimisation combinatoire. L'idée de la technique consiste à supprimer (relaxer) une partie des contraintes (en principe celles qui rendent le problème difficile) en les introduisant dans la fonction objectif sous forme d'une pénalité (combinaison linéaire des contraintes relaxées) si les contraintes relaxées ne sont pas respectées. Les coefficients de cette combinaison sont appelés multiplicateurs de Lagrange, et la nouvelle fonction objectif est appelée le Lagrangien du problème. La résolution du nouveau problème se fait généralement en utilisant la méthode du sous-gradient ou bien la méthode de génération de colonnes.

- *Génération de colonnes* : Dans la pratique, de nombreux problèmes d'optimisation combinatoire sont complexes et de grande taille. Ils possèdent un grand nombre de variables (colonnes). Ceci empêche donc leur résolution par les logiciels, disponibles aujourd'hui, de résolution des programmes linéaires. Pour pouvoir les traiter, la méthode de génération de colonnes est utilisée. Cette méthode repose sur la décomposition du modèle original par la méthode de *Dantzig et Wolfe* [Dantzig & Wolfe 60]. L'idée principale consiste à décomposer le modèle d'origine en un ou plusieurs sous-problèmes plus faciles à résoudre, l'ensemble étant généralement coordonné par un programme linéaire appelé *maître*. Le principe de l'algorithme de génération de colonnes consiste à résoudre un problème original (P) avec un sous-ensemble de colonnes (variables) de taille réduite, puis à l'alimenter itérativement avec de nouvelles colonnes susceptibles d'améliorer la solution courante, jusqu'à atteindre l'optimalité.

Les méthodes utilisant la génération de colonnes rencontrent souvent des problèmes de convergence. En effet, la rapidité de la méthode dépend à la fois, du nombre d'itérations nécessaire jusqu'à preuve d'optimalité (i.e. de l'efficacité de l'algorithme générateur dans le choix de la colonne améliorante), et du temps de calcul consacré à chaque étape de résolution. Il existe dans la littérature des procédures d'accéleration de la convergence [Touati 08].

- *La programmation dynamique* La programmation dynamique n'est pas à proprement parler un algorithme mais un principe général applicable à de nombreux problèmes d'optimisation avec contraintes, possédant une certaine propriété dite de *décomposabilité* [Minoux 08]. En général, les problèmes se ramènent au choix d'une suite de décisions séquentielles pour un objectif donné. Le résultat définit la valeur optimale du critère. Pour obtenir la solution optimale, un parcours à rebours des décisions qui ont menées à la valeur optimale est effectué. Ces méthodes ont une complexité polynomiale ou exponentielle, selon le problème.

1.3.2 Les approches métaheuristiques

Longtemps, la recherche s'est orientée vers la proposition d'algorithmes exacts pour les problèmes de complexité polynomiale. Toutefois, pour tenter de résoudre les problème $\mathcal{NP}$-*difficiles*, plusieurs chercheurs se sont orientés vers la proposition de méthodes de résolution approchées, appelées aussi heuristiques, qui fournissent des solutions de bonnes qualités en un temps de calcul raisonnable. On note que la qualité d'une heuristique est évaluée à la base de plusieurs jeux d'instances, en déterminant la déviation moyenne du critère par rapport à sa valeur optimale, et le temps de calcul en fonction de la taille du problème. On distingue plusieurs types d'heuristiques. Les heuristiques de construction qui génèrent une solution au problème sans remise en cause des décisions, les heuristiques de voisinage qui tentent d'améliorer des solutions initiales dans leur voisinage par des techniques de recherche locale, et les heuristiques à base de population de solutions qui explorent un espace de recherche de solutions très vaste.

Notons qu'une métaheuristique est un cadre général des heuristiques qui, à l'inverse de ces dernières, repose sur des mécanismes généraux "génériques" indépendant de tout problème. Parmi les heuristiques les plus généralement citées dans la littérature, on trouve :

- Les algorithmes *gloutons*. Ce sont des méthodes constructives où l'on construit progressivement une solution locale pour un problème d'optimisation, avec l'espoir d'atteindre la solution globale optimale. Si l'algorithme fournit systématiquement une solution optimale, il constitue une méthode exacte, dans le cas contraire, il est appelé une heuristique gloutonne. En ordonnancement, les algorithmes gloutons sont

des méthodes de construction par règles de priorité simples de type SPT, EDD, etc.

- Les méthodes *de recherche locale* ou métaheuristiques à base de voisinage. Ces méthodes partent d'une solution de départ unique, puis passent d'une solution à une autre voisine par déplacement successif, dans le but d'améliorer le critère. Parmi les méthodes de voisinage, on peut citer les *méthodes de descente* pour lesquelles une solution voisine doit obligatoirement améliorer le critère, i.e., s'arrête dès qu'il n'existe plus de meilleure solution dans le voisinage. On note toutefois, qu'afin de pouvoir s'échapper des minima locaux, d'autres approches autorisent de dégrader provisoirement le critère dans le but de pouvoir obtenir un optimum global. C'est le cas de la méthode du *recuit simulé* introduite en 1983 par Kirkpatrich et al. [Kirkpatrick *et al.* 83] inspirée du recuit physique en métallurgie. Son principe est simple. A chaque itération, une solution voisine est choisie aléatoirement, puis acceptée avec une dégradation du critère sous certaines conditions. Les mouvements non améliorants sont donc autorisés avec une certaine probabilité qui dépend de l'importance de la dégradation et de l'avancée de la procédure. Une autre méthode qui autorise la dégradation du critère est la *méthode tabou* [Glover 89] et [Glover 90]. Contrairement au recuit simulé qui génère aléatoirement une solution dans le voisinage de la solution courante, la méthode tabou explore le voisinage de la solution courante pour choisir la meilleure solution. Pour éviter de cycler, i.e revenir sur la solution déjà visitée, une liste *tabou* de solutions explorées est maintenue.
  Il existe beaucoup d'articles se référant au méthodes sus-citées de recherche locale. La recherche tabou en compte le plus grand nombre. Il existe toutefois d'autres méthodes de même type mais moins référencées dans la litérature, comme la recherche *à voisinage variable*, la méthode *iterated local search* ou bien *guided local search* [Sevaux 04].

- Les *métaheuristiques* à base de population, connues aussi sous l'appellation d'algorithmes évolutionnistes, sont des méthodes de recherche qui travaillent sur une population de solutions plutôt que sur une solution unique. Parmi ces méthode, on retrouve les *algorithmes génétiques* développés par Holland [Holland 75]. Ils sont basés sur la théorie de l'évolution des espèces dans leur milieu naturel énoncée par Darwin. L'idée consiste à combiner une population d'individus entre elles pour former de nouveaux individus enfants ayant une empreinte génétique nouvelle héritée des parents. Une fonction évaluation *fitness* mesure la force d'adaptation de chaque individu de la population. L'algorithme génétique s'appuie alors sur deux phases : la selection (favorisant les individus ayant un meilleur *fitness*), et la recombinaison (opérateurs de croisement et mutation) qui génère une nouvelle population d'individus enfants en conservant les "bonnes" caractéristiques de leurs parents. On note que la difficulté de la procédure réside dans la représentation ou le codage des solutions. Une autre méthode à base de population est l'algorithme de *colonies de fourmis*

introduit initialement par Dorigo et al. [Dorigo 92] pour la résolution du problème du voyageur de commerce, puis repris pour plusieurs autres domaines d'application. Cette métaheuristique reproduit le comportement auto-organisationnel des fourmis qui cherchent à atteindre depuis leur nid un objectif précis "une source de nourriture", analogue à la recherche du plus court chemin pour le problème du voyageur de commerce. Les fourmis ont la particularité de déposer une substance odorante appelée phéromone et marquent ainsi leurs trajets, ceci permettant à leur congénère de les suivre. Il est prouvé expérimentalement que ce comportement permet aux colonies de fourmis de choisir (sous certaines conditions) le plus court chemin vers la source à exploiter.

Après ce rappel sur quelques notions de bases relatives aux problèmes d'ordonnancement, nous discutons dans le chapitre qui suit, des limites des hypothèses classiques en ordonnancement.

Chapitre 2

Limites des hypothèses classiques en ordonnancement

Ce chapitre met l'accent sur le fait qu'il est irréaliste de supposer les paramètres du problème d'ordonnancement déterministes, et s'interesse plus particulièrement à la gestion des incertitudes en ordonnancement. Il décrit d'abord les sources d'incertitudes et le processus de résolution d'un ordonnancement dans un environnement dynamique. Les notions de flexibilité et de robustesse sont ensuite introduites. Enfin, ce chapitre met en évidence, les limites d'une résolution centralisée des problèmes d'ordonnancement et, l'intérêt d'adopter une organisation distribuée des décisions.

2.1 Brève Introduction aux approches de résolution robustes

Dans une discussion concernant les problèmes d'ordonnancement dans la pratique, Pinedo [Pinedo 08] fait le constat suivant :

> *" En pratique, l'environnement d'ordonnancement est généralement sujet à une quantité significative d'aléatoire ; en conséquence, il n'est pas intéressant de passer un temps énorme à déterminer une solution supposée optimale lorsque, en quelques heures à cause de quelque événement aléatoire, la structure du problème ou bien la liste des travaux aura changé."*

L'environnement d'application de l'ordonnancement est donc par nature dynamique et perturbé. De ce fait, la probabilité qu'un ordonnancement soit effectivement mis en œuvre comme prévu est faible.

Pourtant, la littérature scientifique compte un grand nombre de travaux de recherche qui se sont basés sur le développement d'algorithmes déterministes pour la résolution optimale ou sous-optimale de problèmes d'ordonnancement. Toutefois, ces études ne prennent

pas en compte la phase de mise en œuvre de l'ordonnancement dans son environnement réel.

Aujourd'hui, l'hypothèse du déterminisme des problèmes d'ordonnancement est jugée restrictive, et la problématique de l'ordonnancement avec gestion des incertitudes est posée et intéresse plusieurs chercheurs. Le chapitre suivant est consacré à cette problématique.

2.2 Déterminisme des paramètres

Les algorithmes polynomiaux ou approchés cités dans le chapitre précédent sont des algorithmes déterministes utilisés en phase hors-ligne. Ils considèrent toutes les données du problème connues a priori (problème déterministe). Leur résolution permet de produire un ou plusieurs ordonnancements prévisionnels. La phase d'exécution de la solution d'ordonnancement prévisionnelle se fait en ligne. On distingue ainsi le temps de résolution du problème d'ordonnancement du temps d'exécution de la solution trouvée. L'échelle de temps associée à l'exécution qui peut être très variable suivant l'application, n'a aucune influence sur celle de la résolution, qui est celle des moyens de calcul [Sanlaville 05].

En pratique, les données du problème ne sont pas toujours connues à l'avance. De plus, des informations nouvelles apparaissent durant l'exécution du programme prévisionnel susceptible de rendre sa mise en œvre caduque. Des algorithmes d'ordonnancements dynamiques sont alors utilisés en-ligne pour adapter progressivement la solution aux perturbations qui surgissent. En ordonnancement en-ligne, et contrairement à l'ordonnancement déterministe, les traitements pour la résolution du problème sont effectués durant la phase d'exécution.

L'environnement de l'ordonnancement étant par essence dynamique et incertain, la littérature scientifique a connu ces dernières années une émergence de la problématique d'ordonnancement avec prise en compte des perturbations. Une synthèse de ces approches sera donnée dans la partie III de ce manuscrit. Dans ce qui suit, nous définissons les termes élémentaires nécessaire à l'appréhension de cette problématique. Les notions de flexibilité, de robustesse et de modèles d'incertitudes, propres aux méthodes d'ordonnancement sous incertitudes, sont ainsi présentées.

2.3 La gestion des incertitudes en ordonnancement

Dans un environnement industriel, la probabilité qu'un ordonnancement prévisionnel soit exécuté comme prévu est faible [Davenport & Beck 00]. Plusieurs types d'incertitudes modifient et perturbent l'état d'un système de production, et presque toutes les données associées à un problème d'ordonnancement peuvent être affectées. Dans la littérature, une

distinction est faite sur les types de changement susceptibles d'intervenir. Les termes d'incertitudes, d'aléas et parfois de variations les désignent.

- **Incertitudes** : les données d'un problème d'ordonnancement sont soumises à des *incertitudes* si leurs valeurs ne sont pas connues à priori, ou si elles sont sujettes à des modifications, après le calcul de l'ordonnancement ou pendant son exécution. La donnée peut alors être associée à une variable aléatoire de loi connue, ou bien à un ensemble de valeurs possibles. C'est le cas, par exemple, d'une durée opératoire d'une tâche variant durant l'exécution de l'ordonnancement.

- **Aléa** : désigne les types particuliers d'incertitudes "événements" qui modifient les données. La probabilité d'un aléa peut ainsi être connue ou pas. Les cas d'une machine tombant en panne ou une tâche devenant soudainement prioritaire, sont des exemples d'aléas.

Parmi les incertitudes et les sources d'incertitudes rencontrées dans différents travaux [Davenport & Beck 00], [Kouvelis & Yu 97] et [Billaut *et al.* 05], nous citons :

- les durées opératoires réelles des tâches pouvant varier relativement à celles estimées ;
- une tâche pouvant être annulée ou pouvant arriver aléatoirement ; une variation pouvant aussi se produire sur la date de disponibilité ou la date d'échéance d'une opération ;
- des machines pouvant devenir non disponibles en raison de panne, de maintenance ou d'indisponibilité d'opérateur ou de matière première ;
- les coûts associés aux tâches pouvant être modifiés au moment d'exécution.

Il est évidemment possible d'imaginer d'autres sources d'incertitudes. Nous remarquons que dans tous les cas les modifications affectent les données du problème. Ces dernières sont ou bien incertaines ou partiellement connues pour le cas des données chiffrées (la durée opératoire, la date de disponibilité, etc), ou bien modifiées suite aux aléas qui surviennent en temps réel, c'est le cas des données structurelles (arrivée aléatoire, panne machine, etc.).

La prise en compte des incertitudes lors de la construction de l'ordonnancement dépend du niveau de connaissance de l'incertitude : connue, partiellement connue ou inconnue. Dans [Billaut *et al.* 05], plusieurs types de modèles sont distingués :

- le modèle aléatoire ou *stochastique* où les données du problème d'ordonnancement sont associées à des variables aléatoires et à des distributions de probabilité ;

- le modèle *par intervalles* où les paramètres du problème prennent leur valeur dans un ensemble continu de valeurs possibles. Les paramètres auxquels est associé ce modèle

sont généralement les dates de disponibilité ($r_j \in [\underline{r}_j, \overline{r}_j]$, les dates échues $[\underline{d}_j, \overline{d}_j]$ et les durées opératoires $[\underline{p}_j, \overline{p}_j]$;

- le modèle *par scénario* où les paramètres du problème P prennent valeur dans un ensemble discret de valeurs possibles $P \in \{P_1, P_2, ..., P_p\}$. Chaque ensemble P_i correspond à un scénario, c'est-à-dire à un problème d'ordonnancement déterministe distinct.

De manière générale, des schémas de résolution des problèmes d'ordonnancement sont proposés, couplant à la fois une partie hors ligne et une partie en-ligne pour tenir compte ou modéliser les perturbations. Les notions de flexibilité et de robustesse sont alors définies pour caractériser un système d'ordonnancement capable de gérer les incertitudes. Nous discutons de ces deux notions dans les sections suivantes.

2.4 Processus de résolution d'un problème d'ordonnancement avec incertitudes

De manière globale, la résolution pratique d'un problème d'ordonnancement dans un environnement dynamique et perturbé nécessite les étapes suivantes [GOThA 02] :

- **étape 0** : définition du problème statique avec spécification des incertitudes et leur modélisation ;

- **étape statique** : calcul d'un ensemble de solutions par un algorithme *statique* α. On parle aussi d'ensemble d'ordonnancements de référence ("baseline schedules") ;

- **étape dynamique** : lors de l'exécution, calcul d'une solution unique (effectivement mise en œuvre) par un algorithme *dynamique* δ.

À l'issue de la phase statique, un ensemble d'ordonnancements prévisionnels est défini par application de l'algorithme α. Partant de la solution prévisionnelle, l'ordonnancement dynamique doit choisir la solution qui sera réellement mise en exécution. Pour juger de la performance de la solution mise en œuvre, des mesures génériques de robustesse sont proposées pour évaluer si la solution choisie est la plus robuste.

Il est à noter que, lorsque la solution prévisionnelle proposée par l'algorithme statique est constituée d'un ensemble d'ordonnancements, la qualité de cette solution est forcément évaluée par un ensemble de valeurs possibles, et peut ainsi être caractérisée soit par une valeur moyenne, ou bien par les valeurs extrêmes, i.e. la qualité dans le meilleure des cas et la qualité dans le pire des cas [Aloulou *et al.* 02], [Briand *et al.* 03b] et [Briand *et al.* 03a].

De manière globale, dans les environnements d'application soumis à perturbation, les approches de résolution proposées dans la littérature pour l'ordonnancement se différencient par la nature des décisions prises durant les étapes de résolution du problème : statiques et dynamiques. Longtemps, les approches déterministes ont prédominé, supposant implicitement que l'ordonnancement statique peut être appliqué comme prévu. Il est aujourd'hui irréaliste d'ignorer l'aspect dynamique et incertain de l'environnement d'application. On signale toutefois, que l'ordonnancement stochastique a été proposé dès l'origine pour prendre les incertitudes en compte, même si la difficulté à mettre en œuvre de façon pratique une telle approche a limité son développement. Une voie nouvelle, l'ordonnancement robuste, s'est imposée ces dernières années et s'est posée en intermédiaire entre les approches purement déterministes et celles purement en-ligne [Sanlaville 05]. Une classification des méthodes d'ordonnancement robustes est décrite dans la Partie III. La section suivante introduit les notions qui permettent de prendre en compte les incertitudes en ordonnancement, à savoir la flexibilité et la robustesse.

2.5 Les mesures de performance

2.5.1 Flexibilité

Le terme flexibilité est défini dans [GOThA 02] comme la liberté dont on dispose durant la phase de mise en œuvre de l'ordonnancement. Elle est vue comme l'existence de modifications possibles au sein d'un ordonnancement statique (calculé hors-ligne) entraînant des pertes de performances restant acceptables ou nulles. Elle peut aussi être associée à l'existence d'une famille d'ordonnancements pouvant être acceptée à l'exécution, sans privilégier un ordonnancement en particulier.

On trouve toutefois d'autres définitions relatives aux ordonnancements flexibles. Herroelen et Leus, dans [Herroelen & Leus 04b] définissent un ordonnancement flexible comme étant un ordonnancement ayant la capacité d'être réparé facilement, i.e. modifiable en un nouveau ordonnancement de haute qualité.

Deux formes de flexibilité ont aussi été mises en évidence suivant le contexte de l'étude : statique et dynamique [Pujo & Brun-Picard 02].

La flexibilité *statique* d'une solution d'un problème d'ordonnancement réside dans la capacité des contraintes manipulées à autoriser plusieurs stratégies de décision pouvant être acceptées à l'exécution [Pujo & Brun-Picard 02]. On distingue alors dans ce cas plusieurs types de flexibilité [Billaut *et al.* 05] :

- *la flexibilité sur le temps* ou flexibilité temporelle, existe lorsque les dates de début et de fin des opérations peuvent varier. Une telle flexibilité autorise d'avancer ou de

retarder certaines opérations si les conditions le permettent. Elle est souvent présente de manière implicite et est induite par la plupart des autres formes de flexibilité en ordonnancement ;

- *la flexibilité sur les ordres* ou flexibilité séquentielle autorise des modifications de l'ordonnancement en cours d'exécution, et ce au niveau de l'ordre de passage des opérations (séquences) sur chaque ressource. L'introduction de cette flexibilité impose de manipuler non pas un seul ordonnancement, mais une famille d'ordonnancements possibles. De ce fait, il est plus difficile, en évitant la combinatoire liée à une énumération exhaustive et systématique des solutions, de déterminer la performance d'une famille d'ordonnancements, ces derniers ne possédant pas tous une performance identique. Notons que cette flexibilité suppose toutefois une présence plus importante de flexibilité temporelle ;

- la *flexibilité sur les ressources* ou flexibilité sur les affectations. Les ressources existent dans ce cas en plusieurs exemplaires. Cette forme de flexibilité est présente lorsqu'on autorise l'exécution d'une tâche sur une autre ressource que celle prévue initialement. Elle est nécessaire et est de grande utilité lorsque, par exemple, une machine tombe en panne. Elle suppose implicitement la présence d'une flexibilité séquentielle et temporelle ;

- la *flexibilité sur les modes d'exécution* existe lorsqu'il est possible de changer le mode d'exécution des opérations (préemption autorisée, changement de gammes, etc.) pour pallier une situation difficile, au prix d'une dégradation de la performance. Cette flexibilité requiert de la flexibilité séquentielle et temporelle, et dans certains cas de la flexibilité sur les affectations ;

- la *flexibilité de l'environnement de l'entreprise*. Il s'agit par exemple de la possibilité de négocier des délais avec des clients, ou de faire appel a un sous-traitant pour accroître temporairement la capacité de travail.

Nous rappelons que les formes de flexibilité citées précédemment sont de type statique. La flexibilité dynamique (décisionnelle) d'un ordonnancement correspond à la capacité d'un algorithme à adapter l'ordonnancement obtenu hors-ligne, lorsque les contraintes fluctuent (suite à des aléas), de manière à satisfaire les objectifs qui lui sont assignés, en exploitant au mieux la flexibilité statique disponible.

Pour tout type de flexibilité, un outil quantitatif de mesure, appelé indicateur de flexibilité est défini pour traduire le niveau de flexibilité obtenu. On peut ainsi considérer la cardinalité de la famille d'ordonnancement proposé statiquement comme une mesure de flexibilité. D'autres approches proposent de mesurer a posteriori la flexibilité proposée en

comparant la qualité de la solution flexible avec celle de la solution non flexible.

Le terme robustesse apparaît de plus en plus en ordonnancement. La notion de robustesse est étroitement liée à la flexibilité. Pour un ensemble de perturbations que l'on caractérise à priori, dégager de la flexibilité peut permettre de faire face à ces perturbations, et donc d'être robuste. Nous discutons de cette notion de robustesse dans ce qui suit.

2.5.2 Robustesse

Il existe dans la littérature plusieurs définitions de la robustesse en ordonnancement, en voici quelques unes :

- *« La robustesse est vue comme la capacité d'un centre à produire des décisions compatibles avec les contraintes des centres inférieurs »* [Pujo & Brun-Picard 02].

- *« La robustesse caractérise les performances d'un algorithme (ou plutôt d'un processus complet de construction d'un ordonnancement) en présence d'aléas »* [GOThA 02].

- *« La robustesse d'une solution c'est son insensibilité aux changements des paramètres »* [Hall & Posner 04].

- *« Le paradigme classique en programmation mathématique est de développer un modèle qui suppose que les données d'entrée sont connues et égales à leur valeur nominale. Cependant, cette approche ne prend pas en compte l'effet des incertitudes sur ces données qui influence la qualité et la faisabilité du modèle. Il est en effet possible que les valeurs des données diffèrent de leur valeur nominale, induisant ainsi le viol de plusieurs contraintes, et rendant la solution optimale trouvée à partir des valeurs nominales, de mauvaise qualité, voire infaisable. Cette constatation montre l'intérêt de concevoir des approches de résolution qui soient insensibles aux incertitudes, c'est-à-dire robustes. »* [Bertsimas & Sim 04].

- *« Un ordonnancement robuste est un ordonnancement dont la violation de ses hypothèses ont une conséquence moindre ou nulle sur cet ordonnancement »* [Le Pape 91].

- *« Un ordonnancement robuste est un ordonnancement qui a de fortes chances de rester valide même en subissant de grand nombre de perturbations »* [Leon *et al.* 94].

- *« Un ordonnancement robuste est un ordonnancement capable d'absorber une certaine quantité d'événements inattendus sans avoir à être ré-ordonnancé »* [Davenport *et al.* 01].

- *« Un ordonnancement est robuste si sa performance est peu sensible à l'incertitude des données et aux aléas [...] La robustesse d'un ordonnancement caractérise donc cette performance »* [Billaut *et al.* 05]

De manière générale, le terme *robuste* est employé pour qualifier un ordonnancement dont les performances sont acceptables vis-à-vis d'un ensemble d'incertitudes identifié a priori. À travers l'étude de la robustesse d'un processus global d'ordonnancement, on recherche des garanties de performances sur l'ordonnancement réalisé en présence d'aléas. Par *performance*, il est sous entendu, la qualité d'un ordonnancement (valeur de la fonction objectif) et aussi, d'autres mesures envisageables comme la stabilité de l'ordonnancement (solution robustness) ou la stabilité par rapport au critère de performance (quality robustness).

Il est important de pouvoir mesurer la robustesse afin de pouvoir déterminer quelle solution est plus robuste qu'une autre. Plusieurs outils quantitatifs de mesure de robustesse sont alors proposés. L'objectif étant de minimiser la mesure de déviation entre les performances prédites et les performances réalisées. Lorsque la mesure de robustesse est minimale, l'ordonnancement est dit robuste. De nombreux exemples de métriques de robustesse sont proposés dans [GOThA 02], [Billaut *et al.* 05] et [Sanlaville 05] .

Ainsi, la robustesse de la qualité (quality robustness) associée à une solution est généralement utilisée quand la performance de l'ordonnancement est mesurée par la valeur de la fonction objectif [Sevaux & Sörensen 02]. La mesure de cette robustesse est exprimée dans ce cas par l'écart entre la valeur du critère d'optimisation Z donné par l'ordonnancement de référence S_0 et l'ordonnancement réellement mis en œuvre S pour un scénario I donné. Dans le cas où les données sont modélisées de manière stochastique, on cherche alors à minimiser l'espérance de cette distance.

La robustesse de la solution (solution robustness) mesure la stabilité de l'ordonnancement. Elle ne tient pas compte du critère de performance [Sanlaville 05]. Cette mesure cherche dans ce cas à minimiser soit, le plus grand écart (noté d) entre deux solutions S_I et S'_I pour deux instances I et I' de P (P étant le problème statique avec description des incertitudes) donné par $R = \max_{I,I' \in P} d(S_I, S'_I)$, ou bien à minimiser le plus grand écart par rapport à un ordonnancement de référence $\tilde{S}$ donné par $R' = \max_{I \in P} d(S_I, \tilde{S}'_I)$. On cherche en fait à construire le plus petit ensemble possible d'ordonnancements compatibles avec l'incertitude prise en compte.

Plusieurs auteurs [Bertsimas & Sim 04], [Esswein 03] et [Briand *et al.* 03b] mettent en évidence que la robustesse a un prix. En effet, plus la solution est robuste (insensible aux incertitudes) et plus la performance est mauvaise, et inversement. De ce fait, d'autres

approches de résolution bicritère se sont orientées vers le couplage des indicateurs de robustesse et de performance pour assurer un compromis entre ces indicateurs [Esswein 03] et [Artigues *et al.* 05].

Nous avons dans cette section introduit les notions essentielles (flexibilité et robustesse) liées à la problématique d'ordonnancement sous incertitudes. La suite donne un aperçu sur le processus pratique de résolution d'un problème d'ordonnancement sous incertitudes.

2.6 Résolution centralisée/distribuée des problèmes

Comme évoqué précédemment dans la Section 2.2, dans un contexte réel d'application, il est souvent préférable d'opter pour le choix d'une solution non optimale demeurant faisable face à des perturbations, au lieu d'une solution optimale qui devient rapidement obsolète lors de l'exécution de l'ordonnancement.

Globalement, les systèmes d'ordonnancement les plus répandus s'appuient sur une politique centralisée de résolution (en-ligne ou hors-ligne) des problèmes d'ordonnancement avec ou sans prise en compte d'incertitudes. En effet, le processus de résolution des problèmes d'ordonnancement est classiquement assimilé à un problème de décision global car la fonction ordonnancement gère l'organisation de la totalité des ressources, et requiert une connaissance globale des paramètres du système. Cette organisation centralisée des décisions suppose alors l'existence d'une entité qui supervise les activités des ressources, et prend des décisions globales, celles-ci devant être respectées par l'ensemble des ressources qu'elle gère. Ce respect permet alors d'assurer la cohérence globale. La Figure 2.1 illustre une organisation de ce type, à la fois centralisée et hiérarchique.

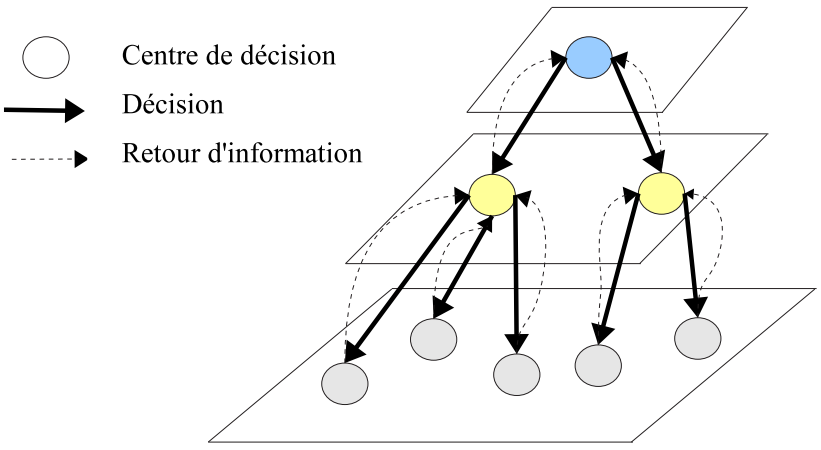

FIG. 2.1: Une organisation centralisée et hiérarchique des décisions

L'avantage d'une organisation centralisée réside dans la cohérence globale des décisions. Son inconvénient est qu'elle exige une transparence totale des différents acteurs, et n'autorise pas de remise en cause des décisions prises globalement. Une telle organisation est connue pour être rigide. De plus, supposer qu'une entité dispose d'une vision globale de tout le système est parfois non réaliste.

Dans de nombreux champs d'application (chaînes logistiques, projets industriels, etc.), les ressources exécutantes sont souvent réparties au sein d'un ensemble d'acteurs se trouvant parfois en situation de concurrence et disposant de fait, de leur propre autonomie de décision et d'objectifs propres dont il est fondamentale de tenir compte. Ce type d'organisation constitue précisément le contexte où les méthodes de décision robustes peuvent être profitables dans la mesure où chaque acteur doit faire face non seulement aux perturbations pouvant surgir de sa propre organisation locale, mais aussi et surtout à celles qui peuvent provenir de son environnement, constitué des autres acteurs.

Il est alors intéressant de considérer l'ordonnancement sous l'hypothèse d'une fonction distribuée (décisions d'ordonnancement distribuées entre les différents acteurs) où la solution globale résulte d'une négociation entre les différents acteurs, chacun gérant son propre ordonnancement local robuste. Le résultat de la coopération devant converger vers un compromis satisfaisant les performances locales ainsi que la performance globale.

Le progrès technologique de ces dernières années en matière de systèmes de calculs et de réseaux de communications a permis de modifier le mode d'organisation classique des systèmes de gestion de production et gestion de projet. Des approches de résolution distribuées sont développées permettant de répartir les calculs d'ordonnancement sur plusieurs acteurs autonomes afin que ces derniers, puissent profiter de leur autonomie et mieux gérer leur intérêts et confidentialié. Les décisions d'ordonnancement étant réparties, ces approches requièrent toutefois des méthodes spécifiques de gestion pour assurer la cohérence globale des différents choix.

Dans le cadre des chaînes logistiques, plusieurs travaux de recherches font l'hypothèse que les décisions sont distribuées au sein d'un réseau de centres de décision interdépendants, et proposent des mécanismes d'aide à la décision pour la coopération. Le secteur industriel compte aussi de plus en plus de logiciels de gestion de chaînes logistiques offrant des solutions basées sur différents aspects d'organisation coopérative.

Toutefois, peu de travaux de recherche se sont intéressés à l'ordonnancement distribué et coopératif. La majorité des travaux proposent des approches basées sur les Systèmes Multi-Agents (SMA), celles-ci étant appliquées aux systèmes flexibles de production (FMS : flexible Manufacturing System).

2.7 Synthèse

Dans ce chapitre, nous avons vu d'une part, qu'une classification des stratégies employées pour l'ordonnancement est fondée sur une analyse de la conception des algorithmes d'ordonnancement : statiques ou dynamiques. En effet, de nombreux travaux de recherche ont été développés pour générer un ordonnancement prévisionnel avec une performance optimale ou proche de l'optimum. Seulement, l'environnement d'application est dynamique et nombreuses sont les perturbations pouvant survenir lors de l'exécution. L'ordonnancement prévisionnel ne peut donc être exécuté comme prévu et la réalisation (e.g. d'autres travaux) est perturbée entraînant ainsi une forte dégradation des performances. Pour pallier ce problème, la littérature scientifique a connu ces dix dernières années une émergence d'approches de résolution des problèmes d'ordonnancement sous incertitudes, domaine de recherche communément appelé *ordonnancement robuste*.

D'autre part, une autre distinction des politiques d'ordonnancement peut être basée selon la nature de la politique choisie : centralisée ou distribuée. Nous avons mis en évidence l'intérêt de la distribution des décisions, ceci pour accorder plus d'autonomie aux différents acteurs afin de réagir aux aléas aussi bien internes qu'externes. La démarche adoptée dans ce cas pour la résolution du problème est l'approche coopérative, visant à apporter des décisions locales cohérentes vis-à-vis de l'organisation globale.

Ce sont ces points qui nous ont motivés pour orienter nos travaux et les inscrire dans le cadre de la distribution des décisions et la coopération pour l'ordonnancement en présence de perturbations. Ainsi, nous nous intéressons, dans la Partie III, au problème d'ordonnancement d'un atelier de type job-shop évoluant dans un contexte perturbé. L'idée de base de l'approche d'ordonnancement robuste et coopératif consiste à supposer que chaque ressource est assimilée à un centre de décision gérant son propre ordonnancement locale et robuste, et disposant de ce fait d'une autonomie décisionnelle pour anticiper les perturbations ; les décisions d'ordonnancement résultent alors d'une négociation/renégociation entre couples de ressources, ces dernières étant liées par les gammes opératoires. À chaque occurrence d'un nouveau événement, la coopération conduit à la caractérisation, sur chaque centre de décision, d'une famille de solutions (ordonnancement robuste) tout en assurant une cohérence globale des décisions individuelles.

Notons que le point central de cette thèse est l'exploitation d'une condition de dominance établie dans les années quatre-vingt par Erschler et al. [Erschler *et al.* 83]. Cette condition, donnée par le théorème des pyramide, définit un ordre partiel entre les opérations. Son intérêt est qu'elle offre de la flexibilité séquentielle. En effet, dans le contexte de recherche de robustesse, i.e., l'ordonnancement robuste à une machine, ce théorème permet de caractériser, au lieu d'un unique ordonnancement, un ensemble d'ordonnancements de référence qui sont proposés aux décideurs au moment de la prise de décision d'ordonnancement et qui permettent d'anticiper les éventuelles perturbations. De plus, une propriété

intéressante de la condition de dominance réside dans la possibilité d'évaluer les performances au mieux et au pire de l'ensemble caractérisé, en un temps polynomial [La 04].

Toujours dans l'objectif d'exploiter la condition de dominance, nous nous sommes penchés, dans une autre partie (Partie II) du manuscrit, à son étude pour la résolution du problème d'ordonnancement à une machine. Nous verrons que de nouvelles extensions de cette condition sont proposées. De plus, nous étudions la dominance de cette condition vis-à-vis du problème de minimisation du nombre de travaux en retard. A la base des résultats trouvés, des modèles de programmation linéaire originaux sont proposés.

Deuxième partie

Problème à une machine

Introduction

Les problèmes d'ordonnancement à une machine, d'apparence simples, sont dans la plupart des cas $\mathcal{NP}$-difficiles dès que des fenêtres d'exécution sont associées aux travaux. Ils sont fondamentalement très importants car souvent rencontrés en pratique. Leur modélisation, sert en effet de base et, permet la compréhension et la résolution des problèmes à plusieurs ressources plus complexes. En effet, les problèmes d'ordonnancement mettant en jeu des environnements à plusieurs machines peuvent être décomposés en sous-problèmes d'ordonnancement plus simples à une machine. L'exemple, souvent cité, est celui d'un problème d'atelier se réduisant au seul problème à une machine lorsqu'un poste de travail constitue un goulot d'étranglement [Pinedo 08].

Tout au long de cette seconde partie du manuscrit, nous nous focalisons sur l'étude du problème général à une machine. Plusieurs critères sont considérés, à savoir, l'admissibilité, le retard algébrique maximal ainsi que le nombre de travaux en retard. Ces critères sont utilisés en pratique comme indicateurs de performance.

Après une introduction, nous commençons par présenter, dans le Chapitre 3, un théorème de dominance (Théorème des pyramides) démontré dans les années quatre-vingt pour le problème à une machine. Nous introduisons la notion de séquence maître-pyramide à partir de laquelle peut être extrait l'ensemble de séquences dominantes que ce théorème caractérise, et montrons comment déduire l'ensemble des ordres totaux (ensemble de problèmes à structure pyramidales distinctes) ayant un même ensemble dominant de séquences [Ourari *et al.* 07]. Puis, en se basant sur la condition nécessaire et suffisante d'admissibilité, sur laquelle est fondée le Théorème des pyramides, nous énonçons de nouvelles conditions numériques de dominance permettant d'identifier, si elle existe, une séquence admissible parmi toutes les séquences dominantes. Enfin, nous nous intéressons à l'étude de la condition de dominance pour le cas du problème de minimisation du nombre de travaux en retard [Briand *et al.* 06b] [Briand *et al.* 07]. Nous montrons sous quelles hypothèses et de quelle manière le Théorème des pyramides peut être dominant vis-à-vis du critère $\sum U_i$ [Ourari & Briand 08].

Dans un second chapitre (le Chapitre 4) et sur la base des résultats numériques démontrés dans le premier chapitre, nous proposons de nouvelles formulations mathématiques

sous forme de programmes linéaires en nombres entiers pour la modélisation du problème à une machine. Considérant tout d'abord le problème d'admissibilité, un premier modèle de programmation linéaire pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité est proposé. Ensuite, nous démontrons l'équivalence du modèle formulé au cas de résolution du problème de minimisation du plus grand retard algébrique $1|r_i|L_{max}$. L'efficacité du modèle est prouvée, dans certains cas, à travers des expérimentations et une comparaison avec la procédure de Carlier [Briand & Ourari 09] [Briand *et al.* 10a]. Enfin, nous nous intéressons au problème de minimisation du nombre de travaux en retard $1|r_i|\sum U_i$. Nous montrons comment adapter le modèle de programmation linéaire établi pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité, pour calculer des bornes inférieure et supérieure de bonne qualité pour le problème $1|r_i|\sum U_i$ [Ourari *et al.* 09b].

Chapitre 3

Structure d'intervalles et dominance

Dans ce chapitre, une condition analytique de dominance établie dans les années 80 pour la recherche de solutions admissibles est rappelée, puis, la notion de séquence maître-pyramide, permettant d'extraire les séquences dominantes est introduite. Des extensions de la condition analytique ainsi que de nouvelles conditions numériques de dominance sont ensuite présentées. Considérant le problème d'ordonnancement à une machine, $1|r_j|\sum U_j$, il est enfin montré sous quelles conditions, la séquence maître-pyramide est dominante vis-à-vis du problème de minimisation du nombre de travaux en retard.

3.1 Définitions

En ordonnancement, la définition d'ordres partiels entre les tâches permet d'une part, de traduire des conditions nécessaires, suffisantes ou dominantes vis-à-vis de l'admissibilité d'un problème ou de l'optimisation d'un critère, et d'autre part, de caractériser des ensembles flexibles de solutions en évitant l'explosion combinatoire liée à leur énumération.

Par définition [Cheng *et al.* 02], un ordre partiel P est caractérisé par une paire $P = (X, \leq_P)$, X étant un ensemble d'éléments, où chaque relation $\leq_P$ sur $X \times X$ est réflexive, antisymétrique et transitive. Pour $u, v \in X$, $u \leq_p v$ est interprété comme u est plus petit ou égal à v. Un ordre partiel $P = (X, \leq_p)$ est un *ordre total* si pour tout couple $(u, v) \in X \times X$ la relation $u \leq_p v$ ou la relation $v \leq_p u$ est vérifiée.
Étant donnés deux ordres partiels $P = (X, \leq_P)$ et $Q = (X, \leq_Q)$ sur le même ensemble X, Q est une *extension* de P (ou P est un *sous-ordre* de Q) si $u <_P v$ implique $u <_Q v$ pour tout couple $(u, v) \in X$. Un ordre linéaire $Q = (X, \leq_Q)$ est une *extension linéaire* d'un ordre partiel $P = (X, \leq_P)$ si Q étend P.

Pour un ordre partiel donné, il est parfois possible de calculer rapidement le nombre de solutions caractérisées, et de borner la qualité au mieux et au pire de l'ensemble flexible caractérisé et de mesurer sa flexibilité. Cette propriété est particulièrement intéressante

puisque, savoir déterminer une performance au mieux et / ou au pire, en un temps de calcul acceptable, peut s'avérer appréciable du point de vue de l'optimisation, notamment au sein de procédures de séparation et évaluation progressive. De plus, certains ordres sont relativement insensibles aux variations des données du problème, propriété intéressante dans le contexte de l'ordonnancement robuste.

Comme énoncé précédemment, la recherche de conditions nécessaires, suffisantes ou dominantes d'admissibilité ou d'optimalité peut se traduire par la définition d'ordre partiel nécessaire, suffisant ou dominant. Dans le cadre de l'admissibilité, la recherche de conditions nécessaires et suffisantes par l'*analyse sous contraintes* du problème vise à déterminer *toutes* les solutions satisfaisant les contraintes du problème. Cette recherche est théoriquement possible, mais pose un problème de complexité [Erschler 76]. Elle a de ce fait donné lieu à deux problématiques de recherche distinctes, l'une concernant des conditions d'admissibilité suffisantes, l'autre des conditions d'admissibilité nécessaires.

L'utilisation de conditions suffisantes d'admissibilité ou d'optimalité en ordonnancement est généralement limitée à des problèmes d'ordonnancement simples à une ou deux ressources. L'ordre *partiel suffisant* caractérise ainsi un sous-ensemble de l'ensemble des ordonnancements admissibles (resp. optimaux relativement à un critère donné). La célèbre *règle de Johnson* [Johnson 54] utilisée dans le cadre du problème flow shop de permutation à deux machines est une condition suffisante d'optimalité. D'autres cas sont cités dans [Esquirol & Lopez 99] ou [Pinedo 08].

Les conditions nécessaires d'admissibilité sont principalement utilisées dans les approches d'ordonnancement par contraintes et permettent d'imposer des ordres partiels *nécessaires* entre les tâches [Lopez 91]. Les ensembles de solutions satisfaisant l'ordre partiel nécessaire comprennent tous les ordonnancements admissibles (resp.optimaux), plus certains ordonnancements non admissibles (resp.non optimaux) [Briand *et al.* 05].

D'autres approches caractérisent des ordonnancements suivant des conditions *dominantes* vis-à-vis de l'optimalité ou de l'admissibilité. Pour un problème d'optimisation, si on utilise la notion d'ordre partiel, on dit qu'un ordre partiel P sur un ensemble de tâches d'un problème d'ordonnancement, est un *ordre partiel dominant* s'il existe une séquence optimale qui est une extension linéaire de cet ordre partiel P [Cheng *et al.* 02]. On dit aussi qu'un sous ensemble de solutions est dominant pour l'optimisation d'un critère donné, s'il contient au moins un optimum pour ce critère. De manière analogue, il est dit dominant par rapport à un ensemble de contraintes lorsque ce sous-ensemble contient au moins une solution admissible, s'il en existe [Esquirol & Lopez 99].

Dans la section suivante, la notion de structure d'intervalles et les concepts qui lui sont

rattachés sont décrits.

3.1.1 Structures d'intervalles, sommets, bases et pyramides

Une structure d'intervalles est définie par un couple $< I, C >$ où $I = \{i_1, \ldots, i_n\}$ est un ensemble d'intervalles et C est un ensemble de contraintes sur $I \times I$. Chaque intervalle i_j est défini par une date de début x_j et une date de fin y_j (cf. Figure 3.1). Une contrainte entre deux intervalles i_j et i_k peut être exprimée soit en spécifiant un ordre total entre les dates de début et de fin des deux intervalles, soit de façon équivalente, en utilisant une des relations de l'algèbre de Allen [Allen 91] récapitulées sur la Figure 3.1.

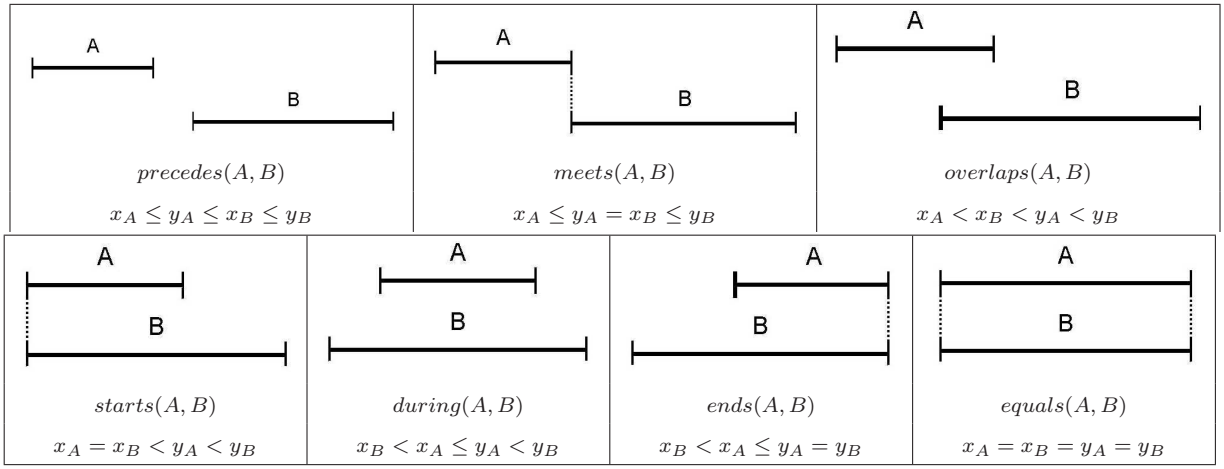

FIG. 3.1: Algèbre de Allen

Par considération des relations de Allen, deux types particuliers d'intervalles peuvent être mis en évidence : L'intervalle de type *sommet* et celui de type *base*.

Définition 3.1. *Un intervalle* t *est dit sommet d'une structure d'intervalles* $< I, C >$ *si* $\nexists\ i \in V$ *tel que* $during(i, t)$.

Définition 3.2. *Un intervalle* b *est dit base d'une structure d'intervalles* $< I, C >$ *si* $\nexists\ i \in I$ *tel que* $during(b, i)$.

En utilisant ces notions de sommet et de base, les notions de *sommet-pyramide* (noté *s-pyramide*) et de *base-pyramide* (noté *b-pyramide*) peuvent être définies [Esquirol & Lopez 99].

Définition 3.3. *Une t-pyramide* P_t, *associée au sommet* t *d'une structure d'intervalles* $< I, C >$, *est le sous-ensemble d'intervalles* $i \in I$ *tel que* $during(t, i)$.

Définition 3.4. *Une b-pyramide* P_b, *associée à la base* b *d'une structure d'intervalles* $< I, C >$, *est le sous-ensemble d'intervalles* $i \in I$ *tel que* $during(i, b)$.

On note que l'intérêt d'utiliser des structures d'intervalles réside dans le fait que cela permet une représentation intéressante des caractéristiques d'un problème. De plus, ces notions permettent de définir des ordres partiels. Ces derniers sont utilisés par différentes approches en ordonnancement pour la caractérisation de solutions. Nous reviendrons sur ce point après avoir présenté succinctement des notions de dominance en ordonnancement.

3.1.2 Notions de dominance en ordonnancement

En optimisation, le concept de dominance d'un sous ensemble de solutions permet de limiter la complexité algorithmique liée à la recherche de solution. Un sous ensemble, dominant pour l'optimisation d'un critère, est tel qu'il garantit l'existence d'au moins une solution optimale, vis-à-vis du critère considéré, dans les solutions qu'il contient. De manière analogue, il est dit dominant par rapport à un ensemble de contraintes s'il contient au moins une solution admissible s'il en existe [Esquirol & Lopez 99]. Un sous ensemble de solutions dominantes est caractérisé à l'aide de conditions de dominance.

Dans le cadre de l'étude de l'admissibilité, une séquence Seq_1 domine une séquence Seq_2 si et seulement si Seq_2 admissible implique Seq_1 admissible [Erschler *et al.* 83]. Les séquences dominantes peuvent être considérées comme potentiellement meilleures que les autres dans le sens où, si aucune n'est admissible, alors cela implique qu'il n'existe pas de solution admissible pour le problème considéré (certificat d'absence de solution, cf. Figure 4.1). On peut donc dire qu'une solution dominante est en quelque sorte "plus admissible" (ou "plus optimale") que d'autres [Fontan 80].

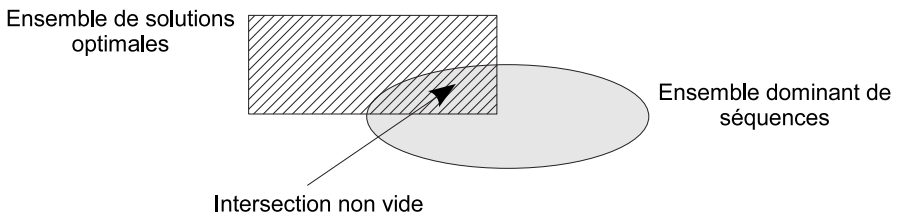

FIG. 3.2: Relation entre les ensembles de solutions dominantes et optimales

Dans le cadre de l'optimisation et pour certains critères, la définition de la dominance fait intervenir la notion de circuits dans des graphes conjonctifs de type $G(X, U)$, où X désigne l'ensemble des tâches associées au problème à ordonnancer et U, l'ensemble des arcs conjonctifs reliant deux tâches du problème [Roy 70]. Considérons deux séquences Seq_1 et Seq_2 décrites respectivement par les graphes conjonctifs $G_1(X, U_1)$ et $G_2(X, U_2)$. La séquence Seq_1 domine la séquence Seq_2 si et seulement tout circuit de $G_2(X, U_2)$ est de

longueur supérieure ou égale au plus long circuit de $G_1(X, U_1)$ [Fontan 80].

Les recherches visant la mise en évidence de conditions de dominance prennent classiquement en compte la nature particulière du problème étudié afin de déduire des propriétés. Ces conditions permettent de réduire l'espace de recherche, et conduisent ainsi à la conception de méthodes de résolution, elles mêmes très efficaces.

3.2 Une condition analytique de dominance

3.2.1 Problème considéré et condition de dominance

Dans les années quatre-vingt, considérant le problème d'ordonnancement à une machine avec fenêtres d'exécution, Erschler et al. [Erschler *et al.* 83] mettent en évidence une nouvelle condition de dominance pour la recherche de solutions admissibles. Un problème V, constitué d'un ensemble $T = \{1, \ldots, n\}$ de travaux à ordonnancer sur une machine, disponible en un seul exemplaire, est considéré. Chaque travail j de V est caractérisé par une date de disponibilité r_j, une date échue d_j et une durée opératoire p_j. Un ordonnancement est dit admissible si la machine n'exécute jamais plus de deux travaux au même instant, et si les travaux finissent à l'heure, i.e., si la condition (3.1) est satisfaite. Trouver une solution admissible pour ce problème est NP-difficile [Lenstra *et al.* 77].

$$\forall j \in V, \quad s_j \geq r_j \quad \text{et} \quad s_j + p_j \leq d_j. \tag{3.1}$$

Les dates de début et de fin des travaux, s_j et f_j, étant entièrement déterminées par la connaissance de la séquence des travaux sur la machine, il y a au plus $n!$ séquences à examiner pour trouver une solution admissible. La condition de dominance proposée dans [Erschler *et al.* 83] permet heureusement de limiter l'étude de l'admissibilité à un sous-ensemble de séquences beaucoup plus restreint.

La donnée considérée par les auteurs pour l'étude de la dominance est l'ordre relatif des dates de début au plus tôt r_j et de fin au plus tard d_j de l'ensemble T des n travaux. Les durées opératoires p_j ainsi que les valeurs explicites des r_j et d_j de chaque travail $j \in T$ ne sont donc pas considérées. La condition de dominance reste donc valide quelles que soient les durées opératoires et quelles que soient les valeurs de r_j et d_j dans le respect de l'ordre relatif retenu dans le corps d'hypothèses.

Dans [Erschler *et al.* 85] et [Briand *et al.* 03b], il est aussi montré que le sous-ensemble de séquences caractérisé par la condition de dominance, est également dominant vis-à-vis des critères réguliers d'optimisation, T_{max}, plus grand retard vrai et, L_{max}, plus grand retard algébrique.

3.2.2 Théorème des pyramides

Les notions de structure d'intervalles, de sommet et de t-pyramide définies dans la partie 3.1.1 sont utilisées. La structure d'intervalles considérée est $< I, C >$ où un intervalle $i_j \in I$, caractérisé par une date de début $x_j = r_j$ et une date de fin $y_j = d_j$ (i.e., $i_j = [r_j, d_j]$), est associé à chaque travail $j \in T$ du problème.

Les sommets sont supposés indicés dans l'ordre croissant de leurs r_j, ou en cas d'égalité, dans l'ordre croissant de leurs d_j. Ceci est équivalent à affirmer que, t_u et t_v étant deux sommets de la structure d'intervalles, alors $u < v$ si et seulement si $r_{t_u} \leq r_{t_v}$ et $d_{t_u} \leq d_{t_v}$. Si deux sommets possèdent des dates de début au plus tôt et de fin au plus tard identiques, alors ils peuvent être indicés de façon quelconque. La t-pyramide notée P_u désigne la pyramide caractérisée par le sommet t_u. Aussi, $u(j), v(j)$ désignent respectivement les indices de la première et de la dernière t-pyramide à laquelle peut être affecté le travail j. Un ordre partiel dominant peut alors être caractérisé grâce au théorème suivant.

Théorème 3.1. *Un ensemble dominant de séquences peut être constitué par les séquences telles que :*

1. *les sommets sont ordonnés dans l'ordre de leur, indice, ;*
2. *avant le premier sommet, seuls sont placés des travaux appartenant à la première t-pyramide, rangés dans l'ordre croissant de leurs r_j ou, en cas d'égalité, dans un ordre arbitraire ;*
3. *après le dernier sommet, seuls sont placés des travaux appartenant à la dernière s-pyramide, rangés dans l'ordre croissant de leurs d_j ou, en cas d'égalité, dans un ordre arbitraire ;*
4. *entre deux sommets t_k et t_{k+1}, sont placés en premier des travaux appartenant à la t-pyramide P_k et n'appartenant pas à P_{k+1} dans l'ordre croissant de leurs d_j (dans un ordre arbitraire en cas d'égalité), puis des travaux communs aux deux pyramides P_k et P_{k+1} dans un ordre arbitraire, et enfin des travaux appartenant à la pyramide P_{k+1} et n'appartenant pas à P_k dans l'ordre croissant de leurs r_j (dans un ordre arbitraire en cas d'égalité).*

Le Théorème des pyramides définit un ordre partiel dominant dans la mesure où il impose des relations de précédence entre les sommets d'une part, puisque $t_k \prec t_{k+1}$, et entre les travaux non-sommets et les sommets d'autre part, puisque $t_{u(j)-1} \prec j \prec t_{v(j)+1}$. Cependant, nous remarquons que le théorème, en imposant par ailleurs de classer les travaux affectés devant un sommet par ordre croissant des r_j, et ceux affectés derrière par ordre croissant des d_j, exclut aussi certains ordres totaux. Ainsi, si deux travaux non-sommets i et j, tels que $r_i < r_j$, appartenant à une seule pyramide de sommet t, sont tous deux séquencés avant t alors $i \prec j$, et l'ordre total $j \prec i \prec t$ est interdit. Ce théorème introduit donc, outre un ordre partiel dominant, des relations de dominance conditionnelles, liées à

la position des travaux relativement aux sommets, et à l'ordre relatif des dates de début au plus tôt et de fin au plus tard.

Une propriété intéressante du Théorème des pyramides est qu'il permet d'évaluer la cardinalité de l'ensemble de séquences dominantes S_{dom} qu'il caractérise grâce à la formule suivante : $card(S_{dom}) = \prod_{q=1}^{N}(q+1)^{n_q}$ où n_q désigne le nombre d'intervalles appartenant exactement à q pyramides et N le nombre total de pyramides.

L'ordre des travaux placés entre les sommets étant imposé par la considération des dates de début au plus tôt et de fin au plus tard (ordre croissant), la formule citée ci-dessus dénombre les affectations différentes des travaux selon qu'ils sont placés avant ou après un sommet s_k, tel que $u(j) \leq k \leq v(j)$.

3.2.3 Exemple illustratif

Pour illustrer le Théorème des pyramides, considérons un problème d'ordonnancement à 6 travaux dont l'ordre relatif entre les r_i et les d_i est tel que : $r_4 < r_5 < r_3 < r_1 < d_1 < d_3 < r_6 < d_4 < r_2 < d_2 < d_5 < d_6$. La structure d'intervalles associée à cet exemple est présentée sur la Figure 3.3. On distingue deux sommets $t_1 = 1$ et $t_2 = 2$ caractérisant les deux pyramides : $P_1 = \{3, 4, 5\}$, et $P_2 = \{5, 6\}$. Remarquons que, compte tenu de la définition d'une s-pyramide, les sommets n'appartiennent pas à la pyramide qu'ils caractérisent.

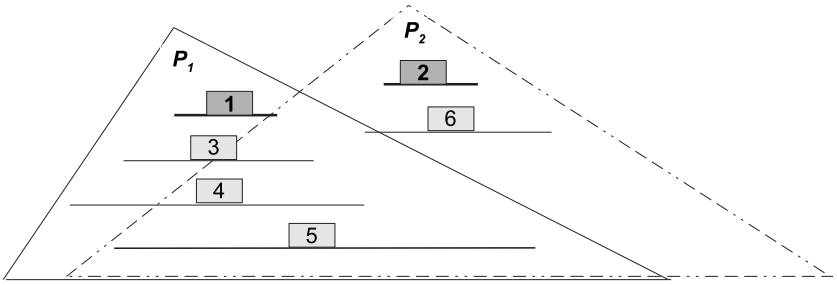

FIG. 3.3: Diagramme des intervalles, sommets et pyramides

L'application du théorème des pyramides à cette structure d'intervalles permet de caractériser un ensemble dominant S_{dom} de cardinalité $card(S_{dom}) = (1+1)^3.(2+1)^1 = 24$ (sur 6 !=720 cas possibles) constitué des séquences données sur la Figure 3.4.

Remarquons que cette formule ne prend pas en compte les possibilités de permutation des travaux placés entre deux sommets t_k et t_{k+1} et appartenant en même temps aux t-pyramides P_k et P_{k+1}. Par exemple, considérons le cas de deux travaux u, v et deux sommets t, t' ayant l'ordre relatif entre les r_i et d_i suivant : $r_u < r_v < r_t < r_{t'} < d_t < d_{t'} < d_i < d_j$.

$4 \prec 5 \prec 3 \prec 1 \prec 6 \prec 2$	$4 \prec 3 \prec 1 \prec 6 \prec 2 \prec 5$	$5 \prec 3 \prec 1 \prec 4 \prec 6 \prec 2$	$3 \prec 1 \prec 4 \prec 6 \prec 2 \prec 5$
$4 \prec 5 \prec 3 \prec 1 \prec 2 \prec 6$	$4 \prec 3 \prec 1 \prec 2 \prec 5 \prec 6$	$5 \prec 3 \prec 1 \prec 4 \prec 2 \prec 6$	$3 \prec 1 \prec 4 \prec 2 \prec 5 \prec 6$
$4 \prec 5 \prec 1 \prec 3 \prec 6 \prec 2$	$4 \prec 1 \prec 3 \prec 5 \prec 6 \prec 2$	$5 \prec 1 \prec 3 \prec 4 \prec 6 \prec 2$	$1 \prec 3 \prec 4 \prec 5 \prec 6 \prec 2$
$4 \prec 5 \prec 1 \prec 3 \prec 2 \prec 6$	$4 \prec 1 \prec 3 \prec 5 \prec 2 \prec 6$	$5 \prec 1 \prec 3 \prec 4 \prec 2 \prec 6$	$1 \prec 3 \prec 4 \prec 5 \prec 2 \prec 6$
$4 \prec 3 \prec 1 \prec 5 \prec 6 \prec 2$	$4 \prec 1 \prec 3 \prec 6 \prec 2 \prec 5$	$3 \prec 1 \prec 4 \prec 5 \prec 6 \prec 2$	$1 \prec 3 \prec 4 \prec 6 \prec 2 \prec 5$
$4 \prec 3 \prec 1 \prec 5 \prec 2 \prec 6$	$4 \prec 1 \prec 3 \prec 2 \prec 5 \prec 6$	$3 \prec 1 \prec 4 \prec 5 \prec 2 \prec 6$	$1 \prec 3 \prec 4 \prec 2 \prec 5 \prec 6$

FIG. 3.4: Ensemble dominant de séquences du problème de la Figure 3.3

Les deux travaux i et j appartiennent donc aux deux pyramides P_t et $P_{t'}$ et peuvent être placés dans n'importe quel ordre entre les deux sommets. Considérons la séquence dominante $t \prec u \prec v \prec t'$. Si cette séquence est comptabilisée dans S_{dom}, la séquence $t \prec v \prec u \prec t'$, bien qu'elle respecte le Théorème des pyramides, ne le sera pas. La valeur calculée par la formule $card(S_{dom})$ est donc une borne inférieure du nombre réel de séquences de l'ensemble dominant.

Enfin, il est proposé et démontré dans [La 04] une extension permettant de restreindre davantage la borne inférieure S_{dom} caractérisée grâce au Théorème des pyramides. Elle est donnée comme suit :

Extension 3.1. *Si un travail j appartient à deux pyramides P_1 et P_2 dont les sommets t_1 et t_2 possèdent des dates de début ou des dates de fin égales, alors toute séquence plaçant j entre t_1 et t_2 est dominée par au moins une autre séquence de l'ensemble dominant.*

3.3 Notion de séquence maître-pyramide

3.3.1 définition

Dans cette partie, et pour éviter d'énumérer toutes les séquences dominantes S_{dom} que le Théorème des pyramides caractérise, nous nous proposons de montrer qu'il est possible de définir une séquence particulière, que nous nommons *séquence maître-pyramide* et notons $\sigma_\Delta(V)$ dans la suite du texte. Cette séquence est de la forme :

$$\sigma_\Delta(V) = (\alpha_1, t_1, \beta_1, \ldots, \alpha_k, t_k, \beta_k, \ldots, \alpha_m, t_m, \beta_m) \quad \text{où :}$$

- m est le nombre total de sommets ;
- α_i est l'ensemble des travaux appartenant à la pyramide P_i de sommet t_i et n'appartenant pas à la pyramide P_{i-1} classés par ordre croissant de leur date de disponibilité ;
- β_i est l'ensemble des travaux appartenant à la pyramide P_i classés par ordre croissant de leur date échue.

Comme on peut facilement s'en assurer, toute séquence de n travaux compatible avec $\sigma_\Delta(V)$ respecte le Théorème des pyramides. L'ensemble des séquences de n travaux compatible avec $\sigma_\Delta(V)$ représente l'ensemble dominant S_{dom}. Une séquence s est dite compatible avec la séquence maître-pyramide $\sigma_\Delta(V)$ si l'ordre des travaux dans s ne contredit pas les ordres possibles définis par $\sigma_\Delta(V)$. Ceci sera noté dans la suite par $s \Subset \sigma_\Delta(V)$.

Examinons à présent la complexité temporelle liée à la construction d'une séquence maître-pyramide. L'algorithme de détermination des sommets et des pyramides, nécessitant de comparer chaque paire de travaux, a une complexité en $O(n^2)$ (cf [Briand *et al.* 03b]). Une fois les sommets et les pyramides déterminés, la construction de la séquence maître-pyramide se ramène à un ensemble de tris (deux pour chaque pyramide) de complexité $O(n \log n)$. La complexité au pire est donc en $O(n^2)$. Pour construire une séquence maître-pyramide, nous proposons l'algorithme décrit sur la Figure 3.5. On suppose que les travaux de V ont été triés par ordre de date de disponibilité croissant et que l'ensemble Δ des travaux sommets est connu. Comme les travaux communs aux pyramides P_k et P_{k+1} peuvent être séquencés dans un ordre arbitraire (les permutations sont temporellement équivalentes), d'après la définition de la séquence maître-pyramide, seuls les travaux appartenant à P_i et pas à P_{i-1} figurent dans α_i. Enfin, $\overline{J}$ est l'ensemble des travaux qui sont déjà placés dans $\sigma_\Delta(V)$.

Proc CREATION-SIGMA-DELTA()
1. for each job $i \in V$
2. do $\sigma_\Delta(V) \leftarrow \sigma_\Delta(V) \cup i$
3. $\quad \overline{J} \leftarrow \overline{J} \cup i$
4. $\quad$ if $i \in \Delta$ then
5. $\quad\quad$ for each job $j \in \overline{J}$ such that $d_j \geq d_i$
6. $\quad\quad$ do $\sigma_\Delta(V) \leftarrow \sigma_\Delta(V) \cup j$ (in increasing order of due dates)

FinProc

FIG. 3.5: Algorithme de construction de $\sigma_\Delta(V)$

Considérons à nouveau l'exemple de 6 travaux donné sur la Figure 3.3. Selon l'algorithme de construction décrit sur la Figure 3.5, la séquence maître-pyramide associée au problème est :

$$\sigma_\Delta(V) = (4, 5, 3, \mathbf{1}, 3, 4, 5, 6, \mathbf{2}, 5, 6)$$

On peut vérifier que toutes les séquences possibles de 6 travaux compatibles avec la séquence maître-pyramide $\sigma_\Delta(V)$, correspondent aux 24 séquences dominantes décrites sur la Figure 3.4 et déduites de l'application du Théorème des pyramides.

3.3.2 Relation entre ordre totaux et la séquence maître-pyramide

Dans les sections précédentes, nous avons d'une part rappelé que pour un problème V de n travaux défini par un ordre total entre les r_i et d_i des jobs $i \in V$, le Théorème des pyramides définit une condition analytique de dominance permettant de caractériser un ensemble de séquences dominantes S_{dom}. D'autre part, en introduisant la notion de séquence maître-pyramide, nous avons montré que toutes les séquences dominantes caractérisées par le Théorème des pyramides, i.e. S_{dom}, sont compatibles avec la séquence maître-pyramide $\sigma_\Delta(V)$. On peut ainsi dire que pour tout problème V de n travaux est associée une séquence maître-pyramide, elle même définissant l'ensemble de séquences dominantes.

Dans cette section, nous montrons, étant donné une séquence maître-pyramide et son ensemble correspondant S_{dom} de séquences dominantes, que plusieurs ordres totaux peuvent être définis pour la même séquence maître-pyramide, et donc, le même ensemble S_{dom} de séquences dominantes. Autrement dit, des problèmes possédant des structures d'intervalles distinctes peuvent avoir un même ensemble de séquences dominantes.

3.3.2.1 Description et notation

Proposition 3.1. *Si deux travaux i et j sont tel que i précède forcément j dans la séquence maître-pyramide i.e. appartiennent à deux pyramides consécutives différentes, alors l'ordre total entre les dates de début et de fin des deux travaux est soit, $r_i < d_i \leq r_j < d_j$ ou bien, $r_i < r_j < d_i < d_j$.*

Démonstration. Évidente du fait que si $i \prec j$ alors nécessairement $r_i < d_j$ tandis que l'ordre entre r_j et d_i peut être quelconque avec $r_i < d_i$ et $r_j < d_j$. La cas où $r_i \leq r_j < d_i \leq d_j$, n'est envisageable que si l'ensemble des travaux de la séquence maître sont constitués uniquement de sommets. □

Nous introduisons à présent la notion *d'ensemble d'ordres totaux* que nous notons E_{OT}, modélisé par un graphe particulier à partir duquel il est possible de dériver tous les ordres totaux relatifs à une séquence maître-pyramide donnée. E_{OT} est un graphe constitué de nœuds et d'arcs. Chaque nœud correspond à une date de début ou une date de fin d'un travail donné, et un arc relie deux nœuds n_i et n_j si $n_i \leq n_j$. La Figure 3.6 schématise le graphe E_{OT} relatif à deux travaux i et j d'une séquence maître pyramide $\sigma_\Delta(V) = (i, j)$ i et j étant deux sommets, respectant la Proposition 3.1. Le graphe E_{OT} représente l'ordre partiel de $\sigma_\Delta(V)$.

Tout ordre total est une extension linéaire de l'ordre partiel définit par le graphe E_{OT}. Si on considère l'exemple de la Figure 3.6, les ordres totaux $r_i < r_j < d_i < d_j$ et $r_i < d_i \leq r_j < d_j$ y sont déduits, et ont la même séquence maître pyramide $\sigma_\Delta(V)$.

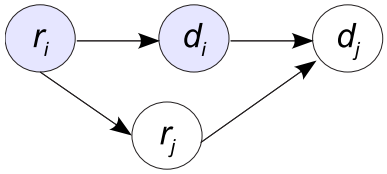

FIG. 3.6: Graphe E_{OT} relatif à deux jobs $i \prec j$

Proposition 3.2. *Étant donnée une séquence maître pyramide à structure pyramidale unique (une seule pyramide) alors l'ordre total lui correspondant est unique.*

Démonstration. Évidente. □

Lemme 3.1. *Étant donnée une séquence maître pyramide à plusieurs structures pyramidales alors il existe plusieurs ordres totaux (problèmes) pour la même séquence maître pyramide.*

Démonstration. Évidente d'après la Proposition **??**. □

3.3.2.2 Exemple illustratif

Considérons à nouveau l'exemple de la Section 3.2.3 et sa séquence maître-pyramide correspondante $\sigma_\Delta(V) = (4, 5, 3, \mathbf{1}, 3, 4, 5, 6, \mathbf{2}, 5, 6)$. Le graphe E_{OT} correspondant est représenté sur la Figure 3.7 suivante :

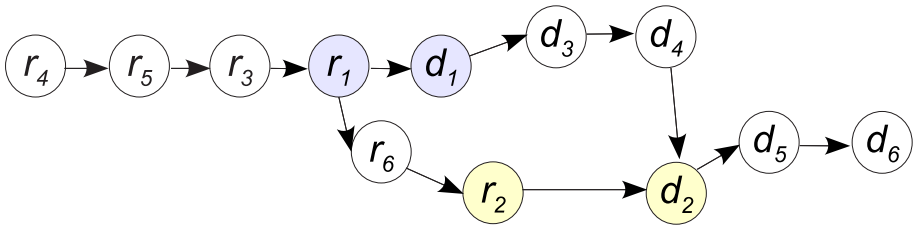

FIG. 3.7: E_{OT} correspondant à $\sigma_\Delta(V)$ de l'exemple de la Section 3.2.3

Étant donnée la séquence maître-pyramide $\sigma_\Delta(V) = (4, 5, 3, \mathbf{1}, 3, 4, 5, 6, \mathbf{2}, 5, 6)$, on peut remarquer que les relations de précédence sont strictes entre les couples de travaux (1,6), (3,6), (4,6), (1,2), (3,2) et (4,2). La Proposition 3.1 leur est ainsi appliquée ceci d'une part. D'autre part, sachant la structure pyramidale est à deux sommets, $P\{1\}$ et $P\{2\}$, avec les travaux 3 et 4 appartenant uniquement à la première pyramide, le travail 6 appartenant uniquement à la seconde pyramide, alors que le travail 5 appartient aux deux pyramides, les sous-ordres totaux suivants sont alors imposés : $r_4 < r_5 < r_3 < r_1 < d_1 < d_3 < d_4 < d_5 < d_6$ et $r_4 < r_5 < r_3 < r_1 < r_6 < r_2 < d_2 < d_5 < d_6$. Nous déduisons ainsi le graphe E_{OT} donné sur la Figure 3.7 décrivant l'ordre partiel des travaux de l'exemple. Si on se focalise sur le travail 6, on remarque que le nœud r_6 peut être placé soit avant d_1, entre d_1 et d_3, ou

entre d_3 et d_4, ou bien après d_4. Sa position est aussi conditionnée par celle du nœud r_2 qui comme le nœud r_6, peut être placé soit avant d_1, entre d_1 et d_3, ou entre d_3 et d_4 ou bien après d_4. Ainsi, il est possible de déduire tous les ordres totaux, en énumérant les ordres topologiques compatibles avec l'ordre partiel donné par E_{OT}. On peut alors compter un ensemble de 10 ordres totaux avec des inégalité strictes, i.e. 10 problèmes différents (cf. Figure 3.8), définissant le même ensemble de séquences dominantes que celui donné sur la Figure 3.4.

$r_4 < r_5 < r_3 < r_1 < r_6 < r_2 < d_1 < d_3 < d_4 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < r_6 < d_1 < r_2 < d_3 < d_4 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < r_6 < d_1 < d_3 < r_2 < d_4 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < r_6 < d_1 < d_3 < d_4 < r_2 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < d_1 < r_6 < r_2 < d_3 < d_4 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < d_1 < r_6 < d_3 < r_2 < d_4 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < d_1 < r_6 < d_3 < d_4 < r_2 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < d_1 < d_3 < r_6 < r_2 < d_4 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < d_1 < d_3 < r_6 < d_4 < r_2 < d_2 < d_5 < d_6$
$r_4 < r_5 < r_3 < r_1 < d_1 < d_3 < d_4 < r_6 < r_2 < d_2 < d_5 < d_6$

FIG. 3.8: Ensemble des ordre totaux correspondant aux extensions linéaires de l'ordre partiel donné sur la figure 3.7

Remarquons que l'ordre total donné en souligné dans la Figure 3.8, est l'ordre total correspondant à l'exemple de la Figure 3.3 à partir duquel nous avons déduit tous les ordres totaux ayant la même séquence maître pyramide $\sigma_\Delta(V) = (4, 5, 3, \mathbf{1}, 3, 4, 5, 6, \mathbf{2}, 5, 6)$.

Nous avons dans cette section, sachant que le Théorème des pyramides caractérise pour tout problème défini par une structure d'intervalles donnée (i.e. un ordre total donné) un ensemble de séquences dominantes, montré qu'il peut exister plusieurs problèmes (ordres totaux) dont l'ensemble de séquences dominantes est le même.

3.4 Extensions du Théorème de dominance

Dans la Section 3.3, la notion de séquence maître-pyramide a été introduite. Elle permet d'extraire l'ensemble de séquences dominantes S_{dom} que le Théorème des pyramides caractérise. De plus, nous avons montré comment caractériser tous les problèmes (ordres totaux) pour lesquels la séquence maître-pyramide est la même. Dans cette section, nous nous proposons d'introduire une nouvelle condition de dominance permettant de restreindre davantage l'ensemble dominant S_{dom} défini par la séquence maître-pyramide en distinguant les cas de sommets avec ou sans bases communes.

Soit γ une sous-séquence de travaux. On note :

- $c(\gamma)$: le makespan de l'ordonnancement γ, i.e. la date de fin du dernier travail dans γ ;
- r_γ (resp. d_γ) : la date de début (resp. date de fin) de la séquence γ, avec :
$$r_\gamma = \min_{k\in\gamma} r_k \text{ et } d_\gamma = \max_{k\in\gamma} d_k$$

On donne la définition suivante :

Définition 3.5. *On dit que deux sommets t_i et t_j sont indépendants s'ils n'ont pas de base commune, i.e. si $k \in P_i \Rightarrow k \notin P_j$*

Rappelons également la condition nécessaire et suffisante d'admissibilité établie par Erschler et al. dans [Erschler *et al.* 80] par le théorème suivant :

Théorème 3.2. *Une séquence de travaux σ est admissible si et seulement si :*

$$r_i + p_i + \sum_{i\prec k\prec j} p_k + p_j \leq d_j \ ,\forall (i,j) \in \sigma \text{ tel que } i \prec j \tag{3.2}$$

3.4.1 Cas de sommets avec base commune

Proposition 3.3. *Étant donnés deux sommets t_1, t_2 avec une base commune i et une séquence maître-pyramide $\sigma_\Delta = (i, t_1, i, t_2, i)$, alors la séquence (t_1, i, t_2) est dominée par (t_1, t_2, i) si $r_{t_1} + p_{t_1} \geq r_{t_2}$ (1) est valide, ou bien par (i, t_1, t_2) si $d_{t_2} - p_{t_2} \leq d_{t_1}$ (2) est valide.*

Démonstration. Affirmer que la séquence (t_1, i, t_2) est admissible implique d'après la condition 3.2 du Théorème 3.2 que les conditions suivantes sont valides : $d_i - r_{t_1} \geq p_{t_1} + p_i$ (3) ; $d_{t_2} - r_{t_1} \geq p_{t_1} + p_i + p_{t_2}$(4) et que $d_{t_2} - r_i \geq p_i + p_{t_2}$ (5). L'étude de l'admissibilité de la séquence (t_1, t_2, i) consiste à vérifier la validité des inégalités suivantes : $d_{t_2} - r_{t_1} \geq p_{t_1} + p_{t_2}$, vraie d'après la condition (4) ; $d_i - r_{t_1} \geq p_{t_1} + p_{t_2} + p_i$, vraie d'après (4) et sachant que $d_i > d_{t_2}$; et $d_i - r_{t_2} \geq p_{t_2} + p_i$ aussi vraie d'après (1) et (4) qui permettent de déduire que $d_{t_2} - r_{t_2} \geq p_i + p_{t_2}$ avec $d_i > d_{t_2}$. De la même manière, montrer que la séquence (i, t_1, t_2) est admissible consiste à vérifier que : $d_{t_2} - r_{t_1} \geq p_{t_1} + p_{t_2}$ vraie d'après (4) ; que $d_{t_2} - r_i \geq p_i + p_{t_1} + p_{t_2}$, vraie aussi d'après (4) et sachant $r_i < r_{t_1}$; et que $d_{t_1} - r_i \geq p_i + p_{t_1}$ vraie également d'après les conditions (2) et (4) permettant de déduire que $d_{t_1} - r_{t_1} \geq p_i + p_{t_1}$ sachant $r_i < r_{t_1}$.

□

3.4.2 Cas de sommets sans bases communes

Considérons à présent le cas de sommets sans base commune. Avant d'énoncer une nouvelle condition de dominance donnée par la Proposition 3.5, et dans un souci de clarté,

examinons d'abord le cas simple d'un problème V de n travaux contenant uniquement deux sommets.

Proposition 3.4. *Si pour une séquence maître-pyramide $\sigma_\Delta(V) = (\alpha_i, t_i, \beta_i, \alpha_j, t_j, \beta_j)$ à deux sommets t_i et t_j, on a $r_{t_j} \leq d_{t_i}$ et s'il n'existe pas de bases communes aux deux sommets, alors la séquence $\sigma = (t_i, \beta_i, \alpha_j, t_j)$ compatible avec $\sigma_\Delta(V)$ est dominée par $(\alpha_i, t_i, \alpha_j, t_j)$ ou bien $(\alpha_i, t_i, t_j, \beta_j)$ ou bien $(t_i, \beta_i, t_j, \beta_j)$, toutes les trois compatibles avec $\sigma_\Delta(V)$.*

Démonstration. On veut montrer que si $\sigma = (t_i, \beta_i, \alpha_j, t_j)$ est admissible alors nécessairement, au moins une des trois séquences citées dans la Proposition 3.4 est aussi admissible. Par hypothèse, toute tâche appartient à une et une seule pyramide, i.e, elle ne peut être rangée qu'en deux positions possibles relativement à tous les sommets. Nous rappelons que si σ est admissible alors d'après le Théorème 3.2 la condition $c(t_i, \beta_i) \leq d_{\beta_i}$ est vraie. Compte tenu des ordres définis par les r_i et d_i des jobs de V, ainsi que la valeur de $c(t_i, \beta_i)$, il est alors possible de distinguer 3 situations possibles (voir Figure 3.9) :

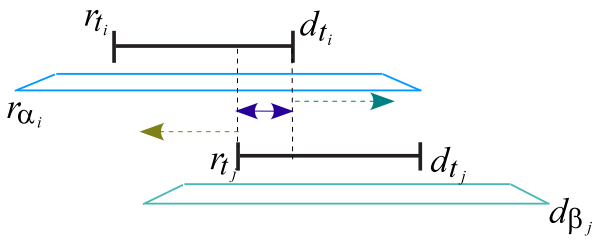

FIG. 3.9: Overlaps (t_i, t_j)

1- $c(t_i, \beta_i) \leq r_{t_j}$: Par hypothèse, nous savons que $r_{t_j} \leq d_k \ \forall k \in \beta_i$. On peut alors supposer que tous les travaux de (t_i, β_i) ont une date d'échéance égale à r_{t_j}. Ainsi, d'après le Théorème des pyramides les travaux de (t_i, β_i) peuvent être rangés par ordre croissant de leurs dates de début. Il résulte dans ce cas que (α_i, t_i) domine (t_i, β_i), et par conséquent, $c(t_i, \beta_i) \leq r_{t_j} < c(\alpha_i, t_i)$. On conclut donc que $(t_i, \beta_i, \alpha_j, t_j)$ est dominée par $(\alpha_i, t_i, \alpha_j, t_j)$.

2- $c(t_i, \beta_i) > d_{t_i}$: Par hypothèse, nous savons que $d_{t_i} > r_k \ \forall k \in \alpha_i$. Il est alors possible de supposer que tous les travaux de (α_j, t_j) ont une date de disponibilité date égale à d_{t_i}. Ainsi, d'après le Théorème des pyramides, les travaux de (α_j, t_j) peuvent être rangés par ordre croissant de leur dates de fin, d'où (α_j, t_j) dominée par (t_j, β_j). D'où $(t_i, \beta_i, \alpha_j, t_j)$ dominée par $(t_i, \beta_i, t_j, \beta_j)$.

3- $r_{t_j} \leq c(t_i, \beta_i) \leq d_{t_i}$: D'une part et comme au premier point, sachant $c(t_i, \beta_i) \leq d_{t_i}$ on peut supposer que tous les travaux dans (t_i, β_i) ont une date d'échéance égale à d_{t_i} et qu'elles peuvent ainsi être rangées dans l'ordre croissant de leur date de

début. D'autre part, puisque $c(t_i, \beta_i) \geq r_{t_j}$, on peut supposer comme au second point, que tous les travaux dans α_j ont une date de disponibilité égale à r_{t_j} et qu'elles peuvent ainsi être rangées dans l'ordre croissant de leur date de fin. Il en découle que $(t_i, \beta_i, \alpha_j, t_j)$ est dominée par $(\alpha_i, t_i, \alpha_j, t_j)$ ainsi que par $(t_i, \beta_i, t_j, \beta_j)$, toutes les deux dominées par $(\alpha_i, t_i, t_j, \beta_j)$.

□

Signalons qu'à travers la Proposition 3.4, nous avons voulu montrer que, toute séquence qui placerait des travaux appartenant à deux pyramides différentes entre leurs sommets est dominée par une autre séquence qui placerait au plus des travaux appartenant uniquement à une et une seule pyramide entre les deux sommets.

Proposition 3.5. *Étant donnés un ensemble V de n travaux, $\{t_1..t_m\}$ sommets et une séquence maître-pyramide $\sigma_\Delta(V)$. Si La condition $r_{t_m} < d_{t_1}$ (Les intervalles des sommets se chevauchent) est vérifié et s'il n'existe pas de bases communes pour tout couple de sommets de V alors, toute séquence dominante $\sigma \Subset \sigma_\Delta(V)$ qui placerait des travaux appartenant à deux pyramides différentes entre leurs sommets, est dominée par une autre séquence $\sigma' \Subset \sigma_\Delta(V)$ tel que les travaux appartenant à seulement une et une seule pyramide sont placés entre les deux sommets.*

Démonstration. Considérons par $\sigma = (\alpha'_1, t_1, ..., t_k, \beta'_k, \alpha'_{k+1}, t_{k+1}, ...\beta'_m)$ une séquence dominante extraite de $\sigma_\Delta(V)$ tels que $\{\beta'_k\} \neq \emptyset$ et $\{\alpha'_{k+1}\} \neq \emptyset$. Sachant qu'entre les sommets t_k et t_{k+1} des travaux appartenant à deux pyramides différentes y sont rangés, nous focalisons alors notre intérêt sur la sous-séquence $(\alpha'_1, t_1, ..., t_k, \beta'_k)$ et à la valeur de sa date de fin d'exécution relativement à d_{t_k}(cf. Figure 3.10). On distingue alors deux cas :

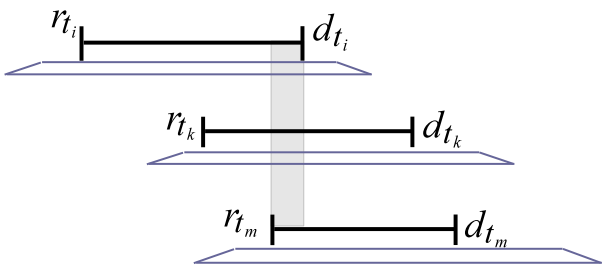

FIG. 3.10: Structure pyramidale de m sommets sans bases communes et $r_{t_m} < d_{t_1}$

1- $c(\alpha'_1, t_1, ..., t_k, \beta'_k) \leq d_{t_k}$) : Dans ce cas, sachant $d_j < d_{t_k}$ $\forall j \in \beta'_k$, on peut alors supposer que tous les travaux dans β'_k ont une date d'échéance égale à d_{t_k}. D'après le Théorème des pyramides, les jobs dans β'_k peuvent être rangés avant le sommet t_k dans l'ordre croissant de leur date de disponibilité. D'où, la séquence $\sigma = (\alpha'_1, t_1...t_k, \beta'_k, \alpha'_{k+1}, t_{k+1}...\beta'_m)$ dominée par la séquence $\sigma' = (\alpha'_1, t_1...\alpha_k, t_k, \alpha'_{k+1}, t_{k+1}...\beta'_m)$

avec $\sigma' \Subset \sigma_\Delta(V)$;

2- $c(\alpha'_1, t_1, ..., t_k, \beta'_k) > d_{t_k})$: Par hypothèse nous avons $r_i < d_{t_1}\ \forall i \in V$ et $d_1 < d_{t_k}$, i.e. $r_i < d_{t_k}\ \forall i \in V$. Nous pouvons alors supposer que le reste des travaux dans σ, i.e. $(\alpha'_{k+1}, t_{k+1}, ...\beta'_m)$, ont une date de disponibilité égale à d_{t_k}. D'après le Théorème des pyramides, les jobs dans $(\alpha'_{k+1}, t_{k+1}, ...\beta'_m)$ peuvent être rangés dans l'ordre croissant de leur date d'échéance. D'où, la séquence $\sigma = (\alpha'_1, t_1, ..., t_k, \beta'_k, \alpha'_{k+1}, t_{k+1}...\beta'_m)$ dominée par $\sigma' = (\alpha'_1, t_1...t_k, \beta'_k, t_{k+1}, \beta'_k...\beta'_m)$ avec $\sigma' \Subset \sigma_\Delta(V)$.

□

3.5 Nouvelles conditions numériques de dominance

Le Théorème 3.2 donné en Section 3.4 énonce, étant donné une séquence σ de travaux, la condition nécessaire et suffisante d'admissibilité. Dans cette partie, en se basant sur l'ordre partiel défini par le Théorème des pyramides, nous cherchons à cerner les conditions qui permettent de définir, si elle existe, une séquence admissible parmi l'ensemble des séquences dominantes. En distinguant les cas mono-pyramidal et multi-pyramidal, des conditions numériques de dominance et des théorèmes sont énoncés et démontrés.

3.5.1 Cas mono-pyramidal

Nous considérons dans cette section que la structure d'intervalles est constituée d'une seule pyramide P associée au seul sommet t. La séquence maître-pyramide associée à la structure pyramidale P est noté $\sigma_\Delta(V) = \alpha \prec t \prec \beta$. Étant donnée une séquence σ de n travaux de la forme $\alpha' \prec t \prec \beta'$ avec $\sigma \Subset \sigma_\Delta(V)$), on note :

$$R = \max_{i \in \alpha'} (r_t, r_i + p_i + \sum_{k \in \alpha', i \prec k} p_k); \tag{3.3}$$

$$D = \min_{j \in \beta'} (d_t, d_j - p_j - \sum_{k \in \beta', k \prec j} p_k). \tag{3.4}$$

L'équation (3.3) définit la date de début au plus tôt du sommet t dans la séquence σ tout en assurant que les travaux dans α' commencent leur exécution après leur date de disponibilité. L'équation (3.4) définit quant à elle, la date de fin au plus tard du sommet t dans la séquence σ, tel que tous les travaux dans β' sont contraints à finir à l'heure.

Théorème 3.3. *Soit P une pyramide de sommet t. Toute séquence σ de la forme $\alpha' \prec t \prec \beta'$ compatible avec $\sigma_\Delta(V)$ est admissible si et seulement si $D - R \geq p_t$.*

Démonstration. Montrons d'abord que si $D - R \geq p_t$ alors $\alpha' \prec t \prec \beta'$ est admissible. Nous savons d'après la définition de D et R que les valeurs de ces deux variables définissent

l'intervalle de réalisation du sommet t sachant que tous les travaux dans α' commencent leur réalisation après leurs dates de disponibilité et que tous les travaux dans β' finissent leur exécution avant leurs dates d'échéance. Il est évident que si $D - R \geq p_t$ alors t est exécuté à l'heure ainsi que tous les travaux dans β' puisque $\forall i \in \alpha'$ on a $d_i > d_t$. Nous allons maintenant montrer que si $\alpha' \prec t \prec \beta'$ est admissible, alors $D - R \geq p_t$. D'après la condition d'admissibilité donnée par le Théorème 3.2, on peut facilement déduire que si la condition est vraie pour tout $i \in \sigma$ et $j = t$ alors la condition est aussi vraie pour tout couple (i, j) dans α'.

Sachant que le sommet t n'appartient ni à α' ni à β', la condition 3.2 peut alors s'écrire :

$$\begin{cases} d_t - r_i - p_i - \sum_{k \succ i, k \in \alpha'} p_k - p_t \geq 0 & \forall i \in \alpha' \\ d_j - r_t - p_t - \sum_{k \prec j, k \in \beta'} p_k - p_j \geq 0 & \forall j \in \beta' \\ d_j - r_i - p_i - \sum_{k \succ i, k \in \alpha'} p_k - p_t - \sum_{k \prec j k \in \beta'} p_k - p_j \geq 0 & \forall i \in \alpha', \forall j \in \beta' \end{cases}$$

Et peut aussi s'écrire :

$$\begin{cases} d_t - \max_{i \in \alpha'}(r_i + p_i + \sum_{k \succ i, k \in \alpha'} p_k) \geq p_t \\ \min_{j \in \beta'}(d_j - p_j - \sum_{k \prec j, k \in \beta'} p_k) - r_t \geq p_t \\ \min_{j \in \beta'}(d_j - p_j - \sum_{k \prec j, k \in \beta'} p_k) - \max_{i \in \alpha'}(r_i + p_i + \sum_{k \succ i, k \in \alpha'} p_k) \geq p_t \end{cases}$$

Par définition, on a $d_t - r_t \geq p_t$. Le système d'inéquations ci-dessus peut alors s'écrire :

$\min_{j \in \beta'}(d_t, d_j - p_j - \sum_{k \in \beta', k \prec j} p_k) - \max_{i \in \alpha'}(r_t, r_i + p_i + \sum_{k \in \alpha', i \prec k} p_k) \geq p_t$;
i.e. $D - R \geq p_t$. □

Le Théorème 3.3 permet une comparaison numérique des séquences dominantes entre elles. De ce résultat nous déduisons une condition numérique de dominance.

Corollaire 3.1. *Soit P une pyramide de sommet t et soient σ_1 et σ_2 deux séquences dominantes respectivement de la forme $\alpha'_1 \prec t \prec \beta'_1$ et $\alpha'_2 \prec t \prec \beta'_2$. Si $D^1 - R^1 \geq D^2 - R^2$, alors σ_1 domine σ_2 vis-à-vis de l'admissibilité.*

Démonstration. Par définition, dire que σ_1 domine σ_2 nécessite de montrer que, si σ_2 est admissible, alors σ_1 est aussi admissible. Nous savons d'après le Théorème 3.3 que σ_2 est admissible si et seulement si $D^2 - R^2 \geq p_t$. Par conséquent, sous l'hypothèse que $D^1 - R^1 \geq D^2 - R^2$, on déduit que $D^1 - R^1 \geq p_t$, et ainsi, σ_1 est nécessairement admissible. □

Corollaire 3.2. *Soit P une pyramide de sommet t et une séquence maître-pyramide $\sigma_\Delta(V) = \alpha \prec t \prec \beta$. Soit S_{dom} l'ensemble des séquences dominantes σ_i compatibles avec $\sigma_\Delta(V)$. Si $\max_{\sigma_i \in S_{dom}}(D^{\sigma_i} - R^{\sigma_i}) \geq p_t$ alors, il existe au moins une séquence admissible au problème P.*

Démonstration. Évidente. □

3.5.2 Cas multi-pyramidal

Dans cette section, nous nous intéressons au problème V de n travaux défini par une structure pyramidale à m sommets et par une séquence maître-pyramide $\sigma_\Delta(V) = \alpha_1 \prec t_1 \prec \beta_1 \cdots \prec \alpha_m \prec t_m \prec \beta_m$ et où t_k représente le sommet de la pyramide $P_k \forall k = \{1 \dots m\}$. Toute séquence dominante de n travaux $\sigma_i \Subset \sigma_\Delta(V)$ est de la forme $\sigma = \alpha'_1 \prec t_1 \prec \beta'_1 \cdots \prec \alpha'_m \prec t_m \prec \beta'_m$, avec $\alpha'_i \in \alpha_i$, $\beta'_i \in \beta_i$, $\alpha'_i \cap \beta'_i = \oslash$ $\forall i \in \{1, ..., m\}$ et avec $\cup_{i=1\dots m}\{\alpha'_i, t_i, \beta'_i\} = V$. Comme dans le cas mono-pyramidal, nous cherchons ici à établir les conditions qui permettent de définir, si elle existe, une séquence admissible.

Étant donné une séquence $\alpha'_k \prec t_k \prec \beta'_k$, on note :

$$\begin{cases} eft_k = R_k + p_{t_k} + \sum_{j \in \beta'_k} p_j \\ \\ lst_k = D_k - p_{t_k} - \sum_{j \in \alpha'_k} p_j \end{cases} \tag{3.5}$$

où eft_k (resp. lst_k) désigne la date de fin au plus tôt (resp. la date de début au plus tard) de la séquence $\alpha'_k \prec t_k \prec \beta'_k$, R_k la date de début au plus tôt de t_k et D_k la date de fin au plus tard de t_k.

Étant données les sous-séquences $\alpha'_i \prec t_i \prec \beta'_i$, $i \in \{1, ..., m\}$, la date de début (resp. date de fin) de tout travail j d'une sous-séquence $\alpha'_k \prec t_k \prec \beta'_k$ est contrainte non seulement par sa date de disponibilité r_j (resp. sa due date d_j), mais aussi par la date de fin de la séquence $\alpha'_{k-1} \prec t_{k-1} \prec \beta'_{k-1}$ (resp. date de début de la séquence $\alpha'_{k+1} \prec t_{k+1} \prec \beta'_{k+1}$), voir Figure 3.11. On a donc $\forall j \in V$:

$$\begin{cases} r_j = \max(r_j, eft_{k-1}) \\ \\ d_j = \min(d_j, lst_{k+1}) \end{cases} \tag{3.6}$$

Dans le cas multi-pyramidal, considérant une sous-séquence $\alpha'_k \prec t_k \prec \beta'_k$ de σ, on note R_k (resp. D_k), dont l'expression est donnée par l'équation 3.3 (resp. 3.4), la date de début au plus tôt(resp. de fin au plus tard) du sommet t_k tout en considérant que les travaux dans α'_k (resp. β'_k) commencent (resp. finissent) leur exécution après leur date de disponibilité (date échue) : expressions données par le système d'équation 3.6.

Théorème 3.4. *Soit $\sigma_\Delta(V)$ une séquence maître-pyramide de n travaux et m sommets. Toute séquence σ de la forme $\alpha'_1 \prec t_1 \prec \beta'_1 \dots \alpha'_m \prec t_m \prec \beta'_m$ compatible avec $\sigma_\Delta(V)$ est admissible si et seulement si $\min_{k=1\dots m}(D_k - R_k - p_{t_k}) \geq 0$.*

Démonstration. Montrons d'abords que si $D_k - R_k - p_{t_k} \geq 0$ $\forall k = \{1...m\}$ alors σ est admissible. D'après les définitions de R_k et D_k données par les formules 3.3, 3.4 et 4.6, nous savons que les sous-séquences $\alpha'_k \prec t_k \prec \beta'_k$ sont construites de manière à ne pas se

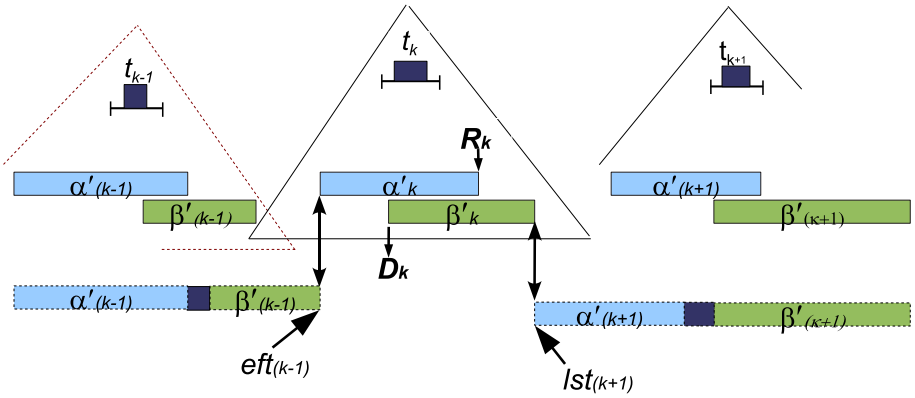

FIG. 3.11: Inter-dépendance des sous-séquences d'une structure multi-pyramidale

chevaucher, voir Figure 3.11. De plus, d'après le Théorème 3.3, on déduit que $\alpha'_k \prec t_k \prec \beta'_k$ est admissible $\forall i = \{1...m\}$. Il découle que la séquence σ est admissible. Supposons maintenant que σ est admissible, ce qui implique que toute sous-séquence $\alpha'_k \prec t_k \prec \beta'_k$ de σ est admissible. D'après le Théorème 3.3, on déduit que $(D_k - R_k - p_{t_k}) \geq 0$ et ce $\forall k = \{1...m\}$, ce qui revient à dire que $\min_{k=1...m}(D_k - R_k - p_{t_k}) \geq 0$. □

Corollaire 3.3. *Soit $\sigma_\Delta(V)$ une séquence maître-pyramide de n travaux et m sommets. Considérons σ_1 et σ_2 deux séquences dominantes compatibles avec $\sigma_\Delta(V)$. Si $\min_{k=1...m}(D_k^1 - R_k^1) \geq \min_{k=1...m}(D_k^2 - R_k^2)$, alors σ_1 domine σ_2 vis-à-vis de l'admissibilité.*

Démonstration. Par définition, dire que σ_1 domine σ_2 nécessite de montrer que, si σ_2 est admissible alors σ_1 est aussi admissible. Nous savons d'après le Théorème 3.4 que σ_2 est admissible si et seulement si $\min_{k=1..m}(D_k^2 - R_k^2 - p_{t_k}) \geq 0$. Par conséquent, sous l'hypothèse que $\min_{k=1..m}(D_k^1 - R_k^1) \geq \min_{k=1..m}(D_k^2 - R_k^2)$, il en découle que $\min_{k=1..m}(D_k^1 - R_k^1 - p_{t_k}) \geq 0$, et ainsi σ_1 est nécessairement admissible. □

Corollaire 3.4. *Soient $\sigma_\Delta(V)$ une séquence maître-pyramide de n travaux et m sommets, et S_{dom} l'ensemble de séquences σ_i compatibles avec $\sigma_\Delta(V)$. Si $\max_{\sigma_i \in S_{dom}}(\min_{k=1...m}(D_k - R_k - p_{t_k})) \geq 0$, alors il existe au moins une séquence admissible au problème V de n travaux.*

Démonstration. Évidente. □

Dans le cadre de l'étude d'admissibilité, nous avons apporté des extensions à la condition analytique de dominance, et avons énoncé des conditions numériques de dominance. Ces résultats seront utilisés dans le Chapitre 4.

3.6 Condition de dominance pour la minimisation de ΣU_i

3.6.1 Introduction

Dans cette partie, nous nous intéressons à généraliser la condition de dominance d'Erschler et al. au problème de minimisation du nombre de travaux en retard, noté $1|r_i|\sum U_i$, où $U_i = 1$ si le travail i est en retard (i.e. $U_i \leftarrow s_i + p_i > d_i$). On rappelle que ce problème est NP-difficile [Lenstra *et al.* 77]. On trouve dans la littérature plusieurs méthodes de résolution exactes basées sur une méthode de séparation et évaluation. Les derniers travaux de recherche traitant de ce problème sont ceux décrits dans [Baptiste *et al.* 03] et [Dauzère-Pérès & Sevaux 04]. Les méthodes proposées sont efficaces et peuvent être appliquées à des problèmes comptant plus d'une centaine de travaux, des problèmes de telle taille étant abordés et résolus de façon exacte pour la première fois durant cette décennie. Durant la préparation de ce travail, M'Hallah et Bulfin publièrent également les résultats de leurs travaux [M'Hallah & Bulfin 07]. Leur méthode très efficace basée sur un Branch and Bound a permis de trouver les solutions optimales pour toutes les instances difficiles n'ayant pas pu être résolues par les approches antérieures.

Toutefois, certains cas particuliers du problème $1|r_i|\sum U_i$ sont faciles. Dans le cas particulier où les dates de disponibilité des travaux sont identiques ($r_i = r$ ou $d_i = d$), le problème noté $1||\sum U_i$ est résolu en $O(n \log n)$ par l'algorithme de Moore [Moore 68]. Pour le même problème, avec durées opératoires unitaires, C.L. Monma [Monma 82] a proposé un algorithme en $O(n)$. Dans le cas où les release et due dates des travaux sont similairement ordonnés (i.e. $r_i < r_j \Rightarrow d_i \leq d_j$), Kise et al. dans [Kise *et al.* 78] ont proposé un algorithme en $O(n^2)$ pour résoudre le problème. Sous les mêmes conditions E.L. Lawler [Lawler 82] a décrit un peu plus tard un algorithme plus efficace en $O(n \log n)$. E.L. Lawler dans [Lawler 90] a également proposé un algorithme en $O(n^5)$ pour le cas préemptif $1|r_i, pmtn|\sum U_i$, complexité ramenée en 1999 à $O(n^4)$ par P. Baptise [Baptiste 99a]. Lorsque les durées opératoires sont identiques, Carlier a démontré dans [Carlier 81] que le problème $1|p_i = p, r_i|\sum U_i$ peut être résolu en $O(n^3 \log n)$. Pour le problème général où des poids sont associés aux travaux, $1|p_i = p, r_i|\sum w_i U_i$, Baptiste [Baptiste 99b] a proposé un algorithme en $O(n^7)$. Récemment, Chrobak et al. ont montré dans [Chrobak *et al.* 06] que la méthode de carlier [Carlier 81] n'est pas optimale, les auteurs ont apporté des modifications à la méthode de baptiste [Baptiste 99b] et ont proposé un algorithme de résolution en $O(n^5)$ pour le problème $1|p_i = p, r_i|\sum U_i$ sans poids sur les travaux.

3.6.2 Théorème des pyramides et minimisation de ΣU_i

Trouver une solution optimale au problème $\sum U_i$ consiste à déterminer une séquence admissible pour une sélection de travaux $E^* \subseteq V$ la plus grande possible. Les travaux de E^* sont donc à l'heure alors que les autres travaux sont tous en retard. Les travaux en

retard peuvent être séquencés après ceux de E^*, dans n'importe quel ordre et à n'importe quelle date. Lors de la recherche d'une solution admissible à k travaux, les $n-k$ travaux en retard peuvent donc être omis. Étant dominant vis-à-vis de l'admissibilité, le Théorème des pyramides peut alors être appliqué pour caractériser un sous-ensemble de séquences dominantes vis-à-vis du critère autant de fois qu'il existe de sélections différentes de k travaux. Le corollaire suivant est alors démontré.

Corollaire 3.5. *L'union de toutes les séquences dominantes que le Théorème 3.2 caractérise pour toute sélection de travaux est dominante vis-à-vis du critère* $\sum U_i$.

Démonstration. la preuve est évidente du fait que l'union de toutes les séquences dominantes que le Théorème 3.2 caractérise pour toute sélection possible inclue nécessairement les séquences dominantes associées à l'ensemble E^*, et donc la solution optimale. □

On remarque que pour un problème à n travaux, il existe un nombre gigantesque de sélections différentes de travaux (*i.e.* $\sum_{k=1..n} A_n^k$). Fort heureusement, comme il sera démontré plus loin, il n'est pas nécessaire d'appliquer le Théorème des pyramides à chaque sélection pour déterminer l'ensemble dominant S_{dom} que le Corollaire 3.5 caractérise. En effet, la notion de séquence maître-pyramide permet de caractériser un ensemble de séquences dominantes pour un nombre très important de sélections (et pas seulement celles à n travaux). Avant de démontrer cette propriété, le concept de *graphe sommet-base* doit être préalablement introduit.

3.6.3 Notion de graphes d'intervalles sommet-base

3.6.3.1 Définitions

Un graphe d'intervalles sommet-base $G = (X, U)$ est un graphe sans circuit dans lequel à tout nœud est associé un intervalle et où un arc $u \in U$ relie le nœud i au nœud j $(i, j \in X)$ si et seulement si l'intervalle i est strictement inclus dans l'intervalle j et s'il n'existe pas d'intervalle $k \in X$ tel que i est strictement inclus dans k et k est strictement inclus dans j.
Le graphe d'intervalles sommet-base définit une hiérarchie entre les intervalles fondée sur la relation d'inclusion stricte. Comme il ne contient pas de circuit, le graphe sommet-base peut être structuré en niveaux. Les nœuds de niveau 0 correspondent aux intervalles n'incluant aucun autre intervalle (ce sont des sommets au sens de la Définition 3.1). Les nœuds de niveau le plus bas correspondent aux intervalles qui ne sont inclus dans aucun autre intervalle, ce sont les feuilles de l'arbre (ces nœuds sont appelés bases au sens de la Définition 3.2). On note $i \prec j$, deux nœuds i, j de même niveau et de même père tel que $r_i \leq r_j \wedge d_i < d_j$ et , j étant alors placé à droite de i dans le graphe.

Sur la base du graphe sommet-base, on donne aussi les définitions suivantes (voir Figure 3.12) :

- un nœud j est dit successeur propre de i (ou d'un ensemble de nœuds e_i) si j n'est pas une feuille et s'il ne possède que i (les nœuds de e_i) comme prédécesseur(s) ;
- si un nœud j possède plusieurs successeurs de même niveau, alors ces successeurs de j sont dits voisins ;
- si un nœud i possède l successeurs voisins $\{j_1...j_l\}$ avec $r_{j_1} < .... < r_{j_l}$ alors j_1 est dit premier successeur voisin de i et j_l dernier successeur voisin de i ;
- un nœud i est dit sommet potentiel s'il possède un degré sortant supérieur à un ;
- un ensemble de nœuds e_i est dit générateur si, chaque élément $k \in e_i$ est sommet potentiel, s'il existe un ensemble e'_i de nœuds tous successeurs voisins de $k \in e_i$, et s'il existe au moins un successeur propre non dernier successeur de k ;
- lorsque l'ensemble générateur n'est constitué que d'un seul nœud k, alors k est dit sommet générateur ;
- un sommet généré i est un sommet potentiel créé suite à l'élimination d'un sommet générateur k lui même prédécesseur de i, ainsi que tous les prédécesseurs de k, noté $Pred(k)$;
- un nœuds k est dit générateur sans base commune si $\forall(i,j)$ successeurs de k tel que $r_i < r_j \Rightarrow d_i < d_j$, il $\nexists k'$ successeur de k tel que $r'_k < r_i$ et $d'_k > d_j$.

On note $i \prec j$, deux nœuds i et j de même niveau tel que $r_i \leq r_j$ et $d_i \leq d_j$.

Proposition 3.6. *Si un nœud i_1 possède 2 successeurs voisins j_1 et j_2 avec $j_1 \prec j_2$, et s'il existe un nœud i_2 tel que $i_1 \prec i_2$ et j_1 successeur voisin de i_2, alors j_2 est forcement successeur voisin de i_2.*

Démonstration. Supposons que j_2 n'est pas successeur voisin de i_2, ceci implique que $d_{j_2} < d_{i_2}$. Cela est en contradiction avec l'hypothèse que $d_{j_1} > d_{i_2} \wedge d_{j_2} > d_{j_1}$. □

Proposition 3.7. *Soient t un sommet générateur, e_t l'ensemble de ses successeurs voisins et $k \in e_t$ premier successeur voisin de t. Si k est successeur voisin d'un autre sommet potentiel t' tel que $t \prec t'$ alors, l'ensemble des nœud $\{t, ..., t'\}$ prédécesseurs voisins de k constitue un ensemble de sommets générateurs.*

Démonstration. D'après la propriétéproposition 3.6, tout successeur voisin de t dans e_k hormis k est aussi prédécesseur voisin de t'. Comme t est premier prédécesseur de k alors par définition, l'ensemble des nœuds $\{t, ..., t'\}$ prédécesseurs voisins de k est un ensemble de sommets générateurs. □

Un graphe d'intervalles sommet-base peut être associé à un problème à une machine avec fenêtres d'exécutions. À titre illustratif, la Figure 3.13 décrit le graphe associé au problème de 6 travaux de la Figure 3.3. Les sommets sont les travaux 1 et 2. Il y a donc deux pyramides. Remarquons que le sous-graphe associé à la descendance d'un sommet correspond à une pyramide au sens de la Définition 3.3. La première (descendance de 1) est formée des travaux 3, 4 et 5 ; la seconde (descendance de 2) est formée des travaux 5 et 6. Aucun sommet de la Figure 3.13 n'est sommet potentiel.

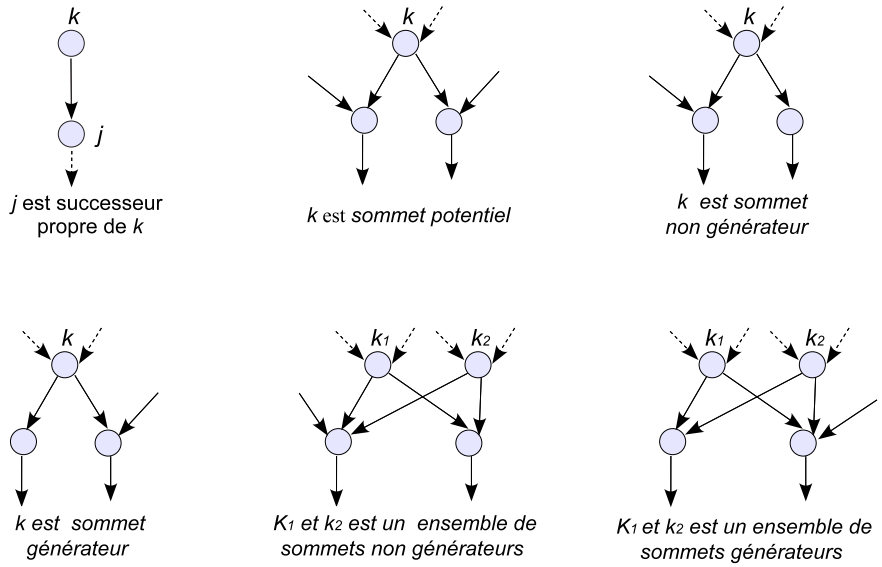

FIG. 3.12: Exemples illustrant les types de sommet(s)

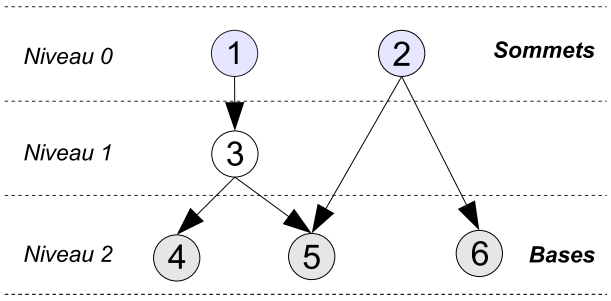

FIG. 3.13: Graphe d'intervalles sommet-base associé à l'exemple de la Figure 3.3

3.6.4 Étude de la dominance du théorème des pyramides

Nous avons, dans la section précédente, introduit la notion de sommet générateur. Cette notion sera utilisée, dans la suite, pour déterminer sous quelles conditions le Théorème des pyramide, dominant vis-à-vis de l'admissibilité, est dominant dans le cas de la minimisation du nombre de travaux en retard. Nous montrons, dans le cas où le Théorème des pyramide n'est pas dominant vis-à-vis du problème de la minimisation du nombre de travaux en retard, et à travers un exemple, qu'il est possible de définir toutes les séquences dominantes, pour toutes les sélections de travaux possibles, garantissant l'existence de la solution optimale.

Le théorème suivant montre que, sous certaines hypothèses, la condition de dominance formulée par le Théorème des pyramides, vraie vis-à-vis de la recherche de solution admissible, est également vraie vis-à-vis du problème de minimisation du nombre de travaux en

retard.

Théorème 3.5. *Étant donné un problème $1|r_i|\sum U_i$ et son graphe d'intervalles sommet-base associé G, si tout nœud de G est tel qu'il ne possède au plus qu'un seul successeur alors, la séquence maître pyramide $\sigma_\Delta(V) = (\alpha_0 \prec t_0 \prec \beta_0 ... \alpha_m \prec t_m \prec \beta_m)$, construite à partir des n travaux de V, caractérise l'ensemble S_{dom} des séquences dominantes vis-à-vis de la minimisation du nombre de travaux en retard.*

Démonstration. Le Théorème 3.5 stipule que la séquence maître-pyramide $\sigma_\Delta(V)$, caractérisant l'ensemble de toutes les séquences dominantes à n travaux, caractérise également les séquences dominantes de k travaux, $k = 1$ à $n-1$. En effet, sous l'hypothèse que chaque nœud de G ne possède au plus qu'un seul successeur, le graphe G est un ensemble de une ou plusieurs anti-arborescences. La séquence maître-pyramide $\sigma_\Delta(V) = (\alpha_0 \prec t_0 \prec \beta_0, ..., \alpha_m \prec t_m \prec \beta_m)$ définit alors un chemin dans cette anti-arborescence, comme illustré sur la Figure 3.14.a). La chaîne reliant un sommet t_i à sa base (unique) correspond à α_i lorsqu'elle est parcourue dans le sens base vers sommet, à β_i, dans le sens inverse. Si on considère une sélection de $k < n$ travaux, le nouveau graphe G' obtenu est également un ensemble de une ou plusieurs anti-arborescences. De plus, ainsi qu'illustré sur la Figure 3.14.b), la séquence maître-pyramide $\sigma_\Delta(V')$ obtenue pour cette sélection définit alors un chemin dans G' qui est forcément inclus dans le chemin précédent. Autrement dit, toutes les séquences à k travaux que $\sigma_\Delta(V')$ caractérise sont déjà caractérisées par $\sigma_\Delta(V)$.

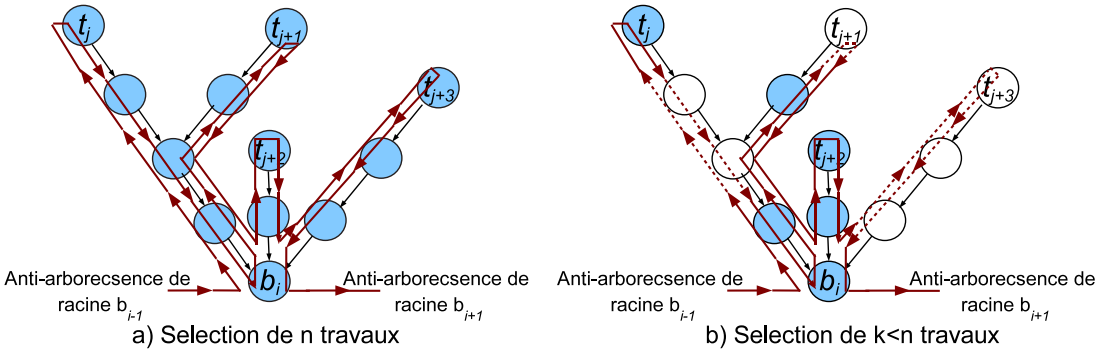

FIG. 3.14: Anti-arborescence et séquence maître-pyramide

□

Notons que le Théorème 3.5 considère le cas particulier où tous les intervalles sont inclus les uns dans les autres i.e., les pyramides sont parfaites (cf. Définition 4.2 donnée à la section 4.3.4.2). Tous les nœuds du graphe sommet-bases G associé au problème V ne sont donc pas des sommets potentiels, et sont forcement non générateurs. Dans ce qui suit, nous allons nous intéresser au cas général où des sommets peuvent être des sommets potentiels. Des propositions sont énoncées, avec comme objectif de montrer, étant donné

un ensemble V de travaux et une sélection V' de V, sous quelles conditions la séquence maître $\sigma_\Delta(V')$ est compatible avec $\sigma_\Delta(V)$ et ce, selon la nature des travaux $U = V \setminus V'$ éliminés de V.

Proposition 3.8. *Étant donnés un ensemble V de n travaux et son graphe intervalle sommet-base associé G, si un sous-ensemble U de V est tel que tout élément de U est non sommet de V au sens de la Définition 3.1 alors, $\sigma_\Delta(V \setminus U) \Subset \sigma_\Delta(V)$.*

Démonstration. Évidente d'après la définition de la séquence maître-pyramide basée sur les position fixes des sommets. En effet, l'élimination d'un nœud i non sommet de V revient à l'éliminer respectivement de toutes les sous-séquences α_k et β_k de $\sigma_\Delta(V)$ tel que $u(i) \leq k \leq v(i)$. D'où $\sigma_\Delta(V \setminus U) \Subset \sigma_\Delta(V)$. □

Proposition 3.9. *Étant donnés un ensemble V de n travaux et son graphe d'intervalles sommet-base associé G, si un nœud t_k de G est un sommet potentiel non générateur alors, $\sigma_\Delta(V \setminus \{t_k\}) \Subset \sigma_\Delta(V)$.*

Démonstration. La séquence maître-pyramide associée à l'ensemble V de n travaux et m sommets s'écrit $\sigma_\Delta(V) = (\alpha_0 \prec t_0 \prec ...t_{k-1} \prec \beta_{k-1} \prec \alpha_k \prec t_k \prec \beta_k \prec \alpha_{k+1} \prec t_{k+1}... \prec \beta_m)$ alors que la séquence maître-pyramide de $V' = V \setminus \{t_k\}$ est de la forme $\sigma_\Delta(V') = (\alpha_0 \prec t_0 \prec ... \prec t_{k-1} \prec \beta'_{k-1} \prec \alpha'_{k+1} \prec t_{k+1} \prec ... \prec \beta_m)$ étant donné que t_k n'est pas générateur (aucun sommet n'est généré suite à son élimination). L'étude de la compatibilité de $\sigma_\Delta(V')$ avec $\sigma_\Delta(V)$ revient alors à comparer la séquence $\beta'_{k-1} \prec \alpha'_{k+1}$ de V' avec $\beta_{k-1} \prec \alpha_k \prec t_k \prec \beta_k \prec \alpha_{k+1}$ de V. On a dans ce cas $\beta'_{k-1} = \beta_{k-1}$ et $\alpha'_{k+1} = \alpha_k \prec \alpha_{k+1}$, et ceci s'explique par le fait que : $\forall j \in P_{t_k} \Rightarrow i \in P_{t_{k-1}}$ ou $i \in P_{t_{k+1}}$ d'après la définition d'un sommet potentiel non générateur t_k. D'où $\sigma_\Delta(V \setminus \{t_k\}) \Subset \sigma_\Delta(V)$. Notons que si l'on s'intéresse au cas particulier de la sélection $V' = V \setminus \{t_m\}$ qui peut présenter un nouveau sommet, on a $\sigma_\Delta(V \setminus \{t_m\}) \Subset \sigma_\Delta(V)$. □

Proposition 3.10. *Étant donnés un ensemble V de n travaux et son graphe d'intervalles sommet-base associé G, si un nœud t_k de G est un sommet générateur alors, $\sigma_\Delta(V \setminus \{t_k\})$ est non compatible avec $\sigma_\Delta(V)$.*

Démonstration. Supposons que pour un ensemble V de n travaux et m sommets, le sommet t_k est générateur d'au moins un sommet u (successeur propre de t_k). Soit v le successeur voisin de t_k juste après u. le nœud u n'est donc relié qu'à t_k puisqu'il est successeur propre de t_k. Plus explicitement, la séquence $\sigma_\Delta(V)$ est de la forme $\alpha_0 \prec t_0 \prec ...(\alpha_u \prec u \prec \alpha_v \prec v) \prec t_k \prec \beta_k \prec t_{k+1}... \prec \beta_m$. La séquence $\sigma_\Delta(V \setminus \{t_k\})$ s'écrit $\alpha_0 \prec s_0 \prec ...\alpha_u \prec (u \prec \beta_u \prec \alpha_v \prec v) \prec t_{k+1}... \prec \beta_m)$, voir Figure 3.15. Il est clair que la sous-séquence $u \prec \beta_u \prec \alpha_v \prec v$ tirée de $\sigma_\Delta(V \setminus \{t_k\})$ n'est pas incluse dans $\sigma_\Delta(V)$, d'où $\sigma_\Delta(V \setminus \{s_k\})$ non compatible avec $\sigma_\Delta(V)$.

□

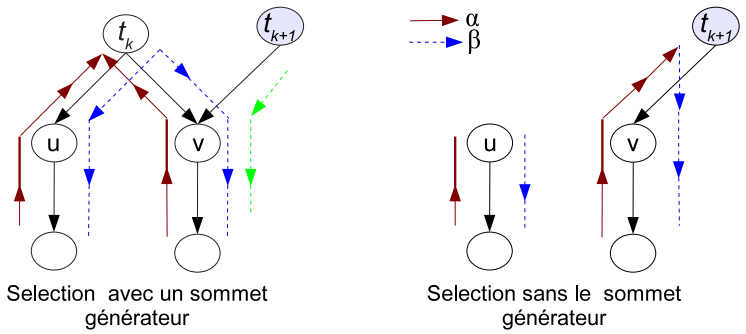

FIG. 3.15: Exemple de sous-séquence sans sommet générateur

Illustrons la Proposition 3.10 sur un exemple simple V de 4 travaux a, u, v, t_k ayant l'ordre total suivant : $r_a < r_u < r_v < r_{t_k} < d_{t_k} < d_u < d_v < d_a$. Le sommet t_k est générateur des sommets u et v. La séquence maître-pyramide correspondante est $\sigma_\Delta(V) = a \prec u \prec v \prec t_k \prec u \prec v \prec a$. Considérons le sous ensemble $V \setminus \{t_k\}$ constitué des deux sommets générés u et v. $\sigma_\Delta(V \setminus \{t_k\}$ s'écrit $a \prec u \prec a \prec v \prec a)$. On a bien $\sigma_\Delta(V \setminus \{t_k\})$ non compatible avec $\sigma_\Delta(V)$, puisque la séquence $u \prec a \prec v$ tirée de $\sigma_\Delta(V \setminus \{t_k\})$ n'est pas incluse dans $\sigma_\Delta(V)$.

Proposition 3.11. *Étant donné un ensemble V de n travaux et son graphe d'intervalles sommet-bases associé G, si un nœud t_k de G est sommet générateur sans base commune alors, $\sigma_\Delta(V \setminus \{t_k\}) \Subset \sigma_\Delta(V)$ dans le sens de la dominance, i.e. $\sigma_\Delta(V \setminus \{t_k\})$ est dominée par $\sigma_\Delta(V)$.*

Démonstration. Considérons que dans $\sigma_\Delta(V)$, le sommet t_k est générateur d'un ensemble $M = \{m_1, ..., m_{p,p>1}\}$ de sommets potentiels. Sachant d'une part que t_k est générateur, il est montré d'après la Proposition 3.10 que la sous-séquence $m_i \prec \beta_{m_i} \prec \alpha_{m_{i+1}} \prec m_{i+1}$ tirée de $\sigma_\Delta(V \setminus \{t_k\})$ avec $i, i+1 \in M$ n'est pas incluse dans $\sigma_\Delta(V)$. D'autre part, du fait que t_k est sommet générateur sans base commune, cela implique par définition que $\forall i, j \in P_{t_k} / r_i < r_j \Rightarrow d_i < d_j$ alors $\nexists k \in P_{t_k}$ tel que $r_k < r_i$ et $d_k > d_j$. Autrement dit, si on considère l'ensemble $V \setminus \{t_k\}$ ainsi que l'ensemble des sommets générés, on peut écrire $r_{m_p} < d_{m_1}(1)$, et que si $k \in P_{m_i} \Rightarrow k \in P_{m_j} \forall k \in V$, $\forall m_i, m_j \in M$ avec $m_i \neq m_j(2)$. Par considération de ces deux conditions (1) et (2), et d'après la Proposition 3.5 de la Section 3.5, toute sous-séquence $m_i \prec \beta_{m_i} \prec \alpha_{m_{i+1}} \prec m_{i+1}$ de $\sigma_\Delta(V \setminus \{t_k\})$ non incluse dans $\sigma_\Delta(V)$ est dominée par une sous séquence tirée de $\alpha_{m_i} \prec m_i \prec \alpha_{m_{i+1}} \prec m_{i+1} \prec \beta_{m_i} \prec m_i \prec \beta_{m_{i+1}} \prec m_{i+1}$, cette dernière étant incluse dans $\sigma_\Delta(V)$. D'ou $\sigma_\Delta(V \setminus \{t_k\}) \Subset \sigma_\Delta(V)$ au sens de la dominance. □

À titre illustratif, considérons un exemple simple de 5 travaux ayant l'ordre total suivant : $r_a < r_u < r_b < r_v < r_{t_k} < d_{t_k} < d_u < d_a < d_v < d_b$. Le sommet t_k est générateur des sommets u et v. La séquence maître-pyramide correspondante est $\sigma_\Delta(V) = a \prec u \prec b \prec$

$v \prec t_k \prec u \prec a \prec v \prec b$. Considérons le sous ensemble $V \setminus \{t_k\}$ constitué de deux sommets u et v. On a $\sigma_\Delta(V \setminus \{t_k\}) = a \prec u \prec a \prec b \prec v \prec b$. La sous-séquence $u \prec a \prec b \prec v$ tirée de $\sigma_\Delta(V \setminus \{t_k\})$ n'est pas incluse dans $\sigma_\Delta(V)$. Si on considère l'ensemble $V \setminus \{t_k\}$, constitué de deux sommets sans bases communes, on sait d'après la Proposition 3.5 que $u \prec a \prec b \prec v$ est dominée soit $a \prec u \prec b \prec v$ ou bien $u \prec a \prec v \prec b \prec$, toutes les deux incluses respectivement dans $\sigma_\Delta(V \setminus \{s_k\})$ et $\sigma_\Delta(V)$. Ainsi on a $\sigma_\Delta(V)$ compatible avec $\sigma_\Delta(V \setminus \{t_k\})$ au sens où les séquences omises sont dominées.

Considérant les différents résultats donnés dans cette section, le théorème suivant montre sous quelles conditions, la condition de dominance donnée par le Théorème des pyramides, valide vis-à-vis de l'admissibilité, est aussi valide vis-à-vis de la minimisation du nombre de travaux en retard.

Théorème 3.6. *Étant donnés un problème $1|r_i|\sum U_i$ et son graphe d'intervalles sommet-base associé G, si tout nœud k (resp. ensemble de nœuds e_k) de G est tel que k (resp. e_k) est non générateur, ou bien générateur sans base commune, alors la séquence maître-pyramide $\sigma_\Delta(V) = \alpha_0 \prec t_0 \prec \beta_0, ..., \alpha_m \prec s_m \prec \beta_m$, construite à partir des n travaux de V, caractérise l'ensemble S_{dom} des séquences dominantes vis-à-vis du problème de la minimisation du nombre de travaux en retard.*

Démonstration. Le Théorème 3.6 stipule les conditions pour lesquelles, la séquence maître-pyramide $\sigma_\Delta(V)$ caractérisant l'ensemble de toutes les séquences dominantes à n travaux S_{dom}, caractérise également les séquences dominantes des sous-ensembles à k travaux, $k \in \{1...n-1\}$. Ainsi, pour chaque sélection $V \setminus S$ et sa séquence maître-pyramide $\Subset \sigma_\Delta(V \setminus S)$ correspondante, nous étudions, en fonction de la nature des nœuds S éliminés de V, les cas où y a compatibilité entre $\sigma_\Delta(V \setminus S)$ et $\sigma_\Delta(V)$. Nous distinguons alors tous les cas possibles :

- si S est un sous-ensemble de nœuds du graphe G de niveaux supérieurs à zéro alors d'après la Proposition 3.8, $\sigma_\Delta(V \setminus S) \Subset \sigma_\Delta(V)$.
- si S est un nœud de niveau zero, il est alors noté t_k et est un sommet au sens de la Définition 3.1. t_k peut alors être sommet potentiel ou sommet non potentiel. Si t_k n'est pas un sommet potentiel, il est évident que son élimination de V revient à l'éliminer de la seule position qu'il occupe dans $\sigma_\Delta(V)$, d'où $\sigma_\Delta(V \setminus S) \Subset \sigma_\Delta(V)$. Par contre, si t_k est un sommet potentiel, on peut conclure que $\sigma_\Delta(V \setminus S) \Subset \sigma_\Delta(V)$ si seulement t_k n'est pas générateur (d'après la Proposition 3.9) ou bien si t_k est générateur sans bases commune (d'après la Proposition 3.11). Il est montré d'après la Proposition 3.10 qu'il y a incompatibilité entre $\sigma_\Delta(V \setminus S)$ et $\sigma_\Delta(V)$ si t_k est générateur.
- si S est un sous-ensemble de nœud de niveau supérieur ou égal à zéro, dont un nœud t_k est sommet potentiel générateur, alors on note $S = \{t_k \cup pred\{t_k\}\}$. La sous séquence à éliminer de V est donc constituée d'un sommet potentiel générateur ainsi

que tous ses prédécesseurs. Ce cas est équivalent au point cité juste au dessus, en supposant que $V = V \setminus pred\{t_k\}$ et ainsi $S = t_k$.

- S est constitué non pas d'un sommet générateur, mais d'un ensemble de sommets générateurs $e_k = \{t_{k_1}, ..., t_{k_p}\}$. Éliminer tous les nœuds e_k de V excepté un sommet t_{k_p} revient donc à éliminer $\{t_{k_1}, ..., t_{k_{p-1}}\}$ de la seule position qu'ils occupent dans $\sigma_\Delta(V)$. La compatibilité de $\sigma_\Delta(V)$ avec $\sigma_\Delta(V \setminus e_k)$ revient à étudier la compatibilité de $\sigma_\Delta(V')$ avec $\sigma_\Delta(V' \setminus t_{k_p})$ sachant $(V' = V \setminus \{t_{k_1}, ..., t_{k_{k-1}}\}$: un cas étudié plus haut.

□

De façon générale, nous avons montré à travers le Théorème 3.6 que la séquence maître-pyramide construite pour les n travaux de V et caractérisant l'ensemble S_{dom} des séquences dominantes, n'est pas dominante vis-à-vis du critère de minimisation du nombre de travaux en retard. Toutefois, une solution optimale au problème de $\sum U_i$ est incluse dans S_{dom} si tout élément du graphe d'intervalles sommet-bases associé à V n'est pas sommet générateur, ou bien, est sommet générateur sans base commune. Il n'est donc pas possible de caractériser les séquences dominantes à l'aide d'une seule et unique séquence maître-pyramide. Dans ce cas en effet, il faut considérer la séquence maître-pyramide obtenue en enlevant le sommet (ensemble) générateur(s) en plus de la séquence maître-pyramide qui comprend tous les nœuds. La démarche de détermination de toutes les séquences maîtres pyramides est illustrée dans l'exemple suivant.

3.6.5 Exemple illustratif

Illustrons la procédure de détermination de l'ensemble des séquences maître-pyramides sur l'exemple du graphe sommet-base donné sur la Figure 3.13 et auquel nous avons ajouté d'autres niveaux de bases, voir Figure 3.16. Pour l'ensemble $V = \{1, 2, ..., 12\}$ de 12 travaux, nous constatons que seuls les nœuds 3 et 4 sont générateurs, on note $G = \{3, 4\}$. Tous les autres sommets potentiels 2 et $\{5, 6\}$ sont candidats à devenir éventuellement générateurs suite à l'élimination des nœuds 3 ou 4.

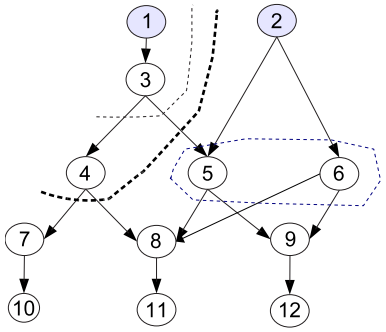

FIG. 3.16: graphe d'intervalles sommets-bases avec sommets générateurs

La séquence maître-pyramide relative à V est :

$$\sigma_\Delta(V) = (10, 7, 11, 8, 4, 12, 9, 5, 3, \underline{\mathbf{1}, 3, 4, 7, 10, 5, 8, 11, 9, 12, 6, \mathbf{2}}, 5, 6, 8, 11, 9, 12)$$

Les sommets sont en gras. Si on considère la sélection de travaux de V sans le sommet générateur 3 et son prédécesseur 1, seul le nœud 4 devient à nouveau générateur de la nouvelle selection. On note que le sommet 3 est générateur du sommet 4 avec la base commune 8. Ainsi, la séquence maître-pyramide correspondante à $V \setminus \{1, 3\}$ s'écrit :

$$\sigma_\Delta(V \setminus \{1, 3\}) = (10, 7, 11, 8, \underline{\mathbf{4}, 7, 10, 8, 11, 12, 9, 5, 6, \mathbf{2}}, 5, 6, 8, 11, 9, 12)$$

Comme on peut le remarquer, la sous-séquence $(8, 11, 12, 9)$ tirée de $\sigma_\Delta(V \setminus \{1, 3\})$ et placée entre les sommets 4 et 2 n'est pas incluse dans $\sigma_\Delta(V)$, d'où incompatibilité de $\sigma_\Delta(V \setminus \{1, 3\})$ avec $\sigma_\Delta(V)$. Examinons maintenant la sélection de V sans le sommet générateur 4 ainsi que ses prédécesseurs 3 et 1. Dans ce cas l'ensemble des nœuds $\{5, 6\}$ devient générateur. La séquence maître-pyramide correspondante est :

$$\sigma_\Delta(V \setminus \{1, 3, 4\}) = (10, \underline{\mathbf{7}, 10, 11, 8, 12, 9, 5, 6, \mathbf{2}}, 5, 6, 8, 11, 9, 12)$$

On constate bien que $(7, 10, 11, 8)$ tirée de $\sigma_\Delta(V \setminus \{1, 3, 4\})$ et placée avant le sommet 2 n'est pas incluse dans $\sigma_\Delta(V \setminus \{1, 3\})$ avant le sommet 2. Toutefois, sachant que le sommet 4 est générateur sans base commune, on sait d'après la Proposition 3.5 que la séquence $(7, 10, 11, 8)$ est dominée par $(10, 7, 11, 8)$ ou bien $(7, 10, 8, 11)$ toutes les deux incluses dans $\sigma_\Delta(V \setminus \{1, 3\})$ et avant le sommet 2. On a alors $\sigma_\Delta(V \setminus \{1, 3, 4\}) \Subset \sigma_\Delta(V \setminus \{1, 3\})$ dans le sens de la dominance.

Enfin, considérons la sélection de travaux sans l'ensemble de sommets générateurs $(5, 6)$. Puisque les sommets 5 et 6 ne possèdent pas de base commune, on conclut que $\sigma_\Delta(V \setminus \{1, 3, 4, 5, 6, 2\}) \Subset \sigma_\Delta(V \setminus \{1, 3, 4\}) \Subset \sigma_\Delta(V \setminus \{1, 3\})$ dans le sens de la dominance.

Nous avons donc énuméré toutes les séquences maîtres-pyramides non compatibles entre elles. Pour ce faire, nous avons éliminé tour à tour dans le graphe sommet-base les sommets (ensembles) générateurs ainsi que leur ascendance, puis avons construit à chaque fois la séquence maître-pyramide pour le graphe (sous-problème) restant.

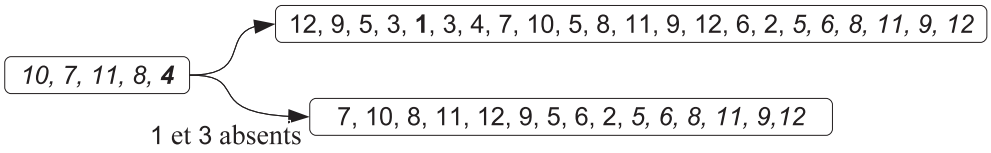

FIG. 3.17: Les séquences maîtres-pyramides relatives à l'exemple de la Figure 3.16

La Figure 3.17 synthétise l'ensemble des séquences maître-pyramides relatif à l'exemple de la Figure 3.16. En effet, on constate que $\sigma_\Delta(V)$ caractérise toutes les séquences dominantes incluant toujours tous les nœuds, (soit 7776 séquences dominantes de 12 travaux) alors que $\sigma_\Delta(V \setminus \{1, 3\})$ caractérise quant à elle toutes les séquences dominantes n'incluant jamais les travaux 1 et 3 (soit 576 séquences dominantes de 10 travaux). Au total, 8352 séquences dominantes sont caractérisées.

Notons enfin que la solution optimale est une séquence dominante parmi toutes celles caractérisées. Elle est tirée, de $\sigma_\Delta(V)$ si elle contient au plus un travail en retard, ou bien de $\sigma_\Delta(V)$ ou de $\sigma_\Delta(V \setminus \{1,3\})$ si la solution optimale contient au moins deux travaux en retard. Comme remarque, on signale que les travaux en retard sont supposés avoir une durée opératoire nulle dans la séquence dominante.

3.7 Conclusion

Dans ce chapitre, nous nous sommes intéressés à un théorème de dominance (Théorème de pyramides) formulé dans les années quatre-vingt pour limiter la complexité algorithmique liée à la vérification de la consistance d'un problème à une machine (existence de solutions admissibles). L'avantage de ce théorème réside dans sa capacité à caractériser sans les énumérer, un ensemble dominant de séquences grâce à un ordre partiel et en considérant exclusivement que les relations d'ordre entre les dates de disponibilité et les dates échues des travaux.

Partant de l'ordre partiel entre les travaux, nous avons introduit la notion de séquence maître-pyramide à partir de laquelle peuvent être extraites toutes les séquences dominantes que le Théorème des pyramides caractérise. Ensuite, nous avons montré comment caractériser, étant donnée une séquence maître-pyramide et son ensemble dominant de séquences S_{dom}, l'ensemble des ordres totaux (ensemble de problèmes à structure pyramidales distinctes) dont l'ensemble dominant de séquences est le même (i.e. S_{dom}). Puis, sous certaines conditions, nous avons énoncé une nouvelle proposition qui permet de restreindre d'avantage l'ensemble dominant de séquences.

Sachant que la condition analytique énoncée par le théorème des pyramides permet de caractériser l'ensemble dominant de séquences pour la recherche, si elle existe, de la solution admissible, nous avons dans ce chapitre cerner puis proposer des conditions numériques de dominance. Ces conditions permettent de statuer sur l'admissibilité ou non d'une séquence dominante (cf. Théorème 3.4) et d'identifier, si elle existe, la séquence admissible (i.e. la plus dominante) parmi toutes les séquences dominantes (cf. Corollaire 3.4).

Enfin, nous nous sommes focalisés sur l'étude de la condition de dominance du théorème des pyramides au cas du problème de minimisation du nombre de travaux en retard. Nous avons montré sous quelles hypothèses et de quelle manière le Théorème des pyramides peut être utilisé pour le problème de minimisation du nombre de travaux en retard. Ainsi, dans le cas où le théorème n'est pas dominant vis-a-vis de $\sum U_i$, plus d'une séquences maîtres-pyramides sont nécessaires pour caractériser l'ensemble dominant de séquences pour le problème $\sum U_i$. Notons qu'il existe autant de séquences maîtres que de sommets générateurs, et que la complexité liée à la recherche de tous les sommets générateurs est

exponentielle.

Ce chapitre constitue, principalement, une étude sur la condition de dominance établie par Erschler et al. en considérant tour à tour les problèmes d'admissibilité et de minimisation du nombre de travaux en retard. Les différents résultats trouvés servent de base pour le chapitre suivant.

Chapitre 4

Une nouvelle formulation PLNE

Dans ce chapitre, en se basant sur les conditions numériques de dominance établies dans la Section 3.5, nous montrons comment modéliser par PLNE le problème d'admissibilité pour la recherche de la solution la plus dominante, et démontrons que la solution trouvée est aussi optimale vis-à-vis du problème de minimisation du retard algébrique maximal. Ensuite, en considérant le critère de minimisation du nombre de travaux en retard, nous montrons comment adapter le modèle PLNE initial pour la détermination d'une borne supérieure et d'une borne inférieure au problème $1|r_j|\sum U_j$.

4.1 Problème d'admissibilité

Dans la Section 3.4, nous avons décrit une condition nécessaire et suffisante d'admissibilité d'une séquence, permettant de dériver une condition analytique de dominance.

Nous rappelons que les séquences dominantes sont considérées comme étant des séquences "plus admissibles" que les autres dans le sens où si aucune séquence dominante admissible n'existe, alors il est certain qu'il n'existe aucune séquence admissible (solution).

Nous avons considéré par S_{dom} l'ensemble des séquences dominantes que le Théorème des pyramides caractérise. En se limitant à ce seul domaine S_{dom}, de nouvelles conditions numériques de dominance ont été établies dans la Section 3.5. Ces conditions permettant de restreindre d'avantage l'ensemble S_{dom} en identifiant une séquence dominante particulière de S_{dom} qu'on note σ^*. C'est une séquence qui vérifie le critère $D^* - R^* = \max_{\sigma_i \in S_{dom}}(D^i - R^i)$ pour le cas d'une structure mono-pyramidale et $\min_{k=1..m}(D^*_k - R^*_k - p_{t_k}) = \max_{\sigma_i \in S_{dom}}(\min_{k=1..m}(D^i_k - R^i_k - p_{t_k}))$ pour le cas d'une structure d'intervalles à plusieurs pyramides.

Si cette séquence σ^*, que nous appelons la séquence "la plus dominante" (cf. Figure 4.1), n'est pas admissible i.e. $D^* - R^* < p_t$, t étant le sommet de la pyramide, ou bien

$\min_{k=1..m}(D_k^* - R_k^* - p_{t_k}) < 0$, alors nous avons la garantie que le problème n'admet pas de solution admissible.

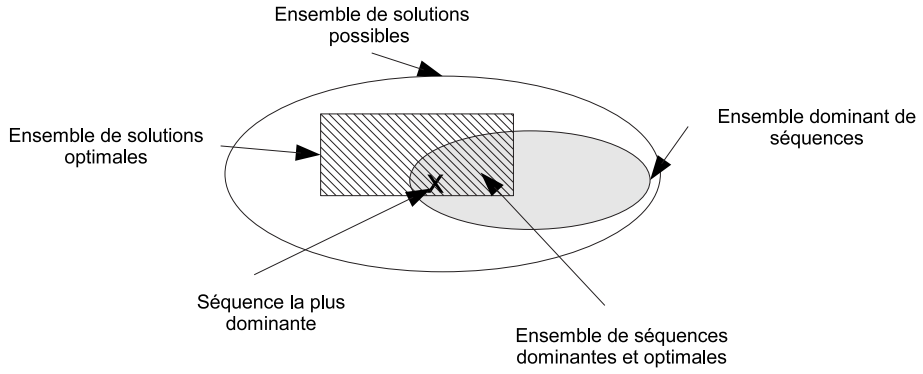

FIG. 4.1: La séquence la plus dominante dans l'ensemble des solutions possibles

Dans la suite, nous allons montrer que la détermination de la séquence la plus dominante peut se modéliser par un programme linéaire en nombres entiers.

4.1.1 Cas mono-pyramidal

Cette section est consacrée au cas mono-pyramidal, où la structure d'intervalles associée au problème à résoudre ne contient qu'un et un seul sommet $t \in V$, et donc une et une seule pyramide P.

Toute séquence admissible, si elle existe, est une séquence dominante qui respecte le Théorème 3.2. L'ensemble S_{dom} de séquences dominantes que ce théorème caractérise est de la forme $\alpha \prec t \prec \beta$, où les travaux appartenant à la sous-séquence α sont ordonnés selon l'ordre des r_j croissant, et ceux de β dans l'ordre des d_j croissant. L'étude de l'admissibilité du problème requiert ainsi l'exploration d'un ensemble de 2^n séquences dominantes au lieu de $(n+1)!$ séquences possibles. La comparaison numérique des séquences dominantes par le biais de la condition numérique de dominance permet de définir la séquence la plus dominante. Ceci peut se faire par une formulation mathématique du problème de recherche de cette séquence, sous la forme d'un programme linéaire en nombres entiers défini comme suit et qu'on note PLNE1 :

$$\max\ z = D - R$$

$$\text{s.t.}\quad \begin{cases} R \geq r_t & (5.1)\\ R \geq r_i + \sum_{\{j\in V\setminus\{t\}|r_j\geq r_i\}} p_j x_j^+ \ , \ \forall i \in V\setminus\{t\} & (5.2)\\ D \leq d_t & (5.3)\\ D \leq d_i - \sum_{\{j\in V\setminus\{t\}|d_j\leq d_i\}} p_j x_j^- \ , \ \forall i \in V\setminus\{t\} & (5.4)\\ x_i^- + x_i^+ = 1 \ , \ \forall i \in V\setminus\{t\} & (5.5)\\ x_i^- \ , \ x_i^+ \in \{0,1\} \ , \ \forall i \in V\setminus\{t\} & \\ D \ , \ R \in \mathbb{Z} & \end{cases}$$

Dans la formulation ci-dessus, la variable x_i^+(resp x_i^-) est égale à 1 si le travail i est séquencé à gauche du sommet (resp à droite du sommet). Les contraintes (5.1) et (5.2) relatives à R, correspondant à la formule 3.3 dans le chapitre précédent, permettent de définir la date de début au plus tôt du sommet de la pyramide, tout en assurant que les travaux placés à gauche du sommet (i.e., α) débutent leur exécution après leur date de disponibilité. De même, les contraintes (5.3) et (5.4) relatives à D et correspondant à la formule 3.4 dans le chapitre précédent définissent, quant à elles, la date de fin au plus tard du sommet t tout en assurant que les travaux placés à droite de ce sommet (i.e., β) finissent leur exécution avant leur date échue. Enfin, la contrainte (5.5) impose à ce que que tout travail i de la pyramide P soit sequencé, soit à gauche ou bien à droite du sommet, mais pas dans les deux positions en même temps.

La valeur optimale $z^* = D^* - R^*$ du PLNE1 définit alors la marge libre maximale qu'offre la séquence la plus dominante $\alpha^* \prec t \prec \beta^*$ au sommet t, en considérant que tous les travaux dans α sont calés après leur date de disponibilité, et les travaux dans β avant leur date échue tout en omettant de considérer la durée opératoire du sommet t. Cette séquence dominante "domine" toutes les autres séquences de S_{dom} dans le sens du Théorème 3.3, c'est à dire si $z^* < p_t$ alors nous avons la garantie qu'il n'existe pas de solution admissible au problème.

4.1.2 Cas multi-pyramidal

4.1.2.1 Un modèle PLNE pour le cas multi-pyramidal indépendant

Avant de considérer le cas général du problème d'ordonnancement à une machine, concentrons nous sur le problème particulier V caractérisé par m sommets i.e. m pyramides, avec l'hypothèse que tout travail non sommet appartient uniquement à une et une seule pyramide i.e., si $i \in P_k \Rightarrow i \notin P_{k'}$, $k \neq k'$ $\forall k \in \{1,\ldots,m\}$ et $i \in V$ (les pyramides

sont dites indépendantes). Ainsi, toute séquence dominante respectant le Théorème des pyramides est de la forme $\alpha_1 \prec t_1 \prec \beta_1 \prec \cdots \prec \alpha_m \prec t_m \prec \beta_m$, où t_k est le sommet de la pyramide P_k, et les travaux de α_k et β_k appartiennent uniquement à la pyramide P_k.

Le problème de recherche de la séquence de travaux la plus dominante pour le problème multi-pyramidal indépendant peut être alors décomposé en m sous-problèmes mono-pyramidaux interdépendants. L'interdépendance de ces sous-problèmes vient du fait que les sous-séquences $\alpha_k \prec t_k \prec \beta_k$ pour tout $k \in \{1...m\}$ ne doivent pas se chevaucher entre elles, i.e. que la date de fin $\alpha_k \prec t_k \prec \beta_k$ doit être inférieure à la date de début de la séquence $\alpha_{k+1} \prec t_k \prec \beta_{k+1}$.

Étant donné une séquence dominante σ, la condition d'interdépendance entre les sous séquences de σ peut être assurée en imposant à chaque travail $j \in P_k$ des contraintes sur les dates de début r_j et de fin d_j de j (cf. Figure 4.2). Ces dates ont été définies dans la Section 3.5.2. Nous les rappelons ci-bas :

$$\begin{cases} r_j = \max(r_j, eft_{k-1}) \\ \\ d_j = \min(d_j, lst_{k+1}) \end{cases} \tag{4.6}$$

Avec eft_{k-1} (resp. lst_{k+1}) désignant la date de fin au plus tôt de la séquence β_{k-1} (resp. la date de début au plus tard de séquence α_{k+1}).

Plus explicitement, en imposant que tout travail j de V soit positionné dans sa fenêtre temporelle $[r_j,\ d_j]$, la date de début (resp. date de fin) de j est contrainte non seulement par sa date de disponibilité (resp. sa date d'échéance) mais aussi par la date de fin au plus tôt (resp. date de début au plus tard) de la séquence qui le précède (resp. succède).

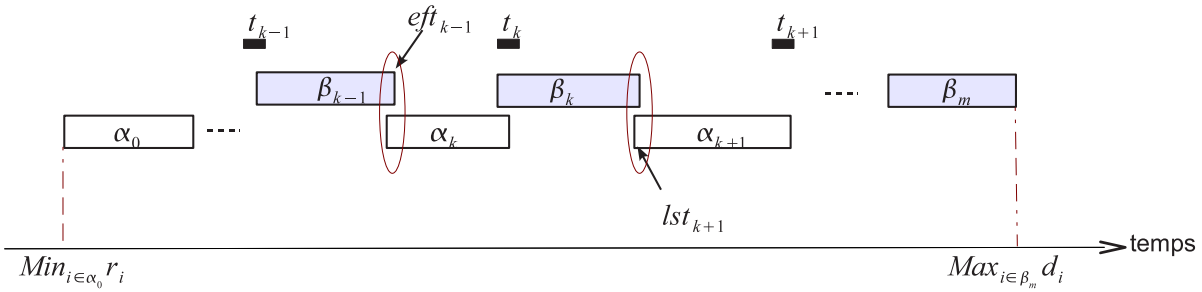

FIG. 4.2: Interdépendance des sous-séquences d'une séquence dominante σ

Considérons à présent les variables binaires x_j^+ et x_j^- définies dans la section précédente. Les expressions mathématiques des variables eft_{k-1} et lst_{k+1} données par le système d'in-

équations 3.5 dans la Section 3.5.2 s'écrivent alors :

$$\begin{cases} eft_{k-1} = R_{k-1} + p_{t_{k-1}} + \sum_{j \in P_{k-1}} (p_j x_j^-) \\ \\ lst_{k+1} = D_{k+1} - p_{t_{k+1}} - \sum_{j \in P_{k+1}} (p_j x_j^+) \end{cases} \tag{4.7}$$

Ainsi, le problème de la recherche d'une séquence de travaux la plus dominante peut être formulé par le problème linéaire en nombres entiers (PLNE2) suivant :

$$\max \; z = \min_{k=1,\dots,m}(D_k - R_k - p_{t_k})$$

$$\text{s.t.} \quad \begin{cases} R_k \geq r_{t_k} & , \; \forall k \in [1\ m] & (5.8) \\ R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_j^+ & , \; \forall i \in V \backslash \{1..m\}, i \in P_k & (5.9) \\ R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x_j^- + p_{t_{k-1}} & & \\ \quad + \sum_{\{j \in P_k\}} p_j x_j^+ & , \; \forall k \in [2\ m] & (5.10) \\ D_k \leq d_{t_k} & , \; \forall k \in [1\ m] & (5.11) \\ D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_j^- & , \; \forall i \in V \backslash \{1..m\}, i \in P_k & (5.12) \\ D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x_j^+ - p_{t_{k+1}} & & \\ \quad - \sum_{\{j \in P_k\}} p_j x_j^- & , \; \forall k \in [1\ (m-1)] & (5.13) \\ x_i^- + x_i^+ = 1 & , \; \forall i \in V \backslash \{1..m\} & (5.14) \\ x_i^- \ , \ x_i^+ \in \{0,1\} & , \; \forall i \in V \backslash \{1..m\} & \\ D_k \ , \ R_k \in \mathbb{Z} & , \; \forall k \in [1\ m] & \end{cases}$$

Comme pour le modèle PLNE1, la variable binaire x_j^+ (resp. x_j^-) définie dans le modèle PLNE2 prend la valeur 1, si le travail j, appartenant uniquement à la pyramide P_k, est séquencé dans α_k(resp. β_k) et seulement dans α_k (resp. β_k). L'affectation d'un travail à une et une seule pyramide est assurée par la contrainte (5.14) (i.e. $x_i^- + x_i^+ = 1$). La contrainte sur R_k permet de définir la date de début au plus tôt du sommet t_k tout en respectant sa date de disponibilité ainsi que la date de fin des travaux qui le précèdent (travaux de la même pyramide P_k et ceux de la pyramide P_{k-1}).

Par définition :

$$R_k = \max(r_{t_k}, eft_{k-1} + \sum_{\{j \in \alpha_k\}} p_j, \max_{i \in \alpha_k}(r_i + p_i + \sum_{\{j \in \alpha_k | i \prec j\}} p_j)) \tag{4.15}$$

où eft_{k-1} formulée dans 3.5, définit la date de fin au plus tôt de la sous-séquence β_{k-1}.

Ainsi les contraintes (5.8), (5.9) et (5.10) formalisées selon l'équation 4.15 permettent de déterminer la valeur de R_k.

De la même manière, la contrainte D_k délimite la date de fin au plus tard du sommet t_k tout en assurant que les travaux qui le succèdent soient réalisés à temps.

Par définition :

$$D_k = \min(d_{t_k}, lst_{k+1} - \sum_{\{j \in \beta_k\}} p_j, \min_{i \in \beta_k}(d_i - p_i - \sum_{\{j \in \beta_k | j \prec i\}} p_j)) \tag{4.16}$$

où lst_{k+1} formulée dans 3.5 définit la date de début au plus tôt de la sous-séquence α_{k+1}. Ainsi les contraintes (5.11), (5.12) et (5.13) formalisées selon l'équation 4.16 permettent de déterminer la valeur de D_k.

Enfin, la valeur optimale de $z^* = \min_{k=1,\dots,m}(D_k - R_k - p_{t_k})$ définit la marge libre maximale qu'offre la séquence la plus dominante $\alpha_1^* \prec t_1 \prec \beta_1^* \dots \alpha_m^* \prec t_m \prec \beta_m^*$, en considérant que l'ensemble V des travaux est séquencé à l'heure. Cette séquence dominante "domine" toutes les autres séquences de S_{dom} dans le sens où, si la marge libre maximale z^* qu'offre cette séquence est négative, alors il est certain d'après le Corollaire 3.4 du Chapitre 3, qu'il n'existe pas de solution admissible au problème.

4.1.2.2 Un modèle PLNE pour le cas multi-pyramidal quelconque

Dans cette partie, nous nous intéressons au problème général d'ordonnancement à une machine où tout travail j non-sommet peut appartenir à plusieurs pyramides : seule différence avec le problème multi-pyramidal indépendant. Notons $u(j)$ (resp. $v(j)$) l'indice de la première (resp. la dernière) pyramide à laquelle appartient le travail j. Par application du Théorème des pyramides, toute séquence dominante de travaux est de la forme $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \cdots \prec \alpha_m \prec t_m \prec \beta_m$, avec t_k, sommet de la pyramide P_k, et j un travail non-sommet appartenant à α_k ou β_k de toute pyramide P_k tel que $u(j) \leq k \leq v(j)$.

Sachant que la solution finale est une séquence qui impose une position et une seule pour chaque travail dans cette séquence, i.e., chaque travail j sera affecté à une et une seule pyramide, le problème devient alors équivalent au problème multi-pyramidal indépendant (résolu par le PLNE2). Le nouveau modèle est donné en transformant les variables binaires (x_j^+) et (x_j^-) utilisées dans le modèle PLNE2 permettant d'assigner un travail j à droite ou à gauche du sommet de la pyramide (la seule) à laquelle il appartient, par les nouvelles variables (x_{kj}^+) et (x_{kj}^-) permettant d'affecter le travail j seulement à droite ou à gauche du sommet de l'une et seulement une des pyramide P_k, $u(i) \leq k \leq v(i)$, à laquelle il appartient.

La formulation générale du problème (PLNE3) est donné comme suit :

$$\max\ z = \min_{k=1,\dots,m}(D_k - R_k - p_{t_k})$$

$$\text{s.t.}\quad \begin{cases}
R_k \geq r_{t_k} & , \ \forall k \in [1\ m] & (5.16)\\
R_k \geq r_i + \sum_{\{j\in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \ \forall i \in V \setminus \{1...m\}, k = u(i) & (5.17)\\
R_k \geq R_{k-1} + \sum_{\{j\in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j\in P_k\}} p_j x^+_{kj} & , \ \forall k \in [2\ m] & (5.18)\\
D_k \leq d_{t_k} & , \ \forall k \in [1\ m] & (5.19)\\
D_k \leq d_i - \sum_{\{j\in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \ i \in V \setminus \{1...m\}, k = v(i) & (5.20)\\
D_k \leq D_{k+1} - \sum_{\{j\in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j\in P_k\}} p_j x^-_{kj} & , \ \forall k \in [1\ (m-1)] & (5.21)\\
\sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) = 1 & , \ \forall i \in V \setminus \{1...m\} & (5.22)\\
x^-_{ki}\ ,\ x^+_{ki} \in \{0,1\} & , \ \forall k \in [1\ m], \forall i \in P_k & \\
D_k\ ,\ R_k \in \mathbb{Z} & , \ \forall k \in [1\ m] &
\end{cases}$$

Dans le PLNE3, modèle similaire au PLNE2, la variable binaire x^+_{kj} (resp. x^-_{kj}) prend la valeur 1 si le travail non sommet j appartenant à la pyramide P_k : $u(j) \leq k \leq v(j)$, est séquencé dans α_k (resp. β_k). Les contraintes sur R_k et D_k, i.e. les contraintes(5.16)-(5.21), sont identiques à celles définies dans le PLNE2. Enfin, la contrainte (5.22) spécifique à ce modèle permet d'assurer que tout travail prend une et une seule position dans une pyramide et une seule.

Calculons à présent le nombre de variables et de contraintes requises dans le modèle PLNE3. Nous avons m contraintes de type (5.16) et (5.19), $(n-m)$ contraintes de type (5.17) et (5.20), $(m-1)$ contraintes de types (5.18) et (5.21) et aussi $(n-m)$ contraintes de type (5.22). Ainsi, le nombre total de contraintes s'élève à $(m+3n-2)$. Intéressons nous à présent au nombre de variables. Il existe $(2\sum_{k=1}^{m}(n_k))$ variables binaires où n_k désigne le nombre de travaux non sommets assignés à une pyramide P_k. Nous comptons aussi $(2m)$ variables réelles pouvant être déduites directement des variables binaires.

4.2 Minimisation du plus grand retard algébrique

Nous avons dans la section précédente modélisé le problème de détermination de la séquence la plus dominante permettant de conclure, selon la valeur optimale du critère z,

quant à l'existence ou non d'une solution admissible. Cette valeur, dans le cas où elle est positive, implique que tous les travaux sommets sont séquencés à l'heure sachant que tous les autres travaux non sommets le sont aussi, i.e., la séquence optimale est admissible.

Dans cette section, nous allons montrer que la solution optimale du problème de détermination de la séquence la plus dominante est aussi optimale vis-à-vis de la minimisation du plus grand retard algébrique.

Nous rappelons que le Théorème des pyramides énoncé dans la Section 3.2.2, dominant vis-à-vis du problème d'admissibilité, est aussi dominant pour le critère $L_{\max}$ [Briand *et al.* 03b].

Définition 4.1. *Étant donné une séquence S à n travaux, le retard algébrique d'un travail i non-sommet de P_k est $L_i = r_i - lst_i$ dans le cas où les travaux sont ordonnés au plus tard, et $L_i = eft_i - d_i$ dans le cas où les travaux sont séquencés au plus tôt.*

avec eft_i, la date de fin au plus tôt du travail i et lst_i, la date de début au plus tard du travail i.

Nous énonçons et démontrons le théorème suivant :

Théorème 4.1. *La séquence optimale du programme linéaire en nombres entiers PLNE3 minimise le retard algébrique maximal (problème $1|r_i|L_{\max}$) et $L^*_{\max} = -z^*$.*

Démonstration. : Considérons d'abord le cas d'un problème à une seule pyramide et son modèle correspondant PLNE1, et démontrons que la solution optimale du PLNE1 minimise le retard algébrique maximale. Soit alors $\alpha \prec t \prec \beta$ une solution réalisable du modèle PLNE1 et $z = D - R - p_t$, la valeur de la fonction objectif pour cette solution. Concentrons nous sur un travail x affecté à α et calculons son retard algébrique L_x. Dans la situation où tous les travaux sont calés avant leur date échue (au plus tard), le retard algébrique de x s'écrit d'après la Définition 4.1 : $L_x = r_x - lst_x$ avec $lst_x = lst_t - \sum_{i\in\alpha|x\prec i} p_i - p_x$. Nous savons d'après le modèle PLNE1 que $lst_t = D - p_t$ et que $D = \min_{i\in\beta}(d_t, d_i - p_i - \sum_{\{j\in\beta_k|j\prec i\}} p_j))$. D'où, $L_x = r_x + \sum_{i\in\alpha|x\preceq i} p_i - D + p_t$. Il est alors évident que le travail $v \in \alpha$ ayant le retard maximal (*i.e.*, $L_v = \max_{x\in\alpha} L_x$) est le travail qui donne à R sa valeur. On a ainsi $L_v = R - D + p_t$. De la même manière, considérons à présent le cas d'un travail y affecté à β. Si tous les travaux sont calés au plus tôt, le retard algébrique de y s'écrit d'après la Définition 4.1 : $L_y = eft_y - d_y$ avec $eft_y = est_t + p_t + \sum_{i\in\beta|i\prec y} p_i + p_y$. Nous savons que $est_t = R$ et que $R = \max(r_t, \max_{i\in\alpha}(r_i + p_i + \sum_{\{j\in\alpha|i\prec j\}} p_j))$. Ainsi, $L_y = R + p_t + \sum_{i\in\beta|i\preceq y} p_i - d_y$. Le travail $u \in \beta$ qui à le retard maximal (i.e, $L_u = \max_{y\in\beta} L_y$) est le travail qui donne à D sa valeur. Ainsi $L_u = R - D + p_t$. On conclut qu'il existe une stricte équivalence entre la minimisation du retard maximal et la maximisation de la fonction objectif z puisque $L_{max} = -z$.

Considérons à présent le cas général du problème à plusieurs pyramides. On peut déterminer pour chaque pyramide P_k son retard algébrique maximal égal à $L_{P_k} = R_k - D_k + p_{t_k}$ avec R_k (resp. D_k) calculé comme formulé dans 4.15 (resp. 4.16). Sachant que $L_{max} = \max_{k \in 1..m} L_{P_k}$, on a $\min L_{max} = \max(-z)$.

□

Afin d'évaluer la performance du modèle PLNE3, nous l'avons exécuté sur des exemples tests et avons comparé les résultats des expérimentations avec ceux trouvés par l'application de la méthode de Carlier sur les mêmes instances.

Avant de présenter les résultats comparatifs, la section suivante introduit brièvement la procédure arborescente de Carlier.

4.2.1 La méthode de Carlier

Considérant le problème $1|r_i|L_{\max}$, une solution optimale peut être obtenue par application de l'efficace PSEP de Carlier [Carlier 82], fournissant en quelques secondes, une séquence optimale pour des problèmes comptant plus de 10000 travaux. La méthode arborescente de Carlier est conçue pour minimiser le makespan $C_{\max}$ dans un problème à une machine avec dates de début au plus tôt r_j et durées de latence q_j (q_j est un temps incompressible entre la fin de j et la fin de l'ordonnancement). Toutefois, comme indiqué dans [McMahon & Florian 75] et [Levy 96], le problème $1|r_i|L_{\max}$ étant équivalent au problème avec durées de latence en imposant $q_i = \max_{j \in T} d_j - d_i$, la méthode de Carlier peut lui être appliquée.

La méthode de Carlier construit une arborescence dans laquelle chaque nœud est associé à un problème à une machine partiellement séquencé. Le choix des relations de précédence est guidé par l'algorithme de Schrage et exprimé par des actualisations de dates de début au plus tôt et de durées de latence.

L'algorithme de Schrage correspond à une reformulation de l'algorithme EDD (Earliest Due Date) de Jackson (les travaux sont séquencés en ordre croissant des d_j). Il sélectionne à chaque itération le travail prêt de plus grande durée de latence q_j.

La séquence de Schrage associée à chaque nœud est modélisée par un graphe conjonctif dont les sommets correspondent aux travaux, plus deux sommets fictifs source (o) (marquant le début des travaux) et puits $(*)$ (marquant la fin des travaux). Pour chaque travail $i \in T$, l'arc (o, i) est valué par r_i, l'arc $(i, *)$ par $(q_i + p_i)$ et l'arc (i, j), tel que j suit immédiatement i dans la séquence de Schrage, par p_i. Soit $C = [i_1, \ldots, i_p]$ le *chemin critique* du graphe, *i.e.* le plus long chemin de (o) à $(*)$. Deux cas sont à considérer :

- si $\forall i_k \in C \backslash \{i_p\}, q_{i_k} \geq q_{i_p}$, alors le nœud considéré ne peut pas être séparé et la séquence de Schrage est localement optimale ;

- sinon, un *travail critique* J_c et un *ensemble critique de travaux* J sont définis :
  - J_c est le travail de $C \backslash \{i_p\}$ le plus proche du sommet $(*)$ de durée de latence inférieure à q_{i_p} ;
  - J est formé de la partie $[J_{c+1}, \ldots, i_p]$ du chemin C, où J_{c+1} est le travail qui suit J_c sur C.

Il est démontré dans [Carlier 82] que si le sommet considéré contient une solution optimale, elle est telle que, soit J_c est séquencé avant tous les travaux de J, soit tous les travaux de J sont séquencés avant J_c. L'arborescence est donc développée en considérant deux nouveaux problèmes partiellement ordonnancés :

- l'un où J_c est séquencé avant tous les travaux de J, ce qui s'obtient en modifiant la valeur de la durée de latence de façon suivante :

$$q_{J_c} \leftarrow max(q_{J_c}, \sum_{i_k \in J} p_{i_k} + q_{i_p});$$

- l'autre où J_c est séquencé après tous les travaux de J, ce qui s'obtient en modifiant la valeur de la date de début au plus tôt de façon suivante :

$$r_{J_c} \leftarrow max(r_{J_c}, \min_{i_k \in J} r_{i_k} + \sum_{i_k \in J} p_{i_k})$$

Le travail critique et l'ensemble critique des travaux sont par ailleurs utilisés pour déterminer une borne inférieure du makespan $C_{\max}$: l'ensemble T' de l'évaluation $h(T')$ est fixé à J pour le sommet père (avant séparation) et à $J \cup J_c$ pour les sommets fils (après la séparation).

Le détail des algorithmes de Schrage et de Carlier peut être trouvé dans [La 04].

4.2.2 Résultats numériques

Le programme linéaire PLNE3 a été implémenté en utilisant le logiciel ILOG-CPLEX sur une station de travail du LAAS possédant un processeur Intel Xeon 1.60 GHz et 2 Go RAM. Les solutions sont déterminées sur des instances de problèmes comptant 100 à 3100 travaux. Pour chacun de ces cas, 50 instances de problèmes sont considérées. Le modèle PLNE3 et la procédure de Carlier sont exécutés afin de comparer leur temps d'exécution.

Concernant la génération des instances de problèmes, nous avons repris la méthode utilisée dans [Hariri & Potts 83]. Pour chaque travail i, une durée opératoire p_i est générée uniformément dans l'intervalle [1,50]. La date de disponibilité r_i est modélisée par la réalisation d'une variable aléatoire uniforme dans l'intervalle $[0, \alpha \times \sum_{i=1}^{n} p_i]$ où α est un paramètre permettant de régler l'étalement des dates de disponibilité à partir de la date 0. Cinq valeurs de α ont été prises : $\{0,2; 0,4; 0,6; 0,8; 1\}$. La génération des dates de fin au plus tard utilise le même principe en fixant chaque d_i à la réalisation d'une variable aléatoire uniforme dans l'intervalle $[(1-\beta) \times a \times \sum_{i=1}^{n} p_i, a \times \sum_{i=1}^{n} p_i]$. Le paramètre $a \in \{100\%, 110\%\}$ permet de régler la marge temporelle maximale associée aux travaux, alors que le paramètre β contrôle quant à lui la dispersion des dates de fin d_i. Il est choisi tel que $\beta \in \{0,2; 0,4; 0,6; 0,8; 1\}$. Afin d'assurer pour chaque travail i la cohérence des dates de disponibilité et les dates échues avec les durées opératoires, tout d_i dont la valeur serait inférieure à $r_i + p_i$ est modifiée afin qu'elle soit égale à cette valeur.

En appliquant ce principe de génération, nous générons des instances de problème à une machine comptant 100, 400, 700, 1000, 1300, 1600, 1900, 2200, 2500, 2800 et 3100 travaux. Pour chacun de ces cas, 50 instances de problème sont générées. Les tests menés ont permis de déterminer pour chaque problème, le temps CPU nécessaire pour trouver la solution optimale du modèle PLNE3, le temps CPU pour trouver la solution de carlier ainsi que le nombre de nœuds parcourus pour atteindre cette solution. Nous rappellons que la procédure de Carlier utilise l'algorithme de Schrage pour construire à chaque nœud de l'arborescence une séquence de travaux, et qu'il est facile de vérifier si cette solution est localement optimale (cf. Section 4.2.1). Lorsque la première séquence déterminée par l'algorithme de Schrage (i.e. la séquence déterminée par le nœud racine en considérant les données initiales de l'instance) est optimale, on dit que l'instance de problème est Schrage-optimale.

Afin de comparer le modèle que nous proposons avec la méthode de Carlier en terme de temps de calcul, plusieurs indicateurs sont définis comme suit et rapportés dans le Tableau 4.1.

- Le temps de calcul min/moyen/max correspondant à la procédure de Carlier (T_{Car}) ;
- le temps de calcul min/moyen/max correspondant à l'exécution du modèle PLNE3 (T_{PLNE}) ;
- le pourcentage de temps tel que $T_{PLNE} > T_{Car}$ ($\%W$) ;
- le nombre d'instances Schrage-optimale ($\#S$) ;
- le temps de calcul min/moyen/max correspondant à la procédure de Carlier pour les instances difficiles (T_{Car}^{hard}) ;
- le temps de calcul min/moyen/max correspondant à l'exécution du modèle PLNE3 pour les instances difficiles (T_{PLNE}^{hard}) ;

- le pourcentage de temps tel que $T^{hard}_{PLNE} > T^{hard}_{Car}$ ($\%W^{hard}$).

n	Toutes les instances								Les instances difficiles						
	T_{Car}			T_{PLNE}			%W	#S	T^{Hard}_{Car}			T^{Hard}_{PLNE}			W^{Hard}
	min	*moy*	*max*	*min*	*moy*	*max*			*min*	*moy*	*max*	*min*	*moy*	*max*	
100	0	0,02	1	0	0,04	1	4	34	0	0,06	1	0	0,06	1	2,9
400	0	0,3	2	0	0,22	1	16	38	0	1	2	0	0,33	1	8,3
700	0	1,4	6	0	1,24	8	40	35	1	3,66	6	0	1,33	3	0
1000	0	1,22	27	1	3,18	26	90	46	5	14,0	27	1	8,25	26	25
1300	0	2,46	20	2	4,82	12	80	41	4	9,22	20	3	5	9	22,2
1600	0	5,48	29	4	8,4	17	70	33	7	14,3	29	4	8,47	17	17,6
1900	0	3,42	40	4	15,1	44	96	44	11	18,6	40	10	22,5	44	66,7
2200	0	1,06	20	5	16,2	36	100	49	20	20	20	26	26	26	100
2500	0	1,9	31	7	21,4	40	100	47	17	23	31	30	33	37	100
2800	0	0,28	1	8	27,9	54	100	50	---	---	---	---	---	---	---
3100	0	0,4	1	11	35,3	71	100	50	---	---	---	---	---	---	---

TAB. 4.1: Tableau des résultats : L_{max}

Nous distinguons dans le Tableau 4.1 deux parties. La première considère toutes les instances alors que la seconde ne s'intéresse qu'aux instances difficiles.

Dans la première partie, nous remarquons que les situations où l'instance de problème est Schrage-optimale (i.e la procédure de Carlier développe un seul nœud pour retourner la valeur optimale de L_{max}) sont fréquentes dans les expérimentations. En effet, lorsque $n \geq 2200$, nous observons que presque la totalité des instances sont des instances Schrage-optimales (i.e., $\#S \simeq 50$). Ce qui explique que leur temps moyen de calcul est faible relativement aux autres instances qui voient leur temps CPU augmenter avec la taille du problème.

Au vu de ce constat, nous nous sommes proposés de doter CPLEX d'une première solution correspondant à la séquence de Schrage afin de pouvoir diminuer le temps de calcul nécessaire à la résolution du PLNE3. Toutefois, nous soulignons que lorsque l'instance du problème est Schrage-optimale, CPLEX doit prouver l'optimalité de la solution en utilisant les techniques de programmation linéaire alors que la procédure de Carlier déduit rapidement l'optimalité de la solution. Nous observons dans ce cas que le temps CPU moyen T_{PLNE} augmente toujours avec la taille du problème, même dans le cas où $\#S \cong 50$. Cependant, la PLNE retourne de bons résultats pour les petites instances (souvent meilleurs en moyenne que T_{Car} quand $n \leq 700$) et demeure acceptable pour les grosses instances.

Afin d'apporter une analyse plus fine des résultats, nous avons omis de considérer les

instances Schrage-optimales et nous nous sommes focalisés uniquement sur les instances difficiles (voir seconde partie du Tableau 4.1). Dans ce cas, nous constatons que le temps de calcul correspondant à l'approche PLNE est meilleur que celui relatif à la procédure de Carlier, et ce pour les instances ayant un nombre de travaux $n \leq 1600$. Ceci montre que l'approche PLNE est performante surtout lorsque les instances sont difficiles (i.e. le nombre de nœuds explorés par la procédure de Carlier est important), bien que pour les grosses instances, il est difficile de tirer une conclusion vu le nombre faible d'instances difficiles généré.

Notons également, que l'efficacité du modèle PLNE proposé est limitée par la taille de l'instance testée ; malgré que CPLEX réussit à résoudre des instances de plus de 4000 travaux, nous observons que lorsque le nombre de sommets est important, le nombre de variables binaires dans le modèle associé croit considérablement rendant ainsi l'instance difficile à résoudre.

4.3 Minimisation du nombre de travaux en retard

À la Section 4.2, nous avons montré comment adapter le modèle PLNE3 pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité au problème de minimisation du retard algébrique maximal. Dans cette section, et grâce aux résultats de la partie 3.6.2, nous allons tirer le meilleur parti du modèle PLNE3 pour dériver de nouveaux modèles de programmation linéaire en nombres entiers afin d'exprimer le problème de minimisation du nombre de travaux en retard. Plus explicitement, nous montrons comment adapter le modèle PLNE3 pour la détermination d'une borne inférieure et d'une borne supérieure du problème $1|r_i|\sum U_i$.

Nous avons dans la Section 3.6.2, introduit tous les travaux traitant du problème $1|r_i|\sum U_i$. Dans la partie qui suit, nous en rappelons, un peu en détail, les méthodes les plus récentes.

4.3.1 Méthodes de Baptiste et Dauzère-Pérès

Des résultats sur le problème général d'ordonnancement à une machine dans lequel l'objectif est de minimiser le nombre de travaux en retard peuvent être trouvés dans les travaux de Baptiste et al. [P. Baptiste & Peridy 98] [Baptiste *et al.* 03], Dauzère-Pérès et al. [Dauzère-Pérès & Sevaux 04] et [M'Hallah & Bulfin 07]. Les méthodes utilisées sont des méthodes exactes très efficaces et peuvent être appliquées à des problèmes comptant jusqu'à 200 travaux.

Baptiste et al. ont développé des méthodes basées sur une recherche arborescente. Une première méthode utilise la programmation par contraintes pour résoudre le problème à chaque nœud de l'arborescence. La seconde, dont les résultats sont meilleurs comparativement à la première, calcule à chaque nœud de l'arborescence une borne inférieure en

utilisant la méthode de relaxation lagrangienne.

Par la suite, Dauzères-Pérès et Sevaux ont aussi développé une approche de résolution basée sur une recherche arborescente. Un élément clé de l'efficacité de leur méthode est l'utilisation, dans le schéma du branch and bound, d'une condition de dominance relative au critère $\sum U_i$. Celle-ci est exprimée par le théorème suivant.

Théorème 4.2. *[Dauzère-Pérès & Sevaux 04] Il existe nécessairement une séquence optimale au problème* $1|r_i|\sum U_i$ *dans laquelle toute paire de travaux consécutifs* (i, j) *est telle qu'une des conditions (1)* $d_i < d_j$ *ou (2)* $d_i > d_j$ *et,* $\forall k$ *séquencé avant* j, $r_k \leq r_j$ *soit vérifiée.*

Cette condition de dominance a conduit les auteurs à définir une notion de séquence maître nommée σ et qui caractérise l'ensemble S de toutes les séquences satisfaisant le théorème précédent. S contient alors au moins une séquence optimale vis-à-vis de la minimisation du nombre de travaux en retard.

C'est cette notion de séquence maître qui, couplée avec des bornes inférieures de bonne qualité, rend leur méthode arborescente particulièrement performante. La séquence maître est construite selon l'algorithme suivant (les travaux sont supposés avoir été triés par ordre de date de disponibilité croissant).

Proc CREATION-SIGMA-Complexité $O(n^2)$
1. for each job $i \in V$
2. do $\sigma \leftarrow \sigma \cup i$
3. $\overline{J} \leftarrow \overline{J} \cup i$
4. for each job $j \in \overline{J}$ such that $d_j \geq d_i$
5. do $\sigma \leftarrow \sigma \cup j$ (in increasing order of due dates)

FinProc

FIG. 4.3: Algorithme de construction de la séquence maître de Dauzère-Pérès et al.

Á titre illustratif, l'application de l'algorithme de la Figure 4.3 sur l'exemple illustré au chapitre précédent par Figure 3.16, fournit la séquence maître suivante :

$$\begin{aligned}\sigma = \quad & (10,\ 7,\ 10,\ 11,\ 8,\ 11,\ 4,\ 7,\ 18,\ 8,\ 11,\ 12,\ 9,\ 12,\ 5,\ 8,\\ & 11,\ 9,\ 12,\ 3,\ 4,\ 7,\ 10,\ 5,\ 8,\ 11,\ 9,\ 12,\ 1,\ 3,\ 4,\ 7,\ 10,\\ & 5,\ 8,\ 11,\ 9,\ 12,\ 6,\ 8,\ 11,\ 9,\ 12,\ 2,\ 5,\ 6,\ 8,\ 11,\ 9,\ 12)\end{aligned}$$

La longueur de cette séquence est de 50. Elle est à comparer avec la séquence maître-pyramide déduite de l'application du Théorème des pyramides sur le même exemple et illustrée sur Figure 3.17, dont la longueur est 42.

Les auteurs ont utilisé la séquence maître pour guider la méthode arborescente. Elle a permis, en effet, de réduire fortement la combinatoire qui découle généralement des recherches arborescentes. Le schéma de séparation proposé est binaire, i.e., une position de la séquence σ est utilisée ou ne l'est pas. Les nœuds sont évalués par des bornes inférieure et supérieure. La borne inférieure est obtenue en relâchant certaines contraintes du problème et en résolvant de nouveau le problème relâché optimalement par application de la méthode de H. Kise et al. [Kise *et al.* 78]. La borne supérieure est déterminée par application de l'heuristique proposée dans [Dauzère-Pérès 95]. Les résultats expérimentaux ont montré qu'au moins 95% des instances ont été résolues sachant que le temps limite d'arrêt était d'une heure.

De la même façon que pour la condition de dominance de Dauzère-Pérès et al. introduite dans cette section, nous avons montré, dans le chapitre précédent, comment transformer la condition de dominance d'Erschler en une séquence maître-pyramide, et avons défini sous quelles conditions la condition d'Erschler est dominante vis-à-vis de la minimisation du nombre de travaux en retard à travers le Théorème 3.6.

Nous rappelons, que la condition Dauzère-Pérès et al. est dominante vis-à-vis du critère $\sum U_i$, ce qui n'est pas le cas pour la condition d'Erschler. Toutefois, nous allons voir dans ce qui suit, comment exploiter la séquence maître-pyramide pour la modélisation mathématique de certains problèmes particuliers de minimisation du nombre de travaux en retard.

4.3.2 Utilisation de la condition d'Erschler pour le critère ΣU_i

Il est démontré dans le chapitre précédent que la séquence maître-pyramide permettant l'extraction de toutes les séquences dominantes que le Théorème des pyramides caractérise, n'inclue pas forcément toutes les séquences dominantes vis-à-vis du critère $\sum U_i$. Néanmoins, nous avons pu définir sous quelles conditions le Théorème des pyramides peut être dominant vis-à-vis de ce critère. La suite de cette partie est basée sur ces résultats.

Corollaire 4.1. *Étant donné un problème $1|r_i|\sum U_i$, s'il existe une séquence optimale au problème avec tous les travaux sommets ordonnancés à temps, alors la séquence maître-pyramide caractérise toutes les séquences dominantes vis-à-vis du critère $\sum U_i$.*

Démonstration. Évidente d'après le Théorème 3.6 donné au chapitre précédent. En effet, le Théorème des pyramides peut ne pas être dominant vis-à-vis de $\sum U_i$ pour les seuls cas où un sommet du problème n'est pas compris dans la solution optimale (i.e ensemble des travaux en avance). □

Dans ce qui suit, nous donnons deux formulations mathématiques qui permettent d'obtenir respectivement une borne supérieure et une borne inférieure au problème $1|r_i|\sum U_i$.

4.3.3 Modèle PLNE pour l'obtention d'une borne supérieure

4.3.3.1 Utilisation de la séquence maître-pyramide

Sous l'hypothèse qu'il existe une solution optimale au problème $1|r_i|\sum U_i$ avec tous les sommets ordonnancés à temps, nous savons d'après le Corollaire 4.1 que la séquence maître-pyramide caractérise toutes les séquences dominantes de la forme $\alpha_1 \prec t_1 \prec \beta_1 \cdots \prec \alpha_m \prec t_m \prec \beta_m$ vis-à-vis du critère $\sum U_i$. La recherche de cette solution optimale peut être facilement modélisée par un programme linéaire en nombres entiers en adaptant la formulation proposée pour le problème d'admissibilité (cf. Section 4.1.2.2).

Plus explicitement, en considérant le problème de minimisation du nombre de travaux en retard, un travail j non-sommet peut être séquencé dans α_k ou β_k de toute pyramide P_k telle que $u(j) \leq k \leq v(j)$, ou bien *pas séquencé* du tout, i.e. en retard. Ce dernier cas constitue une possibilité non envisageable dans le cas du problème d'admissibilité. Pour intégrer cette possibilité, il suffit de relâcher la contrainte sur les x^+_{ik} et x^-_{ik} par la formule suivante : $\sum_{k=u(i)}^{v(i)}(x^-_{ki} + x^+_{ki}) \leq 1$. De plus, il faut imposer que la séquence trouvée soit admissible. Cela est assuré grâce au Théorème 3.4 de la Section 3.5.2 par le respect de la contrainte suivante : $(D_k - R_k - p_{t_k}) \geq 0 \ \forall k = 1...m$.

Concentrons nous sur la fonction objectif $\sum U_i$. Cette fonction peut être facilement exprimée à l'aide des variables de position x^+_{ik} et x^-_{ik} relatives aux différents travaux, en maximisant le nombre de travaux non sommets réalisés à l'heure, sachant que si un travail est en retard on a $\sum_{k=u(i)}^{v(i)}(x^+_{ik} + x^-_{ik}) = 0$. Ceci se traduira alors par l'expression :

$$\min z = \sum_{\{j \in V \setminus \{t_1 ... t_m\}\}} (1 - \sum_{k=u(j)}^{v(j)} (x^-_{jk} + x^+_{jk}))$$

Nous avons, dans la perspective de formaliser le problème de minimisation du nombre de travaux en retard, supposé qu'il existe une solution optimale qui ordonnancerait tous les sommets à l'heure. Évidemment, une telle hypothèse n'est pas forcement réalisable. Trouver une solution optimale dans une telle forme, i.e. en imposant l'ordre partiel défini par le Théorème des pyramides, permet alors de déterminer une borne supérieure au problème $1|r_i|\sum U_i$. Nous verrons néanmoins dans la partie 4.3.5 consacrée aux expérimentations que cette borne supérieure est de très bonne qualité, voire souvent optimale.

On conclut que le PLNE3 (modèle proposé pour le problème d'admissibilité) peut facilement être adapté pour résoudre le problème de détermination d'une borne supérieure au problème $\sum U_i$. Cette nouvelle formulation est donnée par le modèle (PLNE4) suivant :

$$\min\ z = \sum_{\{j\in V\setminus\{t_1\ldots t_m\}\}}(1-\sum_{k=u(j)}^{v(j)}(x_{jk}^{-}+x_{jk}^{+}))+\sum_{k=1}^{m}\mathbf{y_{t_k}}$$

$$\text{s.t.}\quad\begin{cases}
R_k \geq r_{t_k} & , \ \forall k\in[1\ m] & (5.23)\\
R_k \geq r_i + \sum_{\{j\in P_k|r_j\geq r_i\}} p_j x_{kj}^{+} & , \ \forall i\in V\setminus\{1..m\}, k=u(i) & (5.24)\\
R_k \geq R_{k-1} + \sum_{\{j\in P_{k-1}\}} p_j x_{(k-1)j}^{-} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j\in P_k|r_j\geq r_i\}} p_j x_{kj}^{+} & , \ \forall k\in[2\ m] & (5.25)\\
D_k \leq d_{t_k} & , \ \forall k\in[1\ m] & (5.26)\\
D_k \leq d_i - \sum_{\{j\in P_k|d_j\leq d_i\}} p_j x_{kj}^{-} & , \ \forall i\in V\setminus\{1..m\}, k=v(i) & (5.27)\\
D_k \leq D_{k+1} - \sum_{\{j\in P_{k+1}\}} p_j x_{(k+1)j}^{+} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j\in P_k|d_j\leq d_i\}} p_j x_{kj}^{-} & , \ \forall k\in[1\ (m-1)] & (5.28)\\
\sum_{k=u(i)}^{v(i)}(x_{ki}^{-}+x_{ki}^{+})\leq 1 & , \ \forall k\in[1\ m], \forall i\in P_k & (5.29)\\
D_k - R_k \geq p_{tk}(1-\mathbf{y_{t_k}}) & , \ \forall k\in[1\ m] & (5.30)\\
y_{t_k}\ ,\ x_{ki}^{-}\ ,\ x_{ki}^{+}\in\{0,1\} & , \ \forall k\in[1\ m], \forall i\in P_k & \\
D_k\ ,\ R_k \in \mathbb{Z} & , \ \forall k\in[1\ m] &
\end{cases}$$

Les contraintes (5.23)-(5.28) sont similaires aux premières contraintes (5.16)-(5.21) de la formulation PLNE3 dans la mesure où les variables R_k et D_k demeurent inchangées. Permettre à un travail non-sommet d'être en retard, est assuré par la contrainte (5.29). Nous imposons dans la formulation que la séquence optimale soit une séquence dominante compatible avec la séquence maître-pyramide, avec si possible, tous les sommets ordonnancés à l'heure. Cependant, il existe des cas où trouver une solution avec tous les sommets à l'heure est impossible. En effet, si deux sommets t_k et t_{k+1} consécutifs sont tels que $d_{t_{k+1}} - r_{t_k} < p_{t_{k+1}} + p_{t_k}$, alors il est impossible de trouver une solution réalisable qui séquencerait les deux sommets k et $(k+1)$ à l'heure. Aussi, selon la valeur de la durée opératoire des sommets du problème, il est plus avantageux de préférer à un sommet t_k, un autre travail non-sommet dans le but de maximiser le nombre de travaux en avance. Pour cela, nous associons pour chaque sommet k une variable y_{t_k} égale à 1 si le sommet est ignoré (sa durée opératoire est mise à zéro), à 0 s'il est à l'heure. Les variables de ce type sont prises en compte dans la contrainte (5.30) d'admissibilité de chaque pyramide : Si y_{t_k} est égal à 0 alors la contrainte est similaire à celle du modèle PLNE3 ; si elle prend la valeur 1, cela implique que les travaux de la pyramide P_k sont à l'heure puisque $D_k - R_k > 0$. Le critère d'optimisation $\sum U_i$ est exprimé par les variables y_{t_k}, x_{ki}^{+} et x_{ki}^{-} et permet d'optimiser le nombre de travaux en retards, tous travaux confondus (sommets et non-sommets).

4.3.4 Modèle PLNE pour l'obtention d'une borne inférieure

Dans un problème d'optimisation, une borne inférieure pour le cas d'une minimisation (resp. borne supérieure pour le cas d'une maximisation) est obtenue en relâchant certaines contraintes, puis en résolvant de nouveau le problème optimalement. Pour le problème général $1|r_i|\sum U_i$, certains auteurs se sont intéressés à détecter, selon les valeurs des dates de début au plus tôt et des dates au plus tard, des cas particuliers où la résolution devient polynômiale. Ces cas peuvent être utilisés pour déterminer des bornes inférieures et cela, en relâchant certaines contraintes du problème général.

4.3.4.1 Cas polynômial et bornes inférieures

H. Kise et al. dans [Kise *et al.* 78] ont donné un algorithme optimal pour le cas particulier du problème $1|r_i|\sum U_i$ où les dates de début au plus tôt et les dates de fin au plus tard sont ordonnées de la même manière, autrement dit, ordonnées de façon consistante (en anglais : consistent), i.e. la condition $r_i < r_j \Rightarrow d_i \leq d_j$ est vérifiée pour toute paire de travaux $(i, j) \in V \times V$. Ce problème particulier est en fait le cas d'un ensemble V de n travaux, constitué uniquement de sommets (le problème possède une structure à n pyramides). Les auteurs ont proposé un algorithme de résolution de complexité $O(n^2)$. Pour le même problème, E. L. Lawler [Lawler 82] a développé un algorithme plus simple à comprendre et de complexité moindre $O(n \log n)$.

Ainsi, Dauzère-Pérès et Sévaux dans [Dauzère-Pérès & Sevaux 04] ont utilisé l'algorithme de Kise et al. pour obtenir de nouvelles bornes inférieures pour le problème général $1|r_i|\sum U_i$. Leur résultat est basé sur la proposition suivante :

Proposition 4.1. *[Sevaux 98] La fonction objectif optimale d'un problème de minimisation P dans lequel $r_i < r_j \Rightarrow d_i \leq dj$, $\forall i, j \in V$ est une borne inférieure de la fonction objectif de tout problème P' pour lequel $p_i = p'_i$, $r_i \leq r'_i$ et $d_i \geq d'_i$, $\forall i \in V$ (ou bien $[r'_i, d'_i] \subseteq [r_i, d_i]$, $\forall i \in V$.*

Il est donc possible de diminuer les dates de disponibilité r_i ou bien d'augmenter les dates échues d_i des travaux de n'importe quel problème initial afin de vérifier la condition $r_i < r_j \Rightarrow d_i \leq d_j$, $\forall i, j \in V$.

La solution obtenue en résolvant le problème $1|r_i|\sum U_i$ avec comme vecteur des dates échues (resp. des dates de début) relâchés est une borne inférieure du problème initial. Les expérimentation obtenues avaient révélé que la borne inférieure obtenue par relaxation des dates de disponibilité est meilleure que la seconde borne obtenue en relâchant les dates d'échéances, du fait que les modifications sur les r_i sont moins importantes que les modifications sur les d_i pour les instances considérées.

Dans le même ordre d'idée, nous montrons dans ce qui suit comment adapter le modèle PLNE4 puis l'appliquer à une structure pyramidale relâchée pour l'obtention d'une borne inférieure au problème général $1|r_i|\sum U_i$.

4.3.4.2 Un modèle mathématique pour le cas à pyramides parfaites

On rappelle que :

Définition 4.2. *Un problème à une machine et n travaux est dit à pyramides parfaites, si* $\forall(i,j) \in P_k \times P_k, (r_i \geq r_j) \Longleftrightarrow (d_i \leq d_j)\ \forall k = 1, \ldots, m.$

De plus on démontre le corollaire suivant :

Corollaire 4.2. *Étant donné un problème V à n travaux et à pyramides parfaites, la séquence maître-pyramide* $\sigma_\Delta(V)$ *caractérise toutes les séquences dominantes vis-à-vis du critère* $\sum U_i$.

Démonstration. Les pyramides étant parfaites il est évident, qu'éliminer un sommet d'une pyramide quelconque produira un nouveau problème à pyramides parfaites. En d'autre terme, il n'existe pas de sommet générateur. D'après le Théorème 3.6 donné au chapitre précédent, la condition d'Erschler est dans ce cas dominante vis-à-vis de ΣU_i i.e., que la séquence maître-pyramide caractérise l'ensemble des séquences dominantes vis-à-vis de la minimisation du nombre de travaux en retard. □

Nous déduisons alors d'après le Corollaire 4.2, qu'étant donné la séquence maître-pyramide d'une structure pyramidale parfaite à m sommets $\sigma_\Delta(V) = \alpha_1 \prec t_1 \prec \beta_1 \cdots \prec \alpha_m \prec t_m \prec \beta_m$, la solution optimale vis-à-vis de $\sum U_i$, i.e., ensemble des travaux à temps, est une séquence $\alpha'_1 \prec t'_1 \prec \beta'_1 \cdots \prec \alpha'_m \prec t'_m \prec \beta'_m$ tel que $(\alpha'_k \prec t'_k) \Subset \alpha_k$, $(t'_k \prec \beta'_k) \Subset \beta_k$, et j est nécessairement en retard $\forall j \in \{\alpha_k \backslash (\alpha'_k \prec t'_k)\}$, $\forall j \in \{\beta_k \backslash (t'_k \prec \beta'_k)\}$.

De là, et sachant d'après le Corollaire 4.2 que le Théorème des pyramides est dominant vis-à-vis de la minimisation du nombre de travaux en retard pour un problème $1|r_i|\sum U_i$ à structure pyramidale parfaite, le problème peut ainsi être résolu optimalement par le PLNE5 suivant.

$$\min\ z = \sum_{\{j\in V\setminus\{t_1\ldots t_m\}\}}(1-\sum_{k=u(j)}^{v(j)}(x_{jk}^- + x_{jk}^+)) + \sum_{k=1}^{m} y_{t_k}$$

$$\text{s.t.}\ \begin{cases}
R_k \geq r_{t_k} + \mathbf{y_{t_k}}(\mathbf{r_{n_k}} - \mathbf{r_{t_k}}) & , \forall k \in [1\ m] & (5.31)\\
R_k \geq r_i + \mathbf{(1 - x_{ik}^+)}(\mathbf{r_{n_k}} - \mathbf{r_i}) & & \\
\qquad + \sum_{\{j\in P_k | r_j \geq r_i\}} p_j x_{kj}^+ & , \forall i \in V\setminus\{1..m\}, k = u(i) & (5.32)\\
R_k \geq R_{k-1} + \sum_{\{j\in P_{k-1}\}} p_j x_{(k-1)j}^- + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j\in P_k | r_j \geq r_i\}} p_j x_{kj}^+ & , \forall k \in [2\ m] & (5.33)\\
D_k \leq d_{t_k} + \mathbf{y_{t_k}}(\mathbf{d_{n_k}} - \mathbf{d_{t_k}}) & , \forall k \in [1\ m] & (5.34)\\
D_k \leq d_i + \mathbf{(1 - x_{ik}^-)}(\mathbf{d_{n_k}} - \mathbf{d_i}) & & \\
\qquad - \sum_{\{j\in P_k | d_j \leq d_i\}} p_j x_{kj}^- & , \forall i \in V\setminus\{1..m\}, k = v(i) & (5.35)\\
D_k \leq D_{k+1} - \sum_{\{j\in P_{k+1}\}} p_j x_{(k+1)j}^+ - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j\in P_k | d_j \leq d_i\}} p_j x_{kj}^- & , \forall k \in [1\ (m-1)], \forall i \in P_k & (5.36)\\
\sum_{k=u(i)}^{v(i)}(x_{ki}^- + x_{ki}^+) \leq 1 & , \forall k \in [1\ m], \forall i \in P_k & (5.37)\\
D_k - R_k \geq p_{t_k}(1 - y_{t_k}) & , \forall k \in [1\ m] & (5.38)\\
y_{t_k}\ ,\ x_{ki}^-\ ,\ x_{ki}^+ \in \{0,1\} & , \forall k \in [1\ m], \forall i \in P_k & \\
D_k\ ,\ R_k \in \mathbb{Z} & , \forall k \in [1\ m] &
\end{cases}$$

Avec :

$$\begin{cases} r_{n_k} = \min_{i\in P_k} r_i \\ \\ d_{n_k} = \max_{i\in P_k} d_i \end{cases}$$

La différence entre ce programme linéaire et celui donné par le PLNE4 se situe au niveau des termes rajoutés en gras.

En considérant le problème avec une structure pyramidale parfaite, la solution optimale est une séquence dominante de travaux à l'heure. Cet ensemble de travaux constitue une structure pyramidale parfaite à m sommets. Tout nouveau sommet t'_k est soit le sommet t_k du problème V à n travaux, ou bien un travail $j \in P_k$.

Dans le modèle PLNE5, nous recherchons une séquence à $n' \leq n$ travaux à l'heure, et m sommets. Si un sommet t_k ou un travail j est en retard, alors les contraintes sur R_k

i.e. (5.31) et (5.32) et celles sur D_k i.e., (5.34) et (5.35) du travail correspondant doivent être désactivées. Par exemple, si t_k est en retard, i.e., $y_{t_k} = 1$, alors le terme $y_{t_k}(r_{n_k} - r_{t_k})$ ($y_{t_k}(d_{n_k} - d_{t_k})$ resp.) désactive la contrainte (5.31) ((5.34) resp.) et devient $R_k \geq r_{n_k}$ (resp. $D \leq d_{n_k}$) : contrainte toujours vérifiée (vraie). De la même manière, si un travail i d'une pyramide P_k n'est pas affecté (dans la solution optimale) dans α_k (β_k resp.) alors le terme $(1 - x_{ik}^+)(r_{n_k} - r_i)$ ($(1 - x_{ik}^-)(d_{n_k} - d_i)$ resp.) désactive la contrainte (5.32) ((5.35) resp.). La contrainte s'écrira $R_k \geq r_{n_k} + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_{kj}^+$ (resp. $D_k \leq d_{n_k} + \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_{kj}^-$) et est également toujours vérifiée. Notons que la désactivation des contraintes permet de ne considérer que les travaux qui sont à l'heure.

4.3.4.3 Relaxation des dates de disponibilité

Si pour un problème général, sa structure pyramidale n'est pas parfaite, il est alors possible de diminuer les dates de disponibilité des travaux pour que cela devienne le cas. L'algorithme 6.1 décrit la modification des dates de début au plus tôt. Pour l'exécuter, il est supposé que les travaux sont triés selon l'ordre croissant des dates échues et que j_{n_k} est le travail de P_k qui possède la plus grande date échue.

Proposition 4.2. *La solution obtenue, en résolvant le problème* $1|r_i|\sum U_i$ *par le modèle PLNE5 avec comme dates échues celles obtenues par l'algorithme donné sur la Figure 4.4, est une borne inférieure du problème initial. Cette borne sera appelée* LB_r.

Démonstration. La baisse des valeurs des dates de disponibilité constitue une relaxation des contraintes du modèle initial. La solution du problème relaxé est donc une borne inférieure du problème initial.

□

Algo : Modification des dates de disponibilité. Complexité O(n.m)

Pour toute pyramide P_k, $(k = 1 \ldots m)$ faire
  Pour tout $j \leftarrow 1$ to n_k faire
    Si $r_j > r_{j-1}$ alors $r_j \leftarrow r_{j-1}$
    FinSi
  Fin pour tout
Fin pour tout

FIG. 4.4: Algorithme de relaxation des dates de disponibilité

4.3.4.4 Relaxation des dates échues

De façon similaire à la relaxation des dates de début au plus tôt, il est également possible d'augmenter les dates échues dans la perspective de transformer le problème original en

une structure pyramidale parfaite. L'algorithme donné sur la Figure 4.5 décrit le principe. Son exécution suppose que les travaux sont triés selon l'ordre croissant des dates de début au plus tôt. Le travail de P_k possédant la plus grande date de disponibilité est noté j_{n_k}.

Proposition 4.3. *La solution obtenue en résolvant le problème* $1|r_i|\sum U_i$ *par le modèle PLNE5, avec comme dates échues celles obtenues par l'algorithme illustré sur la Figure 4.5, est une borne inférieure du problème initial. Cette borne sera appelée* LB_d.

Démonstration. Idem que pour la Proposition 4.2, sauf que dans ce cas, ce sont les dates échues qui augmentent. □

Algo : Modification des dates de fin au plus tard

Pour toute pyramide $P_k, (k = 1 \dots m)$ faire
  Pour tout $j \leftarrow 1$ to n_k faire
    Si $d_j > d_{j+1}$ alors $d_j \leftarrow d_{j+1}$
    FinSi
  Fin pour tout
Fin pour tout

FIG. 4.5: Algorithme de relaxation des dates d'échéance

4.3.5 Experimentations et analyse de performance

Les programmes linéaires développés dans ce chapitre ont été implémentés en utilisant le logiciel ILOG CPLEX sur une station de travail Dell precision 690 du LAAS. Pour évaluer la performance de ces modèles, nous avons utilisé les instances de Baptiste et al. [Baptiste *et al.* 03]. Pour des instances de problèmes à une machine comptant 80, 100, 120 et 140 travaux, 120 instances pour chaque cas nous ont été fournis par l'auteur avec les résultats détaillés de leur expérimentation, à savoir, la valeur optimale du critère $\sum U_j$ (OPT) quand elle est atteinte, ou au moins la valeur de la borne supérieure (BEST) [Sadki 08].

Pour chaque instance testée, nous nous sommes intéressés à déterminer :
- une borne inférieure par la résolution du modèle PLNE5 en relaxant les dates de début au plus tôt r_i (LB_r) et le temps CPU de résolution ;
- une borne inférieure par la résolution du modèle PLNE5 en relaxant les dates de fin au plus tard d_i (LB_d) et le temps CPU de résolution ;
- une borne supérieure UB par la résolution du modèle PLNE4 et au temps CPU correspondant.

Fichier de données	LB_r	T_{cpu} (s)	LB_d	T_{cpu} (s)	UB	T_{cpu} (s)	LB_{Bap}	UB_{bap}	T_{cpu} (s)
P80_1	22	0,05	22	0,05	22	0,03	14	22	204,6
P80_2	23	0,06	23	0,05	23	0,04	20	23	56,5
P80_119	38	401	37	0,41	38	0,49	37	38	66,4
P80_120	33	2,2	33	0,16	39	0,14	38	39	100
P100_1	23	0,07	23	0,1	23	0,05	17	23	486,7
P100_2	37	11,05	37	18,8	37	0,14	29	37	327,7
P100_119	36	0,44	38	0,57	40	434,91	39	40	2218,9
P100_1120	45	4,83	45	0,3	46	0,14	45	46	292,4
P120_1	18	0,08	18	0,07	18	0,02	14	18	717,2
P120_2	34	7,02	34	6,76	34	0,06	26	34	564,9
P120_119	44	8,74	44	1,69	44	17,93	44	44	188,4
P120_120	51	8,41	52	118,15	53	0,27	53	53	121,1
P140_1	28	0,12	29	0,15	30	0,1	22	30	2294,5
P140_2	42	3,01	42	3,12	42	0,17	30	42	2164,2
P140_119	57	44,35	58	192,29	59	1,56	58	60	3600
P140_120	61	1,89	61	1,92	62	0,38	61	62	751

TAB. 4.2: Comparaison des résultats (bornes) pour un échantillon d'instances

Nous signalons que lors des expérimentations, nous avons limité le temps de recherche de la solution "optimale" à une heure de calcul. L'optimum est atteint lorsque la borne supérieure UB est égale à une des bornes inférieures LB_r ou LB_d. Nous donnons dans le Tableau 4.2 les résultats obtenus pour un échantillon d'instances. Les résultats de Baptiste et al. sont aussi rapportés dans le tableau ; UB_{Bap} représente la valeur de la solution optimale lorsque le temps CPU est inférieur à 1 heure, LB_{Bap} est la valeur de la borne inférieure. Pour une instance donnée, par exemple la première instance à 80 travaux, la même borne inférieure ($LB_r = LB_d = 22$) est obtenue pour un temps de calcul égal à 0.05 secondes. La borne supérieure, ici égale à la borne inférieure, est obtenue en un temps de calcul égal à 0.03 secondes. Notons aussi que la même solution optimale donnée par LB_{Bap} est obtenue au bout de 204.6 secondes (cf. Tableau 4.2) .

Le Tableau 4.3 synthétise les résultats des PLNE ayant permis la détermination des différentes bornes supérieures et inférieures, à savoir, le pourcentage des instances résolues optimalement en moins d'une heure, ainsi que les temps de calcul minimum, moyen et maximum correspondants. Par exemple, pour les cas des instances à 80 travaux, le solveur renvoie la solution optimale UB dans 98.33% des cas (2 instances parmi les 120 n'ont pas été résolues optimalement), avec un temps de calcul min/moyen/max correspondant à 0.02/ 27.03/ 1757.84 secondes respectivement. Nous remarquons dans la majorité des cas que la solution optimale est atteinte malgré que certaines instances soient difficiles à résoudre. Aussi, quelque soit le nombre de travaux considéré, la borne supérieure renvoie un plus grand pourcentage de solutions résolues optimalement, comparativement à celui donné par la borne inférieure LB_r ou LB_d avec un temps de calcul globalement moins coûteux. Ceci peut facilement s'expliquer par la complexité des modèles ayant permis de

calculer ces bornes. En effet, le modèle PLNE5 définit la borne supérieure en supposant que tous les sommets gardent leur position (selon la séquence maître-pyramide) dans la solution optimale des travaux à l'heure même s'ils sont en retard, alors que le modèle PLNE4 détermine une solution optimale (borne inférieure) en autorisant qu'un travail non sommet puisse devenir le nouveau sommet de la solution optimale, et de ce fait, l'espace de recherche est plus important dans le second cas que le premier cas. Les résultats des expérimentations révèlent aussi que la borne inférieure LB_d obtenue par relaxation des d_i est souvent plus efficace que LB_r obtenue par relaxation des r_i. Ceci est certainement lié à la structure d'intervalles de chaque instance, et est dû au fait que les modifications sur les dates de disponibilité r_i sont plus importantes que les modifications sur les dates échues d_i.

Instances	LB_r Instances résolues/ CPU	LB_d Instances résolues/ CPU	UB Instances résolues/ CPU
n=80	94.16 % (0.01 ; 42.35 ; 1395.54)s	96.66 % (0.02 ;61.15 ; 2363.76)s	98.33 % (0.02 ; 27.03 ; 1757.48)s
n=100	82.50 % (0.02 ; 141.15 ; 3531.11)s	81.66 % (0.03 ; 85.70 ; 1318,86)s	94.13 % (0.02 ; 33.84 ; 1778.42)s
n=120	80.83 % (0.02 ; 106.78 ; 1340.29)s	84.16 % (0.04 ; 108.43 ; 2149.83)s	85 % (0.02 ; 127.67 ; 2600.95)s
n=140	65.83 % (0.05 ; 139.77 ; 1490.64)s	65.00 % (0.03 ; 173.97 ; 3072.82)s	73.33 % (0.02 ; 134.62 ; 2600.95)s

TAB. 4.3: Pourcentage et temps CPU des instances résolues optimalement

Dans le Tableau 4.4, nous retrouvons le pourcentage des solutions optimalement atteintes, i.e., les cas où la borne supérieure est égale à la meilleure borne inférieure ($UB = LB = \max\{LB_r, LB_d\}$. Nous distinguons deux cas, selon que l'on considère la totalité des instances testées ou bien seulement celles qui ont été résolues optimalement en moins d'une heure. Comme on peut le remarquer, l'optimalité a été atteinte dans plusieurs cas même si on demeure globalement moins bon que les résultats de Baptiste et al. (cf. Tableau 4.5). Pour les instances de 80 travaux par exemple, 73.21% des solutions trouvées en moins d'une heure sont optimales. La 3ème colonne du Tableau 4.4 illustre les écarts entre la borne supérieure et la meilleure borne inférieure trouvée LB à travers les valeurs $min, moyen, max$. Par exemple, pour les 120 instances de 100 travaux, la différence entre la borne supérieure et la meilleure borne inférieure trouvée est de 0 travaux au minimum (ce sont les cas où $LB = UB$)i, de 2 travaux au maximum, et de 0.39 travaux en moyenne. Nous constatons que la dispersion des valeurs est faible même pour les grandes instances.

Enfin, le Tableau 4.5 permet une analyse de la performance de notre borne supérieure. Nous nous comparons aux résultats de Baptiste et al. du fait que nous disposons en détail de

Instances	$UB = LB$ avec $LB = max(LB_r, LB_d)$ Toutes les instances	Instances avec $CPU < 1h$	$\delta = UB - LB$ $(min, moyen, max)$
n=80	70.83 %	73.21%	(0 ; 0.38 ; 6)
n=100	70%	73.62 %	(0 ; 0.39 ; 2)
n= 120	56.66%	60.68 %	(0 ; 0.5 ; 2)
n=140	60.83 %	62.31 %	(0 ; 1.45 ; 2)

TAB. 4.4: Pourcentage des solutions optimales

Instances	Baptiste et al.		Dauzère-Pérès et al.		$UB = OPT$ $T_{CPU} < 1h$		$UB \leq BEST$
n=80	96.70 %	117.3 s	98.3 %	49.0 s	95.83 %	27.03 s	100 %
n=100	90.00 %	273.5 s	95.0 %	78.4 s	90.00 %	33.84 s	100 %
n=120	84.20 %	538.2 s	93.3 %	89.70 s	81.66 %	127.64 s	99.17%
n=140	72.50 %	1037.3 s	73.3 %	233 s	71.66 %	134.62 s	98.33 %

TAB. 4.5: Analyse de performance de la borne supérieure

leurs expérimentation (OPT et $Best$ pour chaque instance). La borne supérieure est alors comparée à OPT dans le cas où elle est optimale , ou bien à la meilleure solution retournée $BEST$ dans le cas contraire. Ainsi dans ce tableau, figurent sur la seconde colonne, les résultats de Baptiste et al. (pourcentages d'instance résolues optimalement ainsi que les temps moyens de calcul correspondants). Nous rapportons toutefois les résultats Dauzère-Pérès et al. dans la 3ème colonne, qui sont relativement meilleurs à ceux de Baptiste et al. essentiellement en terme de temps de calcul.

Nous pouvons alors observer que, pour presque toutes les instances, notre borne supérieure est égale à la solution optimale ou à la meilleure solution trouvée par la méthode de Baptiste et al. et pour un le temps de calcul beaucoup plus réduit. Ce constat nous conduit a proposer, pour augmenter le pourcentage des solutions certifiées optimales, d'améliorer la borne inférieure. La possibilité de relâcher à la fois les dates de disponibilité ainsi que les dates échues avec comme objectif de minimiser les modifications sur ces dates pourrait être explorée.

4.4 Discussion

Nous avons, dans ce chapitre, proposé de nouvelles formulations de programmation linéaire en nombres entiers pour modéliser le problème à une machine.

À la base de la condition de dominance d'Erschler, et des conditions numériques que nous avons énoncées dans le chapitre précédent, et considérant le problème d'admissibilité, un modèle de programmation linéaire (PLNE3) pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité est proposé. Nous montrons que l'admissibilité de la

solution ainsi trouvée dépend de la valeur de la fonction objectif.

Nous avons ensuite démontré l'équivalence du modèle PLNE3 au cas de résolution du problème de minimisation du plus grand retard algébrique $1|r_i|L_{max}$. En effet, la recherche de la solution la plus dominante est démontrée équivalente à la solution qui minimise le L_{max}. Les expérimentations sur le modèle, comparées en terme de temps de calcul à celles retournées par la procédure de Carlier, ont montré que notre approche est relativement efficace pour des instances difficiles à résoudre (particulièrement quand elles ne sont pas Schrage-optimales), malgré que la procédure de Carlier demeure plus performante dans la majorité des cas.

Nous nous sommes aussi intéressés au problème de minimisation du nombre de travaux en retard $1|r_i|\sum U_i$ en s'appuyant sur les résultats énoncés dans la Section 3.6.2. Partant du modèle de programmation linéaire établi pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité (PLNE3), et sous l'hypothèse que tous les sommets peuvent être séquencés à l'heure rendant ainsi la condition analytique de dominance vraie vis-a-vis de $\sum U_i$, nous avons d'une part montré comment adapter le modèle PLNE3 pour la détermination d'une solution optimale (en proposant le modèle PLNE4) ; cette solution constitue une borne supérieure au problème général $1|r_i|\sum U_i$. D'autre part, le Théorème des pyramides étant démontré dominant vis-à-vis du problème de minimisation du nombre de travaux en retard lorsque la structure pyramidale associée au problème est parfaite, un modèle de programmation linéaire (PLNE5) est proposé dans ce cas pour la détermination de la solution optimale ; notons qu'en relâchant les dates de disponibilité ou bien les dates échues des différents travaux de sorte à rendre le problème original parfait, la résolution du problème ainsi relâché (le modèle PLNE5) constitue alors une borne inférieure au problème général $1|r_i|\sum U_i$.

Les expérimentions réalisées sur les instances de Baptiste et al. dont nous disposions (avec les résultats sur les valeurs des bornes supérieure et inférieure pour chaque instance) avaient pour but d'évaluer la performance de nos modèles en terme de la qualité des bornes obtenues et du temps de calcul, et ce, comparativement aux résultats donnés par l'auteur. Nous avons alors constaté que notre borne supérieure est optimale ou proche de l'optimum dans la majorité des cas, avec un temps de calcul plus réduit.

Signalons toutefois que pour le même problème, d'autres travaux de recherche sont venus améliorer la procédure de Baptiste et al. À commencer par les travaux de Dauzère-Pérés et al. [Dauzère-Pérès & Sevaux 04] dont les résultats sont rapportés dans le Tableau 4.5 ; puis, les travaux de M'Hallah et al. [M'Hallah & Bulfin 07] proposant un algorithme de résolution basé sur la procédure du Branch and bound en utilisant à la fois, une heuristique pour le choix de la solution initiale, et un modèle mathématique du problème du sac à dos à choix multiple pour le calcul des bornes. Cette méthode, la plus récente à notre connaissance et publiée durant la préparation de ce travail, a permis de résoudre toutes les

instances difficiles non résolues auparavant, avec un temps de calcul très réduit.

Enfin, notons que malgré que nous nous comparons à des méthodes ad-hoc pour l'évaluation de nos modèles en terme de performance et temps de calcul, et que nos résultats se révèlent moins performants que les derniers résultats publiés dans la littérature, principalement pour le problème $1|r_i|\sum U_i$, les modèles de programmation linéaire en nombres entiers que nous proposons sont originaux, compact et relativement efficace. De plus, nous avons utilisé une méthode générique. En effet, le développement de modèles PLNE pour la résolution des problèmes d'ordonnancement à une machine est toujours intéressant dans la mesure où il est possible de les adapter pour prendre en compte de nouvelles contraintes ou de nouveau critère de performance (ce qui est plus difficile à faire avec des méthodes ad-hoc).

Troisième partie

Le cas multi-ressource : une méthode coopérative

Introduction

Le travail considéré dans notre étude est dédié à l'ordonnancement. Nous nous sommes intéressés dans la partie précédente à un problème d'optimisation combinatoire pour le cas d'une ressource. Exploitant un théorème de dominance démontré dans les années quatre-vingt, nous avons proposé de résoudre le problème de manière exacte en utilisant des techniques de programmation mathématique.

Dans cette dernière partie du manuscrit, nous nous intéressons au problème d'ordonnancement à plusieurs ressources dans un contexte perturbé.

L'environnement d'application des ordonnancements étant par essence incertain, la probabilité qu'un ordonnancement soit réalisé comme prévu est faible. De ce fait, pour réduire l'écart entre la solution choisie (théorie) et la solution réellement mise en œuvre (pratique), une meilleure prise en compte des incertitudes et des aléas est nécessaire. Ce constat a poussé, ces dernières années, de nombreux chercheurs à développer des méthodes d'ordonnancement sous incertitudes. D'un autre coté, les ressources exécutantes sont souvent réparties au sein d'un ensemble d'acteurs, se trouvant parfois en situation de concurrence, et disposant de fait d'une autonomie de décision, favorable à la réactivité, dont il est fondamental de tenir compte. Il est ainsi possible de tirer profit d'une organisation distribuée de la fonction d'ordonnancement. Ces raisons nous ont orientées pour proposer, dans ce travail, une nouvelle approche d'ordonnancement avec prise en compte des perturbations. C'est une approche qui permet à la fois d'anticiper et de réagir en temps réel aux aléas et, qui intègre une dimension distribuée dans la résolution. Elle s'inscrit dans le cadre d'une approche robuste (apport de flexibilité séquentielle par exploitation de la condition de dominance précitée), et fait appel à la coopération entre les différents acteurs constituants le système de fabrication (distribution des décisions) pour l'ordonnancement global du système.

Dans le Chapitre 5, nous présentons un aperçu des approches traitant de l'ordonnancement sous incertitudes. Ensuite, nous mettons l'accent sur le fait que la plupart des approches (hors-ligne ou en-ligne) supposent qu'une entité décisionnelle gère l'organisation de la totalité des ressources et possède une connaissance globale des paramètres du système, pour ensuite argumenter l'intérêt de distribuer les décisions afin d'accorder plus d'autonomie aux différents acteurs du système et, d'adopter une démarche coopérative.

Enfin, un petit tour d'horizon des approches coopératives est proposé.

Le Chapitre 6 décrit une approche d'ordonnancement robuste à une machine existante [La 04]. C'est une approche qui permet d'anticiper les perturbations en caractérisant hors-ligne un ensemble de solutions flexibles, exploitables en-ligne. La flexibilité est séquentielle, i.e. l'ordre des opérations sur la ressource est différent d'une solution à une autre, et est produite grâce à l'ordre partiel défini par la condition de dominance (théorème des pyramides). La performance de l'ensemble flexible est connue et son évaluation (les performances au mieux et au pire de l'ensemble caractérisé) peut se faire en temps polynomial. Nous verrons que plus un ensemble de solutions est flexible (cardinalité de l'ensemble est importante) et moins sa performance au pire est bonne et vice-versa.

Dans le Chapitre 7, nous formulons la problématique d'ordonnancement multi-ressource (atelier de type job shop) sous incertitudes. L'idée de base de l'approche de résolution proposée consiste à considérer que chaque ressource, assimilée à un centre de décision, possède sa propre autonomie décisionnelle et ses propore objectifs (ordonnancement local robuste à une machine), et que la fonction ordonnancement est répartie entre les centres constituants l'atelier. Nous décrivons les mécanismes de coopération adoptés, et définissons les intervalles de fin des opérations (dates de livraison), correspondant au dates de début (dates de consommation) des opérations successeurs, comme objets de coopération. L'ordonnancement global est construit de façon distribuée et réactive. Afin d'aboutir à terme à une solution satisfaisante, sachant que les décisions d'ordonnancement évoluent dans le temps et sont le résultat des négociations entre les acteurs, nous introduisons des notions de cohérence, de flexibilité, et de risque d'incohérence [Briand *et al.* 08] [Briand & Ourari 08].

Enfin, le Chapitre 8 décrit l'approche adoptée pour la résolution du problème d'ordonnancement sous forme d'une heuristique. La solution d'ordonnancement est construite de façon distribuée et dynamique par coopération asynchrone entre les acteurs. Les fonctions inhérentes à la coopération, à savoir la négociation, la coordination et le renégociation, sont définies. La coopération est réalisée par échanges de messages. Le comportement de chaque centre est décrit par un algorithme permettant la construction des décisions d'ordonnancement. Ces décisions constituant les réponses aux sollicitations des autres centres sont établies de manière réactive et au fur et à mesure de l'évolution du système de production. Elles sont aussi le résultats de l'application des modèles mathématiques que nous proposons et qui se distinguent selon que la réponse est adressée à l'amont ou à l'aval.

Chapitre 5

Revue des approches robustes et coopératives

Ce chapitre fait tout d'abord un tour d'horizon des approches pour l'ordonnancement sous incertitudes. Ces approches sont classées en général dans la littérature en trois catégories : les approches réactives, les approches proactives et les approches proactives-réactives. Elles se distinguent selon la façon de prendre en compte les incertitudes, soit hors-ligne ou en-ligne. Ensuite, nous mettons l'accent sur le fait que la plupart des approches hors-ligne ou en-ligne supposent qu'une entité décisionnelle gère l'organisation de la totalité des ressources et suppose une connaissance globale des paramètres du système. Nous argumentons ensuite l'intérêt de distribuer les décisions afin d'accorder plus d'autonomie aux différents acteurs du système en adoptant une démarche coopérative. Enfin, un tour d'horizon des approches coopératives est donné.

5.1 Tour d'horizon des méthodes

Une classification de référence des méthodes d'ordonnancement avec prise en compte des incertitudes sur les données et les perturbations qui surviennent en temps réel dans l'atelier est donnée dans [Davenport & Beck 00], [Leus 03] et [Herroelen & Leus 04b], les plus citées dans la littérature. Selon les méthodes proposées, on discerne en général trois approches d'ordonnancement. Elles se distinguent essentiellement par les phases (hors-ligne et en-ligne) au cours desquelles sont anticipées les perturbations (cf. Figure 5.1). L'approche prédictive ne tient pas compte de la présence des perturbations et calcule l'ordonnancement en se basant sur des données estimées. En pratique, l'ordonnancement prédictif devient rapidement non faisable et retourne de faibles performances. Les approches proactives calculent par anticipation un ordonnancement en tenant compte a priori des connaissances sur les incertitudes probables. Une autre approche en-ligne, l'approche réactive, calcule l'ordonnancement en temps réel en prenant en compte tout type d'incertitudes pouvant surgir. Enfin, il est possible de coupler, comme le montre la Figure 5.1, deux approches

en-ligne et hors-ligne afin de tirer profit des avantages qu'offre chacune d'elle.

Dans la suite de cette section, nous donnons le principe de chacune de ces différentes approches, en décrivant brièvement quelques travaux connus dans la littérature.

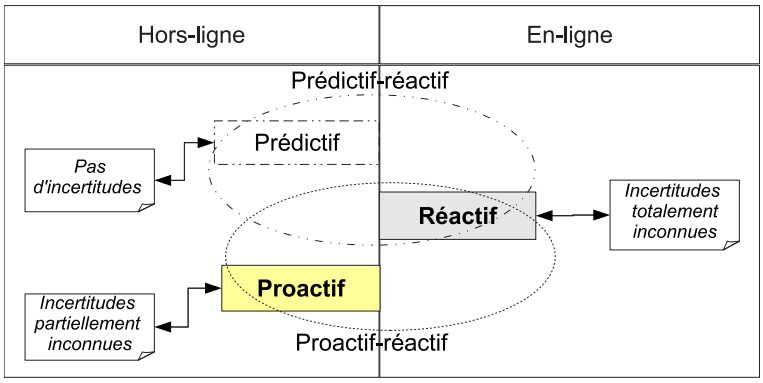

FIG. 5.1: Positionnement des différentes approches d'ordonnancement sous incertitudes

5.1.1 Approches réactives et prédictives-réactives

Les approches réactives prennent les décisions de séquencement et d'affectation au moment même de l'occurrence de l'événement aléatoire en se basant sur les données mises à jour et éventuellement sur l'ordonnancement prédictif initial (baseline schedule en anglais). On peut ainsi distinguer deux types d'approches d'ordonnancement réactif :

Les approches purement réactives : Ces approches qualifiées aussi de "dynamiques", souvent utilisées dans des environnements fortement perturbés, considèrent le cas où aucun programme prévisionnel de fabrication n'est donné à l'avance à l'atelier. Les décisions sont prises dynamiquement et en temps réel au fur et à mesure de l'apparition des aléas, en privilégiant la rapidité des décisions sur leur qualité. Une des méthodes les plus utilisées pour le pilotage réactif est la gestion des files d'attentes par règles de priorité (règle SPT, EDD, LPT, etc.) [Pinedo 08]. Ces approches déterminent, à chaque fois qu'une ressource se libère ou bien à chaque intervalle de temps durant l'exécution, le prochain travail ou l'ensemble de travaux à exécuter, en utilisant une règle spécifique de priorité, ou une combinaison de règles [Blackstone *et al.* 82], [Pierrevall & Mebarki 97], [Holhaus & Rajendran 00], [Ourari *et al.* 01] et [Ourari & Bouzouia 03] . On note que l'efficacité d'une règle dépend de la mesure de performance associée aux critères d'évaluation, qui sont en pratique souvent multiples [Ramasesh 90]. Le choix de la règle n'est donc pas forcement statique, mais peut être dynamique afin de permettre au système de s'adapter aux situations nouvelles par une modification dynamique de la règle de priorité, et d'améliorer ainsi considérablement la performance du système [Pierrevall & Mebarki 97], [Priore *et al.* 98] et [Priore *et al.* 01].

Les approches prédictives-réactives : La démarche dans ce cas consiste, en se basant sur un ordonnancement prédictif généré initialement, à déterminer à chaque occurrence d'événement, ou bien à des intervalles réguliers dans le temps, un nouvel ordonnancement afin de prendre en considération l'état nouveau du système [Sabuncuoglu & Goren 00]. Si on considère le cas du ré-ordonnancement continu qui consiste à calculer un ordonnancement à chaque fois qu'un nouvel événement surgit, on garantit la qualité de la solution au cours du temps. Toutefois, le temps de calcul peut être prohibitif, et peut par conséquent retarder la production. De plus, la solution est très instable dans le temps, ce qui est peu souhaitable en pratique. Le cas du ré-ordonnancement périodique appliqué à des intervalles réguliers requiert moins de calcul que le ré-ordonnancement continu, seulement, il peut donner de moins bonnes performances, surtout si une ou plusieurs perturbations importantes surgissent entre deux instants de ré-ordonnancement. Une technique alternative d'ordonnancement prédictif-réactif consiste simplement, et en se basant sur l'ordonnancement initial, à utiliser des règles simple de réparation.

En optant pour un ré-ordonnancement continu ou périodique, le nouvel ordonnancement généré peut être considérablement différent du précédent, et peut ainsi remettre en question des activités de planification de l'atelier, comme l'affectation du personnel, la distribution des matières premières, etc. Il découle de la génération fréquente et répétée d'ordonnancement une instabilité de l'atelier : phénomène connu sous le nom de "nervosité" dans la littérature [Davenport & Beck 00]. Ainsi, plusieurs méthodes de ré-ordonnancement sont proposées adoptant une stratégie de perturbation minimale des ordonnancements générés (modifiant à chaque fois et le moins possible l'ordonnancement initial). Parmi ces méthodes, nous relevons :

- Les méthodes basées sur des actions de réparation : un exemple de règle de réparation est la méthode de décalage à droite (en anglais, right shift rule). Cette règle, utilisée dans le cas d'une panne ressource, consiste à retarder dans le temps tous les travaux affectés par la panne (les opérations affectées à la ressource en panne et celles affectées par les contraintes de précédences) [Sadeh *et al.* 93]. Il est clair que cette méthode, bien qu'elle préserve la stabilité de l'atelier, peut donner de mauvaises performances, puisqu'elle ne permet pas le ré-ordonnancement des travaux.

- Le ré-ordonnancement avec un minimum de perturbation : C'est une méthode qui consiste à trouver un compromis entre l'efficacité de l'ordonnancement généré et la stabilité de l'atelier. Dans [Sakkout *et al.* 98], les auteurs définissent un problème de perturbation minimal comme étant la recherche d'un nouveau ordonnancement de manière à respecter d'une part les contraintes du problème, et d'autre part, à ce que la somme des différences entre les dates de début des opérations de l'ordonnancement prédictif et de l'ordonnancement généré (ré-ordonnancé) soit minimale. D'autres auteurs dans [Wu *et al.* 93] ont étudié le problème à une machine avec ar-

rivée dynamique de tâches et ont proposé une heuristique qui concilie la qualité de la solution (minimiser le makespan) et sa stabilité (séquence et dates de fin). Il ont montré que leur méthode permet de construire un ordonnancement plus stable au prix d'une faible réduction de la performance relativement à une optimisation globale (algorithme de Carlier [Carlier 82]). Une autre technique de ré-ordonnancement stable qui mérite d'être citée, est l'approche par retour vers l'ordonnancement initial (en anglais, match-up rescheduling). Elle consiste à ré-ordonnancer une partie de l'ordonnancement initial et à revenir en un certain point dans le temps sur l'ordonnancement initial.

Une autre approche d'ordonnancement prédictive-réactive intéressante est proposée dans [Chauvet 99]. Contrairement aux approches sus-citées, l'idée dans ce cas consiste à ordonnancer les produits dès que leur commande est connue, sans attendre les informations sur les demandes à long terme, et sans remettre en question les produits déjà ordonnancés correspondant à des engagements auprès des clients. L'auteur a développé des algorithmes rapides d'ordonnancement permettant de fournir des résultats avant même que d'autres événements ne viennent perturber l'état du système, et ce en utilisant au mieux les périodes de disponibilité des ressources. Plus précisément, sa méthode propose au moment même où une commande arrive un délai minimum de livraison précis et réalisable négociable avec le client, et fabrique le produit au plus tard tout en respectant son engagement afin de minimiser le temps d'occupation des ressources et s'offrir plus de liberté pour ordonnancer les produits futurs.

5.1.2 Approches proactives

Nous avons vu que pour faire face aux aléas lors de la résolution de problèmes d'ordonnancement, les approches réactives tiennent compte des incertitudes durant la phase dynamique du processus de résolution. Une autre orientation, visant à prendre en compte des connaissances sur les perturbations durant la phase de génération de la solution statique initiale, a donné naissance aux approches proactives. Plusieurs méthodes proactives pour l'ordonnancement en présence d'aléas sont proposées dans la littérature [Davenport *et al.* 01] [Herroelen & Leus 04b] [Sabuncuoglu & Bayiz 09]. Elle se distinguent généralement par le type d'aléas pris en considération, le degré de flexibilité fourni, et le niveau de performance à garantir (acceptabilité ou optimalité). On trouve alors des approches basées sur certaines formes de redondances pour tenir compte des incertitudes, des approches stochastiques qui prennent en compte l'incertitude lors du calcul de l'ordonnancement de référence, les approches utilisant une mesure de robustesse, et les approches générant un ensemble d'ordonnancements flexibles. Dans la suite nous résumons quelques unes d'entre elles.

Approches basées sur les redondances : Certaines techniques proactives tiennent compte des incertitudes en introduisant des formes de redondances (flexibilité temporelle) dans

l'ordonnancement prédictif de référence, en prolongeant par exemple certaines durées opératoires ou en augmentant les disponibilités des ressources cela dans le but d'absorber les aléas.

Parmi ces techniques, les approches basées sur l'utilisation de marges de protection temporelle "Temporal protection", en anglais, sont proposées dans [Chiang & Fox 90] et [Gao *et al.* 95]. Pour tenir compte des éventuelles perturbations de type panne machine, les auteurs proposent une approche robuste basée sur la prolongation des durées des opérations. La durée d'une tâche j affectée à une ressource R est alors calculée en ajoutant à sa durée opératoire originale la durée moyenne de panne pouvant se produire durant l'exécution de j par R. Les résultats des expérimentations réalisées pour un problème de type job shop ont montré que l'utilisation de la marge de protection temporelle est pertinente. Cette marge permet en effet de réduire la déviation, en terme de makespan, entre l'ordonnancement prédictif et celui réalisé. Toutefois, on reproche à la technique de protection temporelle d'autoriser la perte des marges supplémentaires accordées aux opérations en cas de pannes. Pour palier cet inconvenient et permettre une exploitation maximale des ressources, Davenport et al. introduisent dans [Davenport *et al.* 01] deux techniques : Time Windows Slack et Focused Time Window Slack. TWS et FTWS incorporent des contraintes supplémentaires au problème d'ordonnancement afin de minimiser les marges temporelles accordées aux opérations.

Leon et al. dans [Leon *et al.* 94] proposent une *approche basée sur les marges*. Dans leurs travaux, les auteurs décrivent une méthode robuste d'ordonnancement et redéfinissent la fonction d'évaluation en incorporant une mesure de robustesse. Le problème ainsi considéré est un problème job shop soumis à des perturbations liées à l'indisponibilité des machines (pannes). Lors de l'occurrence d'une perturbation, la règle de décalage à droite est utilisée comme règle de ré-ordonnancement. Le retard de l'ordonnancement S est alors défini comme une variable aléatoire qui exprime la différence entre l'ordonnancement réel et l'ordonnancement prévu : $\delta(S) = M(S) - M_0(S)$, $M(S)$ étant la variable aléatoire désignant la durée totale de S en présence de perturbations, et $M_0(S)$, représente la durée totale de l'ordonnancement S calculée a priori sans présence de perturbation. Puisque $M_0(S)$ est déterministe, il est déduit que $E[\delta(S)] = E[M(S)] - M_0(S)$ i.e., une valeur faible de l'espérance mathématique du retard implique que l'ordonnancement est faiblement affecté par les perturbations. Une espérance nulle peut être alors réalisée en insérant des temps morts importants dans l'ordonnancement. Les auteurs proposent une mesure de robustesse $R(S)$ correspondant à une combinaison linéaire de l'espérance mathématique de la durée totale et de l'espérance mathématique du retard, soit, $R(S) = r.E[M(S)] + (1-r).E[\delta(S)]$ avec $r \in [0, 1]$. Il est démontré que $R(S)$ peut être calculée de façon exacte dans le cas d'une seule perturbation, mais qu'elle est très difficile à calculer en cas de plusieurs perturbations. Dans ce dernier cas, les auteurs proposent une autre expression pour approcher au mieux la valeur de l'espérance mathématique $E[\delta(S)]$ donnée par l'équation suivante :

$RD_3 = \sum_{i \in N_f} slack_i(S)/|N_f|$ où N_f désigne l'ensemble des opérations exécutées sur les machines pouvant tomber en panne et $Slack_i(S)$, la marge libre de l'opération i représentant la durée dont i peut être retardée sans augmenter la durée totale de l'ordonnancement S (tout en maintenant l'ordre d'exécution des opérations sur la machine). Un algorithme génétique pour minimiser la mesure de robustesse est ensuite développé. Les résultats des expérimentations ont montré que pour des valeurs de r proches de 1, les performances des ordonnancements qui minimisent la mesure de robustesse sont meilleures que celles qui minimisent le makespan.

Herrolelen et Leus dans [Herroelen & Leus 02] se sont intéressés au problème d'ordonnancement de type projet lorsque les durées opératoires assignées aux activités du projet sont sujettes à incertitudes. Leur apport se situe dans le développement de modèles de programmation mathématique pour la génération d'un ordonnancement stable capable d'absorber les perturbations sur les durées sans affecter le programme des autres activités. Ils utilisent le concept de marge libre $F_{ij}(S) = s_i(S) - f_i(S)$ définie comme étant la différence entre la date de début de j et la date de fin de i dans l'ordonnancement S, i et j étant 2 activités reliées par un arc dans le graphe disjonctif relatif au problème. Les auteurs suggèrent de faire commencer les activités quelque part entre leur date de début au plus tôt et leur date de début au plus tard. Ils proposent une mesure de robustesse qui minimise la somme pondérée des déviations des dates de début des activités entre le programme réellement exécuté et le programme prédéfini, quand une ou plusieurs durées d'activités sont perturbées. La mesure de robustesse permet dans ce cas d'assurer une stabilité sur les dates de début des activités, et l'expression qu'ils minimisent est : $\sum_{j=1}^{n} c_j(E[s_j] - s_j(S))$ où c_j est le coût de surcharge sur la date de début de l'activité i, s_j, étant la variable aléatoire représentant la date de début réelle de j après exécution du projet, et $s_j(S)$ désigne la date de début de j dans l'ordonnancement proactif. L'expression de l'espérance mathématique $E(s_j)$ est donnée en fonction de la marge libre définie plus haut, de la date de début de j, de la probabilité de perturbation p_j assignée à l'activité j et de la longueur de perturbation L_j d'une activité j quand elle est perturbée. Les résultats de simulation rapportés montrent que leur méthode offre de meilleurs résultats relativement à d'autres méthodes testées.

L'idée d'optimiser une mesure de robustesse est aussi utilisée dans les travaux de Sevaux et Sörensen [Sevaux & Sörensen 04]. Les auteurs ont étudié le problème à une ressource avec dates de disponibilité des travaux et avec comme objectif, la minimisation de la somme pondérée des retards. Leur but est de calculer un ordonnancement robuste dont la performance est insensible aux variations des dates de disponibilité des tâches. Les auteurs proposent une méthode de résolution basée sur les algorithmes génétiques et définissent la fonction d'évaluation robuste suivante : $f_r(x) = (1/m)\sum_{i=1}^{m} w_i f(x + \delta_i)$, où $x + \delta_i$ représente la solution dérivée, w_i, le poids associé à la solution dérivée et m, le nombre

d'instances générées (fixé dans leur cas à 10). Cette fonction ajoute donc du bruit i.e. une variable aléatoire δ_i, à la solution courante x avant évaluation pour un certain nombre d'instances. Ce bruit représente en fait une modification aléatoire des dates de disponibilité. Les auteurs montrent que l'approche est flexible et est facile à implémenter, d'où l'intérêt de prendre en considération la nature non déterministe de certaines données lors du calcul de l'ordonnancement.

5.1.3 Approches proactives-réactives

Comme nous l'avons mentionné dans les sections précédentes, une approche d'ordonnancement proactive exploite une connaissance a priori des perturbations lors de la construction de l'ordonnancement prédictif initial. Toutefois, il est irréaliste de vouloir tenir compte proactivement de tous les événements aléatoires pouvant surgir au niveau de l'atelier. De ce fait, un ordonnancement capable de gérer les incertitudes doit intégrer, non seulement, une approche proactive, mais aussi une approche réactive afin de réagir aux événements incertains non pris en compte dans la phase proactive.

Certaines approches proactives-réactives s'intéressent non pas à fournir un ordonnancement de référence unique avec une flexibilité temporelle sur les dates de début ou de fin d'exécution, mais également, à déterminer un ensemble flexible d'ordonnancements de référence, offrant ainsi plus de choix durant la phase dynamique de résolution du problème ([Aloulou 02], [Esswein 03], [La 04], [Herroelen & Leus 04a], [Policella 05], etc). La flexibilité dans ce cas n'est pas seulement temporelle, mais séquentielle, et son introduction permet d'augmenter la robustesse de l'ordonnancement.

Parmi ces approches, plusieurs sont basées sur l'utilisation du concept de groupes d'opérations permutables ou l'utilisation de structures d'intervalles.

Le concept de groupe d'opérations permutables, né au LAAS-CNRS dans l'approche proactive-réactive baptisée ORABAID, (ORdonnancement d'Ateliers Basé sur une AIde à la Décision) a fait l'objet de plusieurs travaux [Demmou 77], [Thomas 80], [Billaut 93], [Artigues 97] et [Esswein *et al.* 05]. La méthode ORABAID se compose de deux parties : proactive et réactive. La partie proactive vise à incorporer de la flexibilité sequentielle lors de l'ordonnancement prévisionnel. L'idée de base est de déterminer statiquement un ensemble d'ordonnancements admissibles ayant une performance au pire mesurable en temps polynomial. La partie réactive est chargée de l'exploitation de l'ensemble des ordonnancements admissibles pour le pilotage en temps réel de manière à préserver au mieux la flexibilité pour le futur. On note que l'ensemble de solutions admissibles est constitué d'un ensemble de séquences de groupes d'opérations totalement permutables pour chaque ressource. La caractérisation de cet ensemble pose un problème de complexité [Artigues 97], et la méthode proposée pour la définition d'une séquence de groupe d'opérations repose sur une heuristique ayant comme objectif de satisfaire un compromis entre deux objectifs :

la construction de groupes contenant un nombre important d'opérations (flexibilité), et la recherche de l'admissibilité, i.e. la tenue des délais des travaux (cohérence). Si l'heuristique ne parvient pas à obtenir une séquence admissible, une procédure d'amélioration itérative est engagée, en deuxième étape, pour se rapprocher de l'admissibilité. À l'issue de cette deuxième étape, si la séquence est admissible, elle constitue un plan de fabrication flexible pour le pilotage de l'exécution des opérations. Dans le cas contraire, une méthode permettant de négocier avec les clients des délais des travaux est proposée. Enfin, une phase de regroupement cherche à augmenter le nombre d'opérations de chaque groupe tout en conservant l'admissibilité de la séquence, le but étant d'augmenter la flexibilité de la séquence de groupe. C'est cette séquence qui est transmise au niveau de l'atelier et qui servira de base pour la conduite temps réel de l'atelier.

Dans [Aloulou 02], [Aloulou *et al.* 02] et [Aloulou *et al.* 07], les auteurs s'inspirent du *concept de groupe d'opérations permutables* pour proposer une nouvelle approche d'ordonnancement proactive-réactive. La principale différence avec la méthode précédente est liée à la structure de représentation d'une solution et aux mesures de qualité de cette solution. Un ordre partiel entre les opérations à exécuter est défini, et les groupes d'opérations deviennent ainsi partiellement permutables. L'intérêt d'utiliser un ordre partiel entre les opérations, au lieu de la séquence de groupe totalement permutable, réside dans la possibilité de mieux maîtriser la performance au pire, tout en offrant un bon niveau de flexibilité. En effet, ajouter des ordres partiels sur les opérations d'un groupe peut minimiser le nombre de groupes, et par conséquent, permet d'obtenir des solutions plus flexibles, cela pour la même garantie de performance. L'approche proposée est constituée de deux phases : Une phase en-ligne (algorithme proactif) et une phase hors-ligne (algorithme réactif). L'algorithme proactif est un algorithme génétique qui propose au décideur une sélection d'ordre partiel parmi tous ceux possibles tout en garantissant un bon compromis entre les indicateurs de performance et de flexibilité. Les auteurs suggèrent de restreindre l'ensemble des ordonnancements considérés à ceux souvent utilisés dans la pratique, à savoir, les ordonnancements actifs, semi-actifs et sans-délai. Enfin, l'algorithme réactif guide, en fonction de la politique retenue et des perturbations qui surviennent, pour le choix de la tâche à réaliser parmi celles appartenant aux groupes en cours d'exécution, cela tout en respectant l'ordre partiel imposé.

Une autre approche utilisant le concept de groupe de tâches permutables est proposée dans [Esswein 03] et [Esswein *et al.* 05]. Cette nouvelle approche aborde le problème de construction d'un ordonnancement de groupe de façon multicritère. Partant du constat qu'aucune interaction relative à la recherche d'un compromis entre qualité et flexibilité n'est proposée, les auteurs proposent une nouvelle manière d'aborder la phase proactive, qui consiste à intégrer à la fois la notion de qualité et de flexibilité au sein d'une méthode proactive multicritère. Ainsi, une mesure de flexibilité pour l'ordonnancement de

groupes liée au nombre total de groupes (flexibilité séquentielle) est proposée, et la qualité de l'ordonnancement de groupes est définie par une performance au mieux et au pire. Sachant que plus la flexibilité est grande, plus le nombre d'ordonnancements caractérisé sera grand, et pire sera la qualité du plus mauvais d'entre eux, l'approche bi-critère proposée vise à maximiser la flexibilité de l'ordonnancement de groupe tout en garantissant une qualité donnée du pire ordonnancement caractérisé. De plus, l'approche a l'avantage de garantir également la qualité dans le meilleur des cas (perte de qualité acceptable). La méthode proactive proposée a été mise en œuvre pour la résolution de plusieurs types de problèmes : flow shop, job shop et open shop à deux machines. Des résultats expérimentaux sont présentés attestant de l'efficacité de l'approche, en effet, des solutions très flexibles sont construites en un temps raisonnable sans dégrader la qualité dans le pire des cas.

Dans le même ordre d'idée, Policella dans [Policella 05] a considéré le problème d'ordonnancement de projet sous contraintes de ressources avec des relations de précédence généralisées, problème désigné par l'acronyme RCPSP/max. Le but étant de proposer des ordonnancements robustes, l'auteur a explicitement considéré la question de comment tirer avantage de la flexibilité temporelle pour faire face à la dynamique d'exécution. La solution proposée pour accroître la robustesse de l'ordonnancement consiste à définir un ordre partiel d'ordonnancement (en anglais, Partial Order Schedule, ou *POS*). Un *POS* est un ensemble flexible de solutions représenté par un graphe temporel. Les activités assignées à la même ressource sont alors partiellement ordonnées de manière à garantir le respect des contraintes de ressources en cas de perturbations de type temporelle (e.g. variations des durées opératoires ou des dates de disponibilité), ou de type aléas (e.g. arrivée de nouveaux travaux ou non disponibilité de ressource). Pour générer un *POS*, les auteurs proposent deux méthodes de contraintes (Constraint Satisfaction Problem, en anglais). La première approche est une méthode de réparation itérative qui résout les conflits détectés sur les ressources en évaluant l'ensemble des solutions possibles par le calcul de bornes sur l'utilisation des ressources. Ces bornes sont obtenues en utilisant un concept décrit dans [Muscettola 02]. La seconde approche, plus intuitive, est dénommée "Solve and Robustify". Elle part d'une solution non flexible calculée au départ, puis, un *POS* est ensuite déduit par application d'une heuristique de chaînage appelée en anglais "chaining" [Cesta *et al.* 98]. Les deux méthodes sont validées sur plusieurs benchmarks et les résultats des expérimentations ont montré que la seconde méthode produit des ordonnancements avec de meilleures propriétés de robustesse. On note aussi que l'approche "Solve and Robustify" résout un nombre plus important d'instances, donne de meilleures performances, et fournit les solutions en un temps plus réduit.

Les approches proactives-réactives utilisant le concept de groupe d'opérations permutables citées jusqu'ici ont plusieurs avantages. En plus du fait qu'elles n'utilisent pas de modèles d'incertitudes, ces méthodes permettent de caractériser un ensemble de solutions

très vaste dont la performance au pire est mesurable en temps polynomial. La qualité d'une solution étant mesurée à l'aide d'indicateurs de performance et de flexibilité. Toutefois, le concept de groupe de tâches présente quelques limites [La 04]. D'une part, la robustesse créée à l'issue de l'insertion de la flexibilité séquentielle est potentielle, dans la mesure où il n'est pas garanti qu'elle soit suffisante en regard d'un ensemble de scénarios de perturbations. D'autre part, en utilisant le concept de groupe d'opérations permutables, seules les tâches contiguës sur une ressource peuvent être permutées, ce qui limite quelque peu la flexibilité séquentielle pouvant potentiellement être générée. Pour palier ces problèmes, et donc exprimer davantage de flexibilité séquentielle, des travaux récents proposés au LAAS sont basés sur l'utilisation de structures d'intervalles pour la construction d'ordre partiel [Briand *et al.* 05]. Dans leur travaux, les auteurs proposent des méthodes d'ordonnancement robustes basées sur des approches utilisant des conditions nécessaires, suffisantes ou dominantes d'admissibilité ou d'optimalité. Ces conditions permettent de définir des ordres sur un sous-ensemble de tâches du problème (qualifiés de partiel) tout en bornant la qualité au pire de l'ensemble flexible caractérisé et en mesurant sa flexibilité. De telles approches sont intéressantes du fait qu'elles sont relativement insensibles aux variations des données des problèmes considérés. De plus, ces approches sont interactives et permettent de rechercher un compromis satisfaisant entre flexibilité et performance.

Plus explicitement, les auteurs ont abordé le problème à une machine et le problème du flow-shop de permutation à deux machines. Dans le cas du problème $1|r_i|L_{max}$, les auteurs dans [Briand & La 03] et [Briand *et al.* 03b] utilisent un théorème de dominance établi par Erschler et al. dans [Erschler *et al.* 83]. Leur méthode permet de caractériser un ensemble de solutions sans les énumérer, par utilisation d'un ordre partiel construit par application du théorème de dominance. Cet ordre offre potentiellement d'avantage de flexibilité séquentielle que l'ordre partiel relatif au concept de groupe de tâches. De plus, la qualité de l'ensemble de solutions peut être évaluée en temps polynomial par sa flexibilité (cardinalité de l'ensemble flexible de solutions) ainsi que par une performance au pire et au mieux. Une procédure par séparation et évaluation progressive est développée pour énumérer un large ensemble de solutions optimales du problème $1|r_j|L_{max}$ contenue dans l'ensemble dominant initial, par la définition d'une structure d'intervalles (ordre partiel) optimale. Les expérimentations réalisées sur des problèmes comptant jusqu'à 500 travaux ont prouvé l'efficacité de la procédure qui fournit des ensembles extrêmement larges de solutions, en un temps relativement court. Il est montré que ces ensembles sont robustes vis-à-vis d'un ensemble de valeurs r_j et d_j que l'on peut caractériser par des intervalles, cela quelles que soient les durées opératoires.

S'intéressant au problème flow shop de permutation à deux machines $F2|prmu|C_{max}$, Briand et al. dans [Briand *et al.* 06a] définissent un ordre partiel suffisant vis-à-vis du critère C_{max}. Deux structures d'intervalles définies par les durées opératoires sont considérées et une condition suffisante d'optimalité caractérisant un ensemble de séquences optimales est donnée. Une propriété intéressante de l'ordre partiel suffisant est qu'il est relativement

insensible aux variations des durées opératoires dans la mesure où les relations d'ordre entres les durées restent conservées. De plus, l'ordre ainsi défini est une extension de l'ordre partiel établi par la règle de Johnson [Johnson 54]. Les expérimentations réalisées sur des problèmes comptant jusqu'à 500 travaux ont montré que le nombre de solutions optimales trouvées augmente exponentiellement en fonction du nombre de travaux, et ce, en un temps de calcul très raisonnable.

5.2 Pour une approche d'ordonnancement robuste basée sur la coopération

Globalement, la résolution en-ligne ou hors-ligne des problèmes d'ordonnancement avec prise en compte des incertitudes est classiquement assimilée à un problème de décision globale car la fonction ordonnancement gère l'organisation de la totalité des ressources et suppose une connaissance globale des paramètres du système, chaque ressource devant respecter la solution établie pour assurer la cohérence globale des décisions. Pourtant, dans de nombreux champs d'applications (chaînes logistiques, projets industriels,...) les ressources exécutantes sont souvent réparties au sein d'un ensemble d'acteurs, se trouvant parfois en situation de concurrence, et disposant de fait d'une autonomie de décision, favorable à la réactivité, dont il est fondamental de tenir compte. De plus, l'environnement dans lequel évoluent les différents acteurs est incertain, ce qui impose d'adopter une organisation capable de résister et de réagir rapidement aux perturbations et aléas internes au système. Ce type d'organisation distribuée constitue un champ favorable à l'application de l'ordonnancement robuste ; chaque acteur peut être considéré comme une entité capable de résoudre des problèmes, et peut ainsi faire face non seulement aux perturbations émanant de sa propre organisation interne, mais aussi et surtout à celles qui peuvent provenir de son environnement externe constitué des autres acteurs.

Bien qu'une organisation centralisée permet d'assurer une cohérence globale des décisions, elle exige en contrepartie, une transparence totale des différents acteurs, et n'autorise pas de remise en cause des décisions prises globalement. Connaissant les limites de l'approche centralisée dans certains champs applicatifs, nous préconisons d'adopter une méthode d'ordonnancement robuste basée sur une coopération entre les différents acteurs. La coopération a pour but d'amener les acteurs à converger vers un compromis satisfaisant les exigences de performance globale et celles de performance locale.

Les progrès technologiques énormes observés dans le domaine des sciences et technologies de l'information et de la communication ont considérablement bouleversé les modes d'organisation des entreprises. Le secteur industriel connaît une émergence des plateformes de coopération, et plusieurs travaux de recherche se sont intéressés au développement de systèmes d'information coopératifs et de systèmes d'aide à la décision permettant aux ac-

teurs de mieux échanger les informations dans un souci de meilleure coordination et de réduction des coûts. Toutefois, peu de ces récents travaux abordent la problématique d'ordonnancement coopératif.

Dans le cadre des travaux menés dans cette partie de thèse, et s'appuyant sur les arguments précédemment évoqués dans cette section, nous nous sommes intéressés à définir un axe de recherche original alliant l'ordonnancement robuste et la coopération. Pour cela, considérant la problématique de gestion d'un atelier de production de type job-shop, nous proposons de considérer l'ordonnancement comme une fonction distribuée où la solution globale résulte d'une coopération entre les différentes ressources. Ressources assimilées à des centres de décision dans la suite du manuscrit, chaque centre de décision gérant son propre ordonnancement local de façon robuste, i.e. utilisant une méthode proactive pour créer de la flexibilité séquentielle exploitable en-ligne lors de l'ordonnancement réactif.

Dans les prochains chapitres, nous présentons une nouvelle approche coopérative et robuste pour l'ordonnancement sous incertitudes. Le reste de ce chapitre est consacré à un tour d'horizon sur la coopération en général et la coopération en ordonnancement.

5.3 Tour d'horizon de l'ordonnancement coopératif

5.3.1 Définitions et état de l'art

La notion de coopération est un concept très large qui intéresse différentes disciplines (économie, sociologie du travail ou technique de l'engénierie). Plusieurs définitions sont données dans la littérature. Barnard dans [Barnard 38] définit la *"coopération comme un moyen qui permet de dépasser les limites de l'action individuelle"*. On mentionne dans [Campagne & Sénéchal 02] que *la coopération contribue à l'amélioration de la performance globale et à la minimisation des risques ou des conflits*. Pour Hatchuel [Hatchuel 96], *la coopération est la raison d'être des organisations*. En effet, vue la nature conflictuelle et fluctuante de l'environnement industriel, la coopération permet à la fois l'augmentation de l'autonomie et le renforcement de la dépendance entre les différentes entités, permettant de définir des actions collectives pour une recherche croissante de performance globale, et ce, à travers le développement d'une organisation distribuée et le développement de politiques de coopération.

Dans le domaine des systèmes manufacturiers, cadre auquel nous nous intéressons, la coopération met en jeu des processus d'interaction entre différents acteurs qui représentent soit les entreprises, les ateliers ou bien des pools de ressources. Les définitions suivantes y afférant sont données. Erschler dans [Erschler 96] définit *la coopération comme étant le support de la mise en œuvre de la décision entre plusieurs centres de décisions*. Pour Bou-

jut et al. [Boujut *et al.* 02], *la coopération consiste à l'élaboration et à la prise de décisions collectives.*

Dans le cadre de la gestion des chaînes logistiques, domaine clé de la compétitivité des entreprises, des travaux récents de recherche ont montré l'intérêt de la mise en œuvre de l'organisation coopérative. Le but est de permettre aux différents acteurs constituant la chaîne logistique de préserver leur autonomie et de réagir rapidement aux perturbations sans remettre systématiquement en cause les décisions prises avec les partenaires, et en préservant la cohérence des décisions individuelles, tout en améliorant aussi bien leurs performances locales que globales. L'objectif de la coopération inter-entreprise est alors de concevoir et de produire mieux et plus vite, des produits et services innovants [Sardas *et al.* 02]. Plusieurs travaux de recherche se sont penchés sur la problématique de la coopération inter-entreprise, et sur le développement d'outils de coopération [Monteiro 01], [Telle 03], [Despontin-Monsarrat 04], [Albino *et al.* 05], [Portmann & Mouloua 07] et [Lin *et al.* 08].

L'ordonnancement occupe une place importante pour l'organisation des systèmes manufacturiers. Comme défini au Chapitre 1, son rôle est de planifier les activités sur les ressources de manière à assurer la satisfaction des clients et l'utilisation maximale des ressources, en prenant en compte les différents objectifs et les différentes contraintes physiques (internes et externes) et temporelles (objectifs et contraintes pouvant être potentiellement contradictoires). De ce fait, le processus d'ordonnancement est davantage assimilable à une méthode de résolution basée sur une recherche interactive entre les acteurs qu'à une méthode d'optimisation mono ou multi-critère [Archimede & Coudert 01]. Les problèmes d'ordonnancement étant fort complexes, quelques travaux récents de recherche proposent, comme dans le cas des chaînes logistiques, des méthodes de résolutions basées sur une coopération entre les acteurs (ressources). Nous y reviendrons un peu plus en détail dans la Section 5.3.3.

Notons que plusieurs travaux utilisant les techniques des Systèmes Multi-Agents (SMA), domaine de recherche lié à l'intelligence artificielle distribuée, ont été développés pour les systèmes d'ordonnancement coopératif [Pendharkar 07]. Dans [Archimede & Coudert 01], l'auteur a développé un modèle SMA pour l'ordonnancement flexible des systèmes de production. Son modèle introduit une coopération indirecte entre les agents OFs (ordres de fabrication) représentant les clients et les agents machines représentant les fournisseurs. La coopération est assurée de manière synchrone à travers un blackboard et est contrôlée par un agent superviseur. Les agents s'échangent des propositions/ contre-propositions, leur objectif étant de négocier des intervalles d'exécution des travaux.

Pour résoudre un problème d'ordonnancement multi-projets a contraintes de ressources décentralisées, Homberger dans [Homberger 07] développe un SMA composé de plusieurs agents ordonnanceurs, chacun gérant les activités d'un projet parmi tous les projets concur-

rents. Afin allouer les ressources aux agents, un processus de négociation itératif est amorcé entre les agents ordonnanceurs et sous le contrôle d'un agent médiateur. Le processus est réitéré tant que la fonction objectif global (minimisation du makespan moyen) est améliorée.

Citons aussi les travaux de [Tranvouez 01] sur la problématique de prise en compte des perturbations pouvant surgir au niveau d'un atelier de production tout en préservant au mieux les ordonnancements préétablis. L'auteur propose une approche de réordonnancement distribué et coopératif, intégée dans un système multi-agents. Un formalisme de représentation de comportement, consistant en un graphe d'états, est défini pour spécifier par des plans comportementaux les activités des agents en coopération.

Une autre vision de la coopération peut être trouvée dans les problèmes de satisfaction de contraintes, CSP en anglais pour Constraint Satisfaction Problem. Un CSP consiste à trouver une assignation consistante de valeurs à des variables prenant leurs valeurs dans des domaines discrets et finis. Un sous domaine des CSPs est les CSPs distribués : DiCSP ; une discipline de recherche émergente [Bessiere *et al.* 01]. Dans ce cas, les variables et les contraintes sont distribuées à travers des agents autonomes et ne peuvent être centralisés. Une fonction de répartition est ainsi définie pour la répartition des variables entre les agents. Chaque agent connaît alors les contraintes liées à ses variables. Citons comme exemple les travaux de Eisenberg [Eisenberg 03]. L'auteur a examiné un modèle organisationnel de coopération et de coordination pour la gestion distribuée des ressources d'un projet réel, et a formalisé le problème de gestion distribuée des ressources du projet de collaboration sous la forme d'un problème à satisfaction de contraintes distribuées. Dans le modèle DiCSP considéré, chaque tâche est représentée par une variable qui dénote sa date de début. Les contraintes qui lui sont liées sont de 3 types : contraintes temporelles, les contraintes de précédence et les contrainte de capacité de ressource.

De manière générale, les algorithmes standards de résolution des DiCSP, comme les algorithmes asynchrones de recherche en arrière, supposent un ordre de priorité entre les agents pour l'envoi des messages. Les agents les plus prioritaires font part de leurs décisions aux agents de plus faible priorité et dont ils sont liés. D'autres modèles plus réalistes, permettant aux agents de garder privées leurs décisions, n'autorisent qu'une connaissance partielle des contraintes d'un agent aux autres agents [Brito *et al.* 09].

Soulignons aussi que le plus souvent dans le cadre des DiCSP, un agent est associé à chaque variable, c'est le cas du problème de génération d'un emploi du temps d'employés [Moyaux *et al.* 03].

5.3.2 Les formes de coopération

La coopération s'appuie sur les interdépendances mutuelles des différents acteurs afin de mettre en place des processus d'interaction et d'aboutir à une solution harmonieuse. Cette solution dépend de la forme de coopération adoptée. En effet, plusieurs aspects de la coopération peuvent être identifiés : à savoir la coordination et la collaboration [Monteiro 01], [Despontin-Monsarrat 04].

La *coordination* est définie dans [Thomassen & Lorenzen 00] comme *une mise en cohérence des actions des acteurs qui entreprennent différentes activités en réduisant au minimum les coûts de division du travail.* Elle vise d'une part l'organisation et la synchronisation des actions dans le temps, et gère d'autre part la cohérence des actions individuelles par rapport au fonctionnement du groupe à travers des règles et procédures.

La *collaboration* signifie participer avec d'autres à l'exécution d'une action pour produire un résultat final. Dans [Dillenbourg *et al.* 96], les auteurs énoncent que la *collaboration s'appuie sur un engagement mutuel des participants dans un effort de coordination afin de réaliser les activités.* Ainsi, la collaboration n'entraîne pas de prises de décisions collectives. Son expression est utilisée à la place du terme coopération lorsque les actions individuelles ne sont pas différentiables.

Lorsque des décisions sont prises en collaboration entres différents acteurs, on parle de *codécision.* La codécision est donc une collaboration relative au processus de décision. C'est aussi le résultat de mécanismes de *négociation* ou *renégociation* entre acteurs. Lorsque deux partenaires collaborent pour la première fois pour une prise de décision, on parle alors de négociation. Si au contraire, la collaboration a pour objet la remise en cause d'une décision déjà négociée, on parlera de renégociation [Camalot 00].

5.3.3 Organisation de la coopération en centres de décision

5.3.3.1 Organisation de la coopération dans un système distribué

Les premiers systèmes d'organisation des décisions étaient centralisés, dans le sens où le problème est abordé d'un seul point de vue global, c'est à dire qu'il existe une entité (centre de décision superviseur) qui supervise toutes les activités des acteurs et qui prend des décisions globales. Connaissant les limites de ce type d'organisation (cf. Section 5.2), des travaux récents de recherche se sont penchés sur la distribution des décisions permettant ainsi d'aborder le problème selon plusieurs points de vue partiels (décomposition en sous-problèmes). L'approche distribuée permet donc de répartir l'autonomie de décision de façon plus homogène sur chaque centre de décision, et autorise chaque centre de remettre éventuellement en cause des décisions prises en amont. Toutefois, l'inconvénient de cette approche est qu'elle requiert des méthodes spécifiques de gestion pour une cohérence globale des différents choix.

Selon si l'organisation des décisions est centralisée ou distribuée, la coopération peut avoir différents sens. Dans le cadre d'une organisation centralisée, et en reprenant les expressions définies dans [Sardas *et al.* 02], la coopération est verticale et reflète ainsi la hiérarchie entre les centres de décisions. On peut remarquer que la coopération s'apparente plus dans ce cas à une action de coordination qu'à une réelle prise de décision collective. Comme déjà rapporté dans [Tranvouez 01], le terme coopération ne sera pas employé pour décrire des interactions entre centres de décision de niveau hiérarchique différents, le terme coordination lui sera préféré.

Dans le cadre d'une organisation distribuée, la coopération est à la fois verticale et horizontale. En effet, en plus de la coopération entre les hiérarchies décisionnelles, une coopération horizontale concerne la réalisation distribuée des fonctions d'un même niveau, cf. figure 5.2 a). Toujours dans ce même cadre, on trouve aussi les architectures autonomes qui considèrent uniquement une coopération verticale entre les centres de décisions. Ce sont des architectures qui s'adaptent bien pour le développement des systèmes de production distribué avec un nombre réduit de centres de décisions [Shen 02].

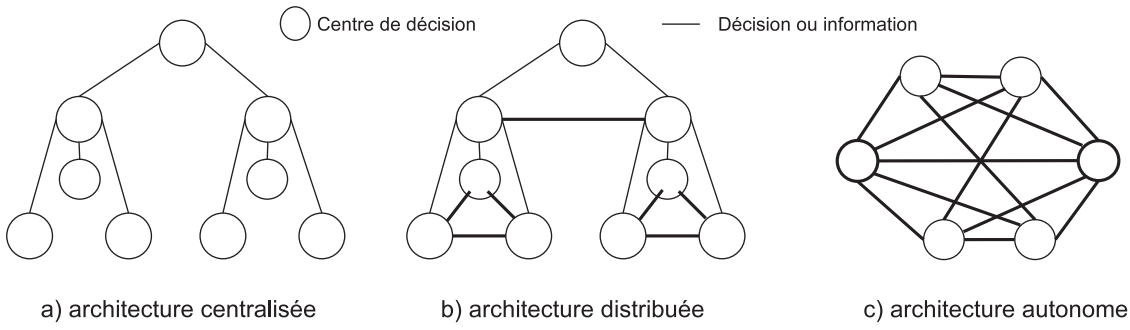

FIG. 5.2: Organisations centralisée et distribuée

Nos travaux de recherche touchent aux systèmes d'ordonnancement distribués. La fonction ordonnancement est ainsi répartie sur plusieurs acteurs autonomes ayant leurs propres objectifs et coopérant ensemble pour aboutir à une solution globale satisfaisante. Selon la littérature scientifique, il existe plusieurs modes de distribution. Ils se distinguent selon le cadre conceptuel d'identification du problème global en niveaux d'abstraction, des unités de résolution allouées au sous problèmes, de l'organisation sociale de ces sous problèmes ainsi que de la nature d'interaction qui régit les acteurs pour la construction d'une solution d'ordonnancement. Dans ce qui suit, et en nous appuyant principalement sur les travaux de [Tranvouez 01], nous citons trois modes de distribution de l'ordonnancement :

- *Ordonnancement par l'aide à la décision* : L'approche ordonnancement par l'aide à la décision autorise l'intégration de l'utilisateur humain dans le processus de décision. Elle permet aux décideurs d'orienter le processus de résolution pour une ra-

tionalisation des prises des décisions. La charge de résolution est alors distribuée entre les acteurs logiciels (centres de décision) et les acteurs humains (cf. Figure 5.3.a). Placé à la périphérie du système décisionnel, l'homme coopère avec le système informatique pour évaluer les conséquences de toutes les options envisagées pour prendre sa décision. Plusieurs travaux de recherche se sont intéressées à développer des modèles intégrant au mieux l'utilisateur dans le processus décisionnel, nous citons [Lopez *et al.* 96], [Camalot *et al.* 97] et [Urbani 06].

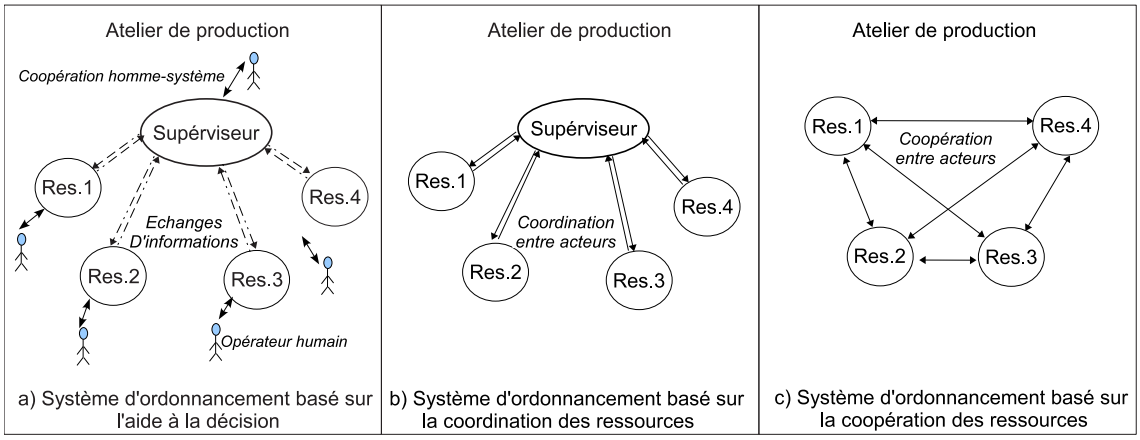

FIG. 5.3: Différents modes de distribution en ordonnancement

- *Ordonnancement par coordination des ressources* : Un modèle générique d'ordonnancement propose dans ce cas une approche distribuée des ordonnancements en décomposant le problème d'ordonnancement en plusieurs sous-problèmes : chaque centre de décision prenant en charge un problème partiel. Les sous-problèmes étant par essence interdépendants, la recherche de la cohérence des décisions prises localement et le maintien d'une performance globale sont réalisés, par des mécanismes de coordination entre les centre de décision et une entité appelée superviseur (cf. Figure 5.3.b). Ainsi, lorsqu'une perturbation surgit au niveau de la partie physique d'un centre de décision, ce dernier interagit avec le superviseur pour l'informer. Le superviseur négocie alors avec tous les centres dont les décisions dépendent de celles prises par le centre perturbé, pour obtenir une nouvelle solution d'ordonnancement. Dans certains travaux de recherche, il est toutefois autorisé à des centres de décisions de communiquer sans passer par le superviseur pour échanger des informations et coordonner leurs actions, i.e. chaque centre établit des décisions locales en prenant en compte les actions entreprises par les autres centres de décision [Mebarki *et al.* 96], [Archimede & Coudert 01]. Ce type de coordination est appelé coordination par ajustements mutuels.

- *Ordonnancement par coopération des ressources* : Contrairement à l'ordonnancement

par coordination des ressources qui n'autorise qu'une coopération verticale pour la prise de décision, et limite ainsi l'autonomie des centres de décision, l'ordonnancement par coopération entre les ressources s'appuie sur la capacité de réactivité et de coopération des centres pour obtenir une solution d'ordonnancement (cf. Figure 5.3.c). La gestion hospitalière est un exemple de domaine d'application d'ordonnancement par coopération des ressources. Le personnel (ou un service) d'un hôpital est considéré, du fait de son haut niveau d'autonomie, comme une ressource ou un centre de décision disposant de son propre programme et d'objectifs propres. Les ressources doivent alors coopérer pour définir une solution globale performante pour une organisation globale efficace tout en respectant les objectifs locaux [Haspeslagh & Causmaecker 07], [Kaplansky & Meisels 07].

5.3.3.2 Notion de réseau de centres de décision

Dans le cadre de l'ordonnancement distribué, nous considérons que la prise de décisions d'ordonnancement est répartie entre les ressources autonomes, ces dernières étant assimilées à des centres de décision.

Un centre de décision est défini dans [Erschler *et al.* 96] comme étant une fonction de décision caractérisée par un ensemble de variables de décision, représentant l'ensemble des actions possibles du centre sur un certain horizon. La prise de décision correspond alors à l'instanciation d'une ou plusieurs variables de décision.

Dans [Artigues *et al.* 02], les auteurs proposent une vision générique et fonctionnelle d'un centre de décision. Sur la base d'un modèle, et à partir de paramètres internes ou externe et de contraintes locales, un centre de décision adapte et prévoit son comportement d'une part, et communique d'autre part des décisions qui vont elles-mêmes influer sur la prise de décision des autres centres.

Dans [Erschler *et al.* 96], les auteurs définissent une approche par coopération basée sur le concept de réseau de centres de décision. Leur approche s'inspire des travaux orientés contraintes, développés au LAAS [Erschler 76], [Huguet 94]. Une approche par contraintes permet, à partir d'un problème décrit en terme de variables et contraintes à satisfaire, de détecter des incohérences, d'affiner le domaine des variables, et de caractériser un ensemble de solutions pour faciliter la prise de décision. Dans le cadre du concept de réseau de centre de décisions, les variables et les contraintes du problème sont donc réparties entre les centres de décisions et les paramètres mis en jeu dans les contraintes sont susceptibles d'être négociés entre les centres. La solution globale résulte alors des décisions locales, ces dernières mettant en jeu des processus de négociation. Enfin, un réseau de centres de décision peut être ainsi représenté par un graphe dont les sommets représentent les centres de décision, et les arcs orientés l'action de la prise de décision d'un centre sur les contraintes d'un autre

centre. De ce fait, la décision dans un centre est contrainte par les décisions prises par les centres en amont et elle contraint elle même les décisions des centres en aval (cf. Figure 5.4).

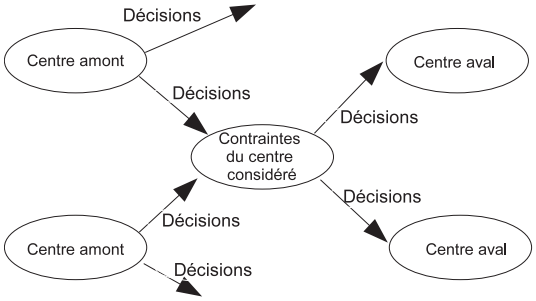

FIG. 5.4: Réseau de centres de décision

La prise de décision dans un centre peut présenter à la fois un caractère local lié aux objectifs propres à ce centre, et aussi un caractère distribué lié à son impact vis-à-vis des autres centres (amont et aval) inter-agissant avec ce centre.

Dans [Huguet 94], l'auteur définit quatre modes de coopération entre centres de décisions :

- *Négociation aval* : un centre de décision ayant pris une décision cherche à la valider avec les centres situés en aval et chargés de la mettre en œuvre.
- *Négociation amont* : un centre de décision coopère avec un centre amont qui cherche à valider une de ses décisions.
- *Renégociation aval* : soit un centre de décision sollicite un centre aval pour modifier une décision précédemment prise, soit un centre est sollicité par un centre aval pour modifier une décision précédemment acceptée.
- *Renégociation amont* : soit un centre de décision sollicite un centre amont pour modifier une décision précédemment acceptée, soit un centre est sollicité par un centre amont pour modifier une décision.

Pour qu'une décision prise localement au niveau d'un centre soit consistante vis-à-vis du fonctionnement global du réseau de centres de décision, elle doit être à la fois [Erschler *et al.* 96] :

- **cohérente** vis-à-vis des décisions à prendre en amont. Cela se traduit par la nécessité de respecter les décisions prises en amont ou de les remettre en cause explicitement ;
- **robuste** vis-à-vis des décisions à prendre en aval. Cela se traduit par la nécessité de prendre des décisions qui pourront être mises en œuvre par l'aval ou d'accepter que ces décisions soient remises en cause.

Il en découle que les décisions prises par un centre doivent être à la fois cohérentes et robustes. La cohérence est d'autant plus facile à assurer que les décisions en amont sont robustes, et la robustesse vise donc ici à favoriser la cohérence des décisions en aval [Despontin-Monsarrat 04].

5.4 Synthèse

L'environnement d'application des ordonnancements étant par essence incertain, un ordonnancement prévu pour exécution ne sera généralement pas l'ordonnancement effectivement réalisé. Ce constat a poussé récemment de nombreux chercheurs à développer des méthodes d'ordonnancement sous incertitudes. Dans ce chapitre, nous avons commencé par passer en revue ces différentes approches d'ordonnancement sous incertitudes. Elles sont classées selon la terminologie proposée par [Davenport & Beck 00] en trois catégories : les approches réactives, les approches proactives et les approches proactives-réactives. Ces approches se distinguent essentiellement par les phases (hors-ligne et en-ligne) au cours desquelles sont anticipées les perturbations. Les approches d'ordonnancement proactives tiennent compte des incertitudes lors du calcul, hors-ligne, d'un ordonnancement prédictif initial ; elles supposent que les informations sur les perturbations qui peuvent surgir sont connues à l'avance, mais, ne permettent pas d'anticiper hors-ligne toutes les perturbations, certaines étant en effet imprévisibles. De manière symétrique, les approches réactives calculent l'ordonnancement en temps réel et font face aux incertitudes à fur et à mesure de leur apparition. Elles sont certes plus appropriées lorsque l'incertitude est complètement inconnue, mais fournissent en général un ordonnancement de faible performance et non évaluable à priori du fait qu'on ne connaît l'ordonnancement qu'une fois celui-ci est réalisé. Enfin, remarquons que la prise en compte des perturbations, lors de la construction de la solution d'ordonnancement, à la fois en phases en-ligne et hors-ligne, ne peut être qu'avantageux ; il est alors nécessaire de coupler l'approche proactive avec un algorithme réactif afin de réagir en-ligne aux aléas qui ne peuvent pas être absorbés par l'algorithme proactif.

Dans le cadre des travaux menés dans cette dernière partie du manuscrit sur la problématique d'ordonnancement en présence de perturbations, nous nous intéressons à la classe d'approches proactive-réactives qui permettent d'anticiper les incertitudes par construction d'un ordonnancement robuste et qui présentent une performance stable relativement à un ensemble de perturbations possibles. Plus précisément, notre intérêt s'est porté sur une méthode d'ordonnancement robuste offrant de la flexibilité séquentielle et pouvant être utilisée pour caractériser hors-ligne un ensemble de solutions possibles, exploitables en-ligne pour le choix de la solution la mieux adaptée à la situation réelle. Dans le chapitre suivant, nous présentons cette approche robuste basée sur un concept de dominance (cf. Chapitre 3).

D'un autre point de vue, nous savons que dans la plupart des approches en-ligne ou hors-ligne existantes en recherche opérationnelle, l'ordonnancement est vu comme une fonction globale gérant l'organisation de la totalité des ressources. Seulement dans la pratique, les ressources de production sont caractérisées par des besoins et contraintes spécifiques, et disposent ainsi d'une autonomie de décision, favorable à la réactivité, dont il est fondamental de tenir compte. C'est pour cela qu'il est intéressant de préconiser une organisation distribuée des décisions pour la gestion d'un système de fabrication en présence de perturbations, i.e., des décisions à répartir entre les différents acteurs du système dont la cohérence globale est assurée par coopération.

Nous nous sommes alors penchés, dans la seconde partie de ce chapitre, à poser quelques définitions sur la coopération, et à décrire les formes de la coopération et son organisation en centres de décisions. Nous avons ensuite passé en revue quelques méthodes basées sur la coopération. Nous avons cité principalement, les approches d'aide à la décision dans le cadre de la gestion de la chaîne logistique, les techniques des systèmes multi-agents et les méthodes de résolution du DiCSP. Notons que la relative nouveauté des approches de résolution de l'ordonnancement distribué et coopératif explique le faible nombre de travaux dans la littérature.

Au vu de ce qui a été retracé dans ce chapitre, nous proposons dans les chapitres suivants une nouvelle approche d'ordonnancement avec prise en compte des perturbations. C'est une approche qui permet à la fois d'anticiper et de réagir en temps réel aux aléas et qui intègre une dimension distribuée dans sa méthode de résolution. Elle s'inscrit dans le cadre d'une approche robuste (apport de flexibilité séquentielle), et fait appel à la coopération entres les différents acteurs constituant le système de fabrication (distribution des décisions) pour l'ordonnancement global du système.

Chapitre 6

Élements préliminaires

6.1 Introduction

Le champ d'application de l'ordonnancement considéré dans cette partie de thèse est celui des systèmes de production automatisés. On considère un atelier de fabrication de type job-shop. Le problème consiste à ordonnancer un ensemble $T = \{1, ..., n\}$ de travaux sur un ensemble M de m machines $M = \{M_1, ..., M_m\}$. Chaque travail $i \in T$ est caractérisé par une gamme opératoire et est donc constitué de $O_i = \{1, ..., n_i\}$ opérations devant chacune s'exécuter sur une des m machines de M dans l'ordre défini par la gamme opératoire. La j-ème opération du travail i est notée o_{ij} et possède une durée opératoire p_{ij}. Elle est réalisée par une et une seule ressource m_{ij} disponible en un exemplaire unique. L'objectif est de minimiser la durée totale d'exécution (makespan) notée C_{max} (L_{max} si des délais sont associés au travaux). Ce problème $J_n||C_{max}$ est NP-difficile [Lenstra *et al.* 77].

Classiquement, il est possible de décomposer le problème $J_n||C_{max}$ en m sous problèmes à une machine interdépendants, où chaque opération o_{ij} est caractérisée par une fenêtre d'exécution $[r_{ij}, d_{ij}]$. Les sous-problèmes sont interdépendants car la séquence optimale des opérations déterminée sur une machine donnée doit être cohérente (au sens des gammes opératoires) avec les séquences d'opérations trouvées sur les autres machines [Adams *et al.* 88]. Dans chaque sous-problème, il s'agit de minimiser le plus grand retard algébrique (problème noté $1|r_j|L_{max}$).

On désigne par f et e les deux opérations fictives representant respectivement le début et la fin de l'ordonnancement. Les dates de disponibilité ainsi que les dates échues des tâches peuvent alors être déterminées ainsi que précisé dans [Adams *et al.* 88] par les formules suivantes :

$$r_{ij} = L(f, (i, j))$$

et

$$d_{i,j} = L(f, e) - L((i, j), e) + p_{ij}$$

où $L(x, y)$ est la longueur du plus long chemin de x à y sur le graphe conjonctif du problème qui peut être déterminée grâce à l'algorithme de Bellman-Ford [Esquirol & Lopez 99].

6.2 Une approche d'ordonnancement robuste à une machine

Nous avons vu dans la Partie II que le Théorème des pyramides permet de caractériser pour chaque problème V défini par une structure d'intervalles, un ensemble de séquences dominantes S_{dom} dont la cardinalité peut être calculée. Cet ensemble dépend de l'ordre total existant entre les dates de disponibilité r_j et les dates d'échéance d_j des travaux, mais ne dépend pas des valeurs numériques de ces dates (tant que l'ordre total est respecté) ainsi que des valeurs des durées opératoires.

Notons que la qualité de cet ensemble dominant peut être calculée en temps polynomial (cf. [La 04]). En effet, étant donné un problème V et son ensemble S_{dom} de séquences dominantes, il est possible d'associer à chaque travail j un intervalle de retard noté $[L_j^{min}, L_j^{max}]$, où L_j^{min} (resp L_j^{max}) désigne le meilleur (resp le pire) retard algébrique de j parmi toutes les séquences de S_{dom}. Cet intervalle de retard se calcule en temps polynomial grâce à la détermination, pour chaque travail j, des séquences au mieux et au pire, c'est-à-dire des séquences induisant le plus petit et le plus grand retard pour le travail considéré parmi toutes les séquences de S_{dom}. Dans la suite, nous montrons comment sont calculées ces performances.

6.2.1 Évaluation des performances d'un ensemble dominant

On rappelle que dans la suite, $u(j)$ et $v(j)$ désignent respectivement les indices de la première et de la dernière pyramide à laquelle peut être affecté un travail j.

6.2.1.1 Calcul de la performance au mieux d'un travail

Pour calculer $L_j^{\min}$, il faut minimiser la longueur du chemin critique associé à j. Cela revient à affecter j à la première pyramide d'indice $u(j)$ et de ne considérer que les travaux nécessairement séquencés, en cohérence avec la structure de pyramides, avant j (i.e. l'ensemble $Pred_j^{\min}$ des travaux k tel que $v(k) < u(j)$).

Déterminer $L_j^{\min}$ revient alors à minimiser la durée totale $C_{\max}$ du problème d'ordonnancement constitué des travaux de l'ensemble $Pred_j^{\min}$, et on peut donc relâcher les dates d_j des travaux ($d_j = D$, D étant une constante grande). La structure d'intervalles de $Pred_j^{\min}$ ainsi obtenue définissant une structure en escalier, une séquence $S_j^{\min}$ optimale

minimisant le $C_{\max}$ est obtenue par application de la règle de Jackson, en séquençant les travaux dans l'ordre croissant de leurs dates de début.

La séquence $(S_j^{\min}, j)$, construite comme illustrée sur la Figure 6.1, définit alors la séquence induisant le plus petit retard pour le travail j. On parle aussi de séquence la plus favorable.

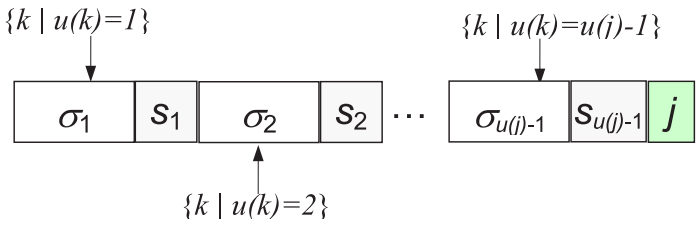

FIG. 6.1: La séquence la plus favorable du travail j

En notant $s_k(S_j^{\min})$ les dates de début des travaux $k \in Pred_j^{\min}$ dans $S_j^{\min}$ et, $C_{\max}(S_j^{\min})$ le makespan correspondant, on a alors :

$$C_{\max}(S_j^{\min}) = \max(s_k(S_j^{\min}) + p_k), \forall k \in Pred_j^{\min}$$

D'où :

$$L_j^{\min} = \max(C_{\max}(S_j^{\min}), r_j) + p_j - d_j$$

Pour calculer $L_j^{\min}$, un algorithme est décrit dans [La 04]. Sa complexité temporelle est en $O(n \log n)$.

6.2.1.2 Calcul de la performance au pire d'un travail

De façon symétrique au cas précédent, le calcul de $L_j^{\max}$ nécessite de chercher la séquence de S_{dom} qui maximise le retard du travail j. Pour cela, on doit affecter j à la pyramide d'indice $v(j)$, c'est-à-dire le plus tard possible. Nous notons $Pred_j^{\max}$ l'ensemble des travaux k pouvant être séquencés avant j (i.e. les travaux k tels que $u(k) \leq v(j)$). Trouver $L_j^{\max}$ consiste alors à maximiser le makespan $C_{\max}$ des travaux de $Pred_j^{\max}$ tout en respectant le Théorème des pyramides.

La structure d'intervalles ainsi obtenue étant à nouveau "en escalier", il est possible d'utiliser la version inversée de la règle de Jackson qui consiste à classer les travaux par

ordre de r_i décroissant pour maximiser leur $C_{\max}$. Toutefois, la séquence obtenue ne respecterait alors pas le Théorème des pyramides. Pour pallier ce problème, on distingue d'une part les travaux k tels que $v(k) < v(j)$, (*i.e.* appartenant à des pyramides antérieures à celle de j) et, d'autre part, les travaux k tels que $u(k) \leq v(j) \leq v(k)$ (*i.e.* pouvant être affectés à la même pyramide que celle de j).

Afin de maximiser la longueur du chemin critique associé à j, les travaux k tels que $v(k) < v(j)$ sont affectés à la pyramide d'indice $v(k)$, de façon à ce qu'ils soient ordonnancés le plus tard possible. Les affectations des travaux aux pyramides étant établies, le problème de maximiser le $C_{\max}$ des travaux situés dans les $v(j)-1$ pyramides d'indice inférieur à $v(j)$ se décompose alors en $v(j)-1$ problèmes de maximisation indépendants. Chaque problème consiste à maximiser le makespan $C_{\max}^{P_i}$ des travaux affectés à la pyramide P_i. Il peut être résolu optimalement, avec respect du Théorème des pyramides, en construisant une sous-séquence dans laquelle les travaux sont séquencés par ordre de d_i croissant. La séquence S_j^1 conduisant au C_{max} le plus grand des travaux situés dans les pyramides d'indice inférieur à $v(j)$ correspond alors à la juxtaposition des sous-séquences déduites précédemment.

Pour maximiser le retard de j, les travaux k tels que $u(k) \leq v(j) \leq v(k)$ sont affectés à la pyramide d'indice $v(j)$. Le problème est alors de déterminer quelle séquence, parmi celles de l'ensemble dominant plaçant j en dernier dans la pyramide $v(j)$, possède le C_{max} le plus grand. Il est montré que cette séquence correspond à celle où le plus de travaux sont placés entre le sommet de la pyramide $v(j)$ et la tâche j. Pour l'obtenir, il suffit de placer toutes les travaux k tels que $d_k > d_j$ avant le sommet et tous les travaux restants après le sommet (cf. Figure 6.2). En effet, si les travaux k tels que $d_k > d_j$ étaient placés après le sommet, ils seraient séquencés après j (cf. Théorème 3.2.2). En suivant cette règle, il est donc garanti que le travail j est mis à la dernière position dans la pyramide d'indice $v(j)$ et que le C_{max} des travaux précédant j est maximisé. Soit S_j^2 la séquence construite par application de cette règle.

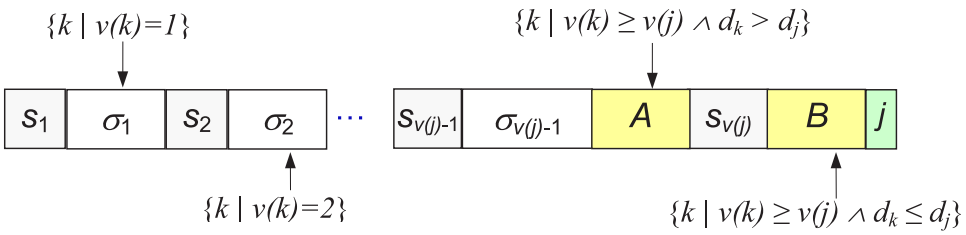

FIG. 6.2: La séquence la plus défavorable de la tâche j

Si S_j^{max} dénote la juxtaposition des deux séquences S_j^1 et S_j^2 précédemment déterminées (i.e. $S_j^{max} = S_j^1 \prec S_j^2$) et $C_{max}(S_j^{max})$, le makespan correspondant à cette séquence, on a alors :

$$L_j^{\max} = \max(C_{\max}(S_j^{max}), r_j) + p_j - d_j$$

Pour calculer $L_j^{\max}$, un algorithme est décrit dans [La 04]. Sa complexité temporelle est en $O(n \log n)$.

6.2.2 Intervalles des retards et adaptation des retards

Il est possible, étant donné un problème V_k associé à une ressource k donnée, et son ensemble $S_{dom}^{V_k}$ de séquences dominantes déterminé selon le Théorème des pyramides, d'associer à chaque travail j de V_k un intervalle de retard $[L_j^{min}, L_j^{max}]$ calculé comme décrit dans les paragraphes précédents.

La connaissance des retards aux mieux et au pire de chaque travail permet de déduire des bornes inférieure et supérieure pour le retard algébrique optimal L_{max}^* :

$$\max(L_j^{min}) \leq L_{max}^* \leq \max(L_j^{max}), \forall j$$

La performance d'un ensemble dominant de séquences peut être alors jugée acceptable ou non, relativement à la performance au pire. Étant donnée la structure d'intervalles associée au problème à 6 travaux du tableau 3.3, les données suivantes correspondent aux valeurs des r_j et d_j relatives à l'ordre partiel de l'exemple, et aux durées opératoires des travaux : $r_4 = 0 < r_5 = 10 < r_3 = 11 < r_1 = 13 < d_1 = 21 < d_3 = 23 < r_6 = 26 < d_4 = 27 < r_2 = 30 < d_2 = 34 < d_5 = 35 < d_6 = 43$; $p_1 = 5$, $p_2 = 2$, $p_3 = 7$, $p_4 = 5$, $p_5 = 8$, $p_6 = 3$.

Le tableau 6.1 donne les valeurs des retards L_j^{min} et L_j^{max} ainsi que les séquences au mieux et au pire ayant permis leur calcul.

On remarque d'après ce tableau que quel que soit la séquence considérée de S_{dom}, le travail 6 ne sera jamais en retard. On a aussi $-2 \leq L_{max}^* \leq 9$ sachant que la valeur optimale de L_{max}^* est égale à 2.

On note aussi que les valeurs de L_j^{min} et L_j^{max} permettent de déduire les valeurs des dates de début au mieux et au pire s_j^{min} et s_j^{max} de chaque travail j :

$$s_j^{min} = L_j^{min} + d_j - p_j$$

$$s_j^{max} = L_j^{max} + d_j - p_j$$

Le calcul des retards au mieux et au pire permet au décideur d'avoir une vision globale des performances associées à l'ensemble des séquences dominantes S_{dom} du point de vue du retard algébrique. En effet, dans le cas de la production à la commande, le décideur peut juger si un retard au pire est acceptable ou non et ce relativement aux délais de livraison arrêtés avec le client. Dans la négative, une réduction des dates au pire serait possible en agissant sur la structure d'intervalles. Ceci conduit à éliminer, au détriment de la flexibilité, les séquences de l'ensemble dominant qui produisent des retards non acceptables.

Travail j	$S_j^{\min}$	L_j^{min}	$S_j^{\max}$	L_j^{max}
1	1	-3	$4 \prec 5 \prec 3 \prec 1$	9
2	$4 \prec 3 \prec 1 \prec 2$	-2	$1 \prec 3 \prec 4 \prec 5 \prec 6 \prec 2$	6
3	3	-5	$4 \prec 5 \prec 1 \prec 3$	7
4	4	-22	$5 \prec 3 \prec 1 \prec 4$	8
5	5	-17	$1 \prec 3 \prec 4 \prec 6 \prec 2 \prec 5$	8
6	$4 \prec 5 \prec 3 \prec 1 \prec 6$	-10	$1 \prec 3 \prec 4 \prec 2 \prec 5 \prec 6$	0

TAB. 6.1: Tableau des retards obtenu pour l'exemple de la Figure 3.3

Si on considère de nouveau l'exemple de la Figure 3.3 et son tableau associé des retards 6.1, la séquence dominante qui produit $\max L_i^{max} = L_1^{max} = 9$ est $4 \prec 5 \prec 3 \prec 1$. Pour réduire le retard maximal de 1, sachant que ce dernier est un sommet, il et possible d'augmenter la date de début du travail 3 (le travail le plus proche de 1) à la valeur de r_1 i.e. $r_3 \longleftarrow r_1$ $(1 \prec 3)$, et ainsi on obtient un nouveau tableau des retards donné par Tab 6.2.

On note que le changement opéré sur la structure d'intervalles dans le but d'augmenter la performance a produit une diminution de la flexibilité séquentielle. Cette flexibilité qui était de $(1+1)^3.(1+2)^1 = 24$ passe à $(1+1)^2.(1+2)^1 = 12$, et diminue de moitié.

Travail j	$S_j^{\min}$	L_j^{min}	$S_j^{\max}$	L_j^{max}
1	1	-3	$4 \prec 5 \prec 1$	2
2	$4 \prec 5 \prec 1 \prec 3 \prec 2$	-2	$1 \prec 3 \prec 4 \prec 5 \prec 6 \prec 2$	6
3	3	-3	$4 \prec 5 \prec 1 \prec 3$	7
4	4	-22	$5 \prec 1 \prec 3 \prec 4$	3
5	5	-17	$1 \prec 3 \prec 4 \prec 5$	8
6	$4 \prec 5 \prec 1 \prec 3 \prec 6$	-10	$1 \prec 3 \prec 4 \prec 5 \prec 2 \prec 6$	0

TAB. 6.2: Tableau des retards obtenu pour l'exemple de la Figure 3.3 avec $r_3 = r_1$

6.3 Job shop et ordonnancements locaux robustes

Dans le problème job shop, l'ensemble des opérations à ordonnancer sont distribuées sur l'ensemble des ressources. Nous rappelons qu'un travail i compte n_i opérations et est défini par une gamme opératoire. La $j^{\text{ème}}$ opération du travail i notée o_{ij} précède l'opération $o_{i,j+1}$ ($o_{ij} \prec o_{i,j+1}$) et s'exécute pendant un temps p_{ij} sur la machine m_{ij}. Nous considérons que chaque ressource m_k gère l'ordonnancement de l'ensemble $V_k = \{o_{ij} | m_{ij} = k\}$ des opérations devant être exécutées sur cette même ressource. On suppose qu'un intervalle d'échéance $[d_{i,n_i+1}]$ (contracté avec le client du produit i) est associé à chaque travail i. Clairement, il n'existe pas de contraintes de précédence entre les opérations affectées au même centre de décision. Si nous nous focalisons sur le problème à résoudre au niveau de chaque ressource m_x, nous considérons que la solution est un ordonnancement local construit selon l'approche décrite dans la section précédente. L'ordonnancement est alors robuste, et chaque opération o_{ij} de V_x est caractérisée par une date de lancement r_{ij} et une date d'échéance d_{ij}, ainsi qu'une performance au pire L_{ij}^{max} (resp. au mieux L_{ij}^{min}) ; notons aussi $u(ij)$ (resp. $v(ij)$) l'indice de la première (resp. dernière) pyramide à laquelle peut être affectée l'opération o_{ij}.

Dans le cadre d'un job shop multi-acteurs, si on considère un sous-problème à une machine avec des fenêtres de lancement $[r_{ij}]$ et des fenêtre d'achèvement $[d_{ij}]$, les dates de disponibilité r_{ij} et d'échéance d_{ij} ne sont pas connues. De ce fait, le théorème des pyramides ne peut être appliqué dans la mesure où un ordre total entre les r_{ij} et les d_{ij} des opérations à ordonnancer sur la machine n'existe pas. Nous reviendrons sur le problème de détermination des ordres totaux pour chaque centre de décisions dans les prochains chapitres. Notons entre autre, qu'il est possible, étant donnée une structure d'intervalles associée au problème à une machine (ordre total donné entre les opérations) de définir des dates de lancement et d'échéance de toute opération, dans la limite des intervalles délimités par les dates de début au plus tôt (resp. plus tard) au mieux et au pire, elles mêmes déduites des séquences au plus tôt au mieux (la plus favorable) et au pire (la plus défavorable) données respectivement plus haut, par les Figures 6.1 et 6.2, et des séquences au plus tard au mieux et au pire. Dans ce qui suit, nous introduisons ces notions de dates et les séquences dominantes permettant de les déduire.

Nous considérons un ensemble d'opérations à ordonnancer sur une machine constitué de n opérations et m sommets. Par souci de clarté, on note une opération o_{uv} par j.

Une *date de début au plus tôt et au mieux* d'une opération j notée est_j^b peut être déduite de la séquence la plus favorable de l'opération. C'est une date déterminée dans le meilleur des cas en considérant que l'opération j ainsi que ses prédécesseurs dans la séquence au mieux commencent au plus tôt et juste après leurs date de disponibilité (au plus tôt et à r_j constant). La séquence au plus tôt au mieux permettant de déduire cette date de début au plus tôt (earliest) au mieux (best) de j notée $Seq_e^b(j)$: e pour earliest et b pour best,

est donnée comme suit :

$$Seq_e^b(j) = \sigma_1 \prec t_1 ... \prec \sigma_k \prec t_k ... \prec \sigma_{u(j)-1} \prec t_{u(j)-1} \prec j \tag{6.1}$$

Avec $\sigma_i = \{k|u(k) = i\}$, $\forall i$ sommet tel que $i < u(j)$.

Une autre *date de début au plus tôt mais au pire* notée est_j^w peut être déduite de la séquence la plus défavorable de l'opération. C'est une date de début déterminée dans le pire des cas, calculée en considérant que l'opération j ainsi que tous ses prédécesseurs dans la séquence au pire, commencent leur exécution au plus tôt et juste après leurs dates de disponibilité (au plus tard à r_j constant). La séquence au plus tôt (earliest) et au pire (worst) de j notée $Seq_e^w(j) : w$ pour worst, permettant de déduire cette date est la suivante :

$$Seq_e^w(j) = t_1 \prec \sigma_1 \prec ... \prec t_k \prec \sigma_k ... \prec t_{v(j)-1} \prec \sigma_{v(j)-1} \prec t_{v(j)} \prec \sigma_{v(j)} \prec j \tag{6.2}$$

Avec $\sigma_i = \{k|v(k) = i\}$ $\forall i < v(j)$, et $\sigma_i = \{k|u(k) = i$ ou $v(k) = i\}$ pour $i = v(j)$.

Une *date de début au plus tard au mieux* notée lst_j^b peut être déterminée à partir de la séquence la plus favorable, en considérant que l'opération ainsi que tous ses successeurs finissent leur exécution à la limite de leurs dates échues (au plus tôt à d_j constant). Cette date peut être déduite de la séquence au plus tard (latest) au mieux (best) de j notée $Seq_l^b(j) : l$ pour latest, donné comme suit :

$$Seq_l^b = j \prec t_{v(j)+1} \prec \sigma_{v(j)+1} ... \prec t_k \prec \sigma_k ... t_m \prec \sigma_m \tag{6.3}$$

Avec $\sigma_i = \{k|v(k) = i\}$, $\forall i$ sommet tel que $i > v(j)$.

Enfin, une *date de début au plus tard et au pire* noté lst_j^w peut être déterminée à partir de la séquence au pire, en considérant que l'opération ainsi que tous ses successeurs finissent leur exécution à la limite de leurs dates échues (au plus tard à d_j constant). Cette date peut être déduite de la séquence plus tard (latest) au pire (worst) suivante :

$$Seq_l^w = j \prec \sigma_{v(j)} \prec t_{u(j)} \prec \sigma_{u(j)+1} \prec t_{u(j)+1} ... \sigma_k \prec t_k ... \sigma_m \prec t_m \tag{6.4}$$

Avec $\sigma_i = \{k|u(k) = i\}$, $\forall i > u(j)$ et $\sigma_i = \{k|v(k) = i$ ou $u(k) = i\}$ pour $i = u(j)$.

En se limitant au seul domaine caractérisé par l'ensemble de séquences dominantes, l'intervalle $[\underline{s_j}, \overline{s_j}]$ délimitant la date de début au pire d'une opération j est $[lst_j^w, est_j^w]$, en supposant que $lst_j^w \leq est_j^w$, i.e., au plus tôt (à r_j constant) et au pire, l'opération finirait sa réalisation avant son deadline (cf. Figure 6.3 qui montre un exemple de positionnement des dates de début et de fin d'une opération j dans les situations la plus favorable et la moins favorable, et cela relativement aux dates de disponibilité r_j et d'échéance d_j) ; dans

le cas contraire l'intervalle serait $[lst_j^w, d_j - p_j]$. Signalons que cet intervalle peut évoluer au fil du temps en fonction de la dynamique du centre ; sa limite inférieure étant est_j^b, i.e., la date de début au plus tôt de j dans la situation la plus favorable.

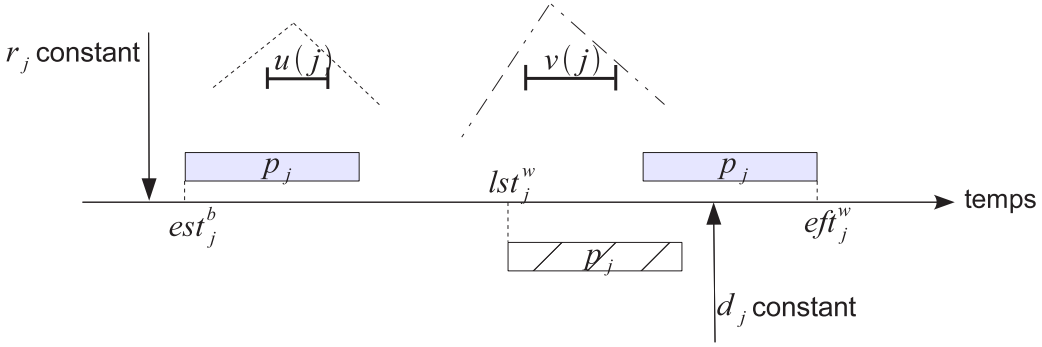

FIG. 6.3: Domaine de variation de j dans la limite de l'ensemble dominant

6.4 Discussion

Dans ce chapitre, nous avons décrit une approche existante d'ordonnancement robuste à une machine [La 04], produisant de la flexibilité séquentielle et permettant de faire face aux incertitudes. Cette flexibilité, induisant de la flexibilité temporelle, est utilisée pour caractériser hors-ligne un ensemble de solutions exploitables en-ligne.

Généralement, plus la flexibilité séquentielle est grande, plus la capacité de réagir aux aléas est importante et, plus aussi la performance de l'ordonnancement est faible ; le contraire étant aussi vrai. En effet, nous avons montré à travers un exemple, que l'augmentation de la performance de l'ordonnancement se traduit par une diminution de la flexibilité de l'ensemble dominant D'où l'intérêt et la nécessité d'assurer un compromis entre flexibilité et cohérence.

Dans cette dernière partie du manuscrit, nous nous intéressons au problème d'ordonnancement à plusieurs ressources évoluant dans un contexte perturbé. Une approche d'ordonnancement coopérative est proposée (Chapitres 7 et 8). L'idée de base est que, chaque ressource assimilée à un centre de décision, gère son propre ordonnancement local ; l'ordonnancement local est un ordonnancement robuste à une machine disposant d'une flexibilité séquentielle pour absorber les aléas et dont la méthode de calcul est décrite dans ce Chapitre 6 ; l'ordonnancement global est le résultat de la coopération entre les divers acteurs.

Chapitre 7

Schéma de coopération proposé

Dans la Section 5.2, nous avons mis en évidence l'intérêt de distribuer la fonction ordonnancement afin de parvenir à une solution globale issue d'une coopération entre plusieurs acteurs. L'objectif de ce chapitre est de décrire le contexte de l'étude et de détailler les mécanismes de coopération inter-ressources adoptés, en identifiant les liens entre les ressources. La notion d'intervalle consommation/livraison est définie en tant qu'objet de coopération. Les notions de cohérence, de risque d'incohérence et de flexibilité sont aussi introduites.

7.1 Contexte

Nous rappelons que le champ d'application de l'ordonnancement considéré dans cette étude est celui des systèmes de production automatisés (job shop). Les ressources sont celles de l'atelier de production. Elles sont supposées disponibles en un exemplaire unique (ressources disjonctives). Les opérations nécessaires à la réalisation des travaux doivent s'enchaîner dans le respect des gammes opératoires associées aux différents travaux.

L'ordonnancement est réalisé de façon distribuée sur un ensemble d'acteurs. En effet, comme nous l'avons déjà évoqué dans la Section 5.3.3, nous assimilons les ressources appartenant à l'atelier de production à des centres de décisions (CDDs). La fonction ordonnancement est répartie sur les différents CDDs. Chaque CDD gère l'ensemble des opérations devant être exécuté par la ressource. Si on s'intéresse au problème à résoudre au niveau de chaque centre, il s'agit alors d'un problème à une machine avec fenêtres de disponibilité et d'échéance, l'objectif étant la recherche d'un séquencement réalisable des opérations sur la ressource. Nous considérons ainsi que chaque centre gère son propre ordonnancement local et dispose de sa propre flexibilité décisionnelle (ordonnancement robuste). Cette flexibilité est séquentielle et correspond au nombre de séquences dominantes que le Théorème des pyramides caractérise, étant donnée la structure d'intervalles définie par les r_{ij} et d_{ij} des opérations réalisées par le CDD. Les différents CDDs constituant le job shop fixent leurs

décisions en coopérant dans la perspective d'aboutir à une solution globale satisfaisante. Nous reviendrons sur cette notion de satisfaction dans la suite du texte.

Comme l'illustre la Figure 7.1, les centres de décision sont reliés par des flux de produits et des flux d'informations (données et décisions). En considérant un CDDi particulier, il est possible de distinguer, selon les gammes opératoires associées aux produits manufacturés, supposées initialement connues, ses CDDs amont et ses CDDs aval. Le CDDi reçoit des centres de décision amont les produits qu'il doit transformer. Et une fois les produits transformés, il les transfère à son tour aux centres de décision aval chargés de la prochaine transformation de la gamme.

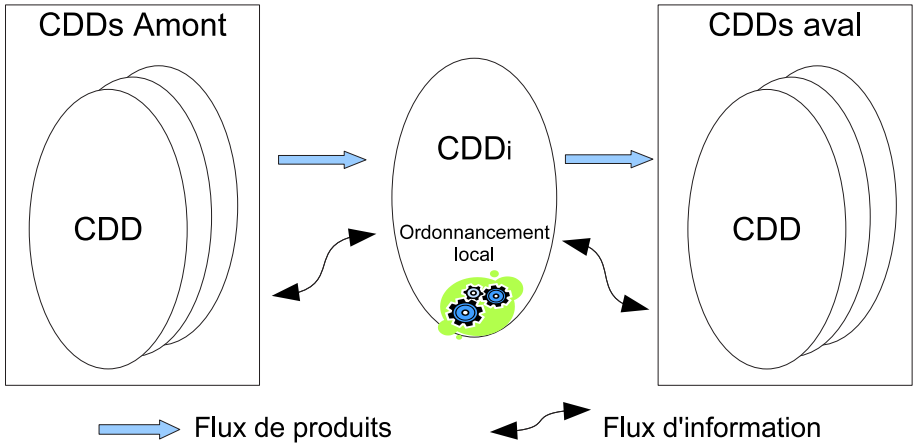

FIG. 7.1: Relations entre centres de décision d'un atelier

Remarquons qu'ici, l'atelier étant de type job shop, la distinction entre amont et aval est logique, i.e., un CDD se trouvant en amont d'un CDD peut également se trouver en aval, celui-ci pouvant à la fois être fournisseur pour un CDDi d'un produit donné et consommateur d'un autre produit que le même CDDi transforme.

Nous supposons que les travaux ne sont pas tous connus initialement, et qu'un ordre de fabrication peut apparaître à tout moment et concerne la fabrication d'une instance unique de produit, devant être achevée pour une date d'échéance souhaitée (spécifiée par le demandeur du produit ou bien négociée entre le client et le décideur). L'ordonnancement est donc adapté en temps réel (ordonnancement réactif), au fur et à mesure de la réalisation des activités et /ou de l'arrivée de nouveaux ordres de fabrication. À chaque occurrence d'un événement, un centre de décision définit, dans le cas où un nouveau produit est pris en charge, la date de livraison du produit, et éventuellement les nouvelles dates de livraison des autres travaux que la ressource doit réaliser si les anciennes valeurs ont été modifiées.

Comme évoqué précédemment, nous supposons que chaque CDD coopère avec ses

centres amont (fournisseurs de produit) et aval (consommateur de produit). Classiquement, chaque centre souhaite avoir le produit à réaliser dans un délai de disponibilité de manière à préserver au mieux l'avenir, i.e., conserver une capacité de travail pour être capable de gérer les imprévus. De la même manière, chaque centre de décision souhaite livrer un produit dans un délai donné de manière à pouvoir absorber, le cas échéant, les aléas qui peuvent surgir. L'objectif de la coopération inter-ressources est donc d'amener les acteurs à expliciter des marges de sécurité que chacun se réserve pour définir les dates de livraison des produits, tout en respectant l'autonomie décisionnelle (la flexibilité) de chaque acteur afin de pouvoir réagir aux perturbations.

7.2 Mécanisme de coopération

Dans le contexte de cette étude, les conversations menées entre les acteurs ont pour but d'amener chaque acteur à s'engager pour une mise à disposition (livraison) d'un produit auprès du centre aval chargé de sa transformation dans un intervalle temporel. La difficulté de la recherche réside alors dans la détermination de mécanismes de coopération (négociation/renégociation) efficaces convergeant rapidement vers des décisions satisfaisantes.

Nous supposons que chaque centre de décision coopère avec ses centres amont et aval selon un mode de communication point à point, i.e., la communication est individuelle entre les deux CDDs et n'est pas diffusée au réseau de CDDs. Le but de la coopération est d'amorcer des conversations entre les paires de CDDs dans la perspective de dimensionner l'intervalle de livraison de chaque produit. Ainsi, comme schématisé sur la Figure 7.2, si on considère un CDDi devant exécuter un ensemble de travaux V_i, celui-ci doit négocier pour chaque opération $o_{uv} \in V_i$:

- en amont avec le CDD réalisant l'opération $o_{u,v-1}$ de sorte à s'entendre pour définir un intervalle temporel $[r_{uv}^{min}, r_{uv}^{max}]$, noté $[r_{uv}]$, dans lequel le produit ayant subi l'opération $o_{u,v-1}$ sera disponible ;

- en aval avec le CDD réalisant $o_{u,v+1}$ de sorte à s'entendre pour fixer un intervalle temporel $[d_{uv}^{min}, d_{uv}^{max}]$, noté $[d_{uv}]$, dans lequel le produit ayant subi l'opération o_{uv} sera livré par CDDi.

L'intervalle de début $[r_{uv}]$ de la ressource réalisant l'opération o_{uv} correspond, pour la ressource réalisant $o_{u,v-1}$, à son intervalle de fin $[d_{u,v-1}]$. De la même manière, l'intervalle de fin $[d_{uv}]$ de la ressource réalisant l'opération o_{uv} correspond, pour la ressource réalisant $o_{u,v+1}$, à son intervalle de début $[r_{u,v+1]}$.

On peut alors écrire :

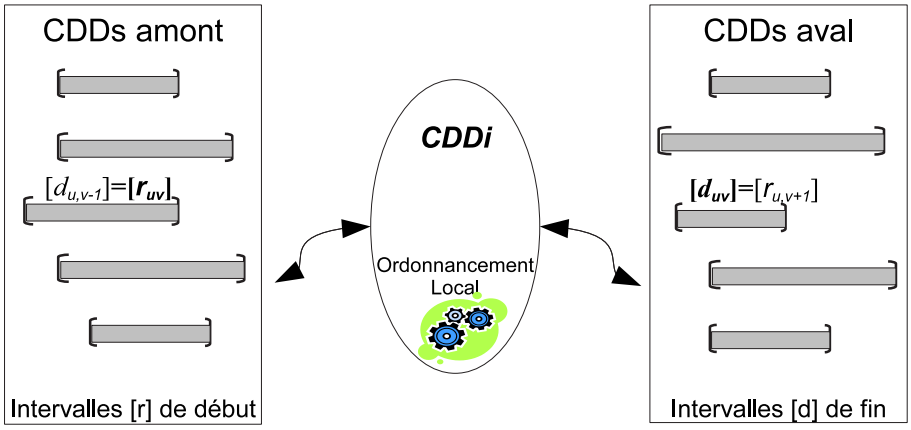

FIG. 7.2: Interaction d'un CDD avec son environnement

$$[d_{ij}] = [r_{i,j+1}], \forall j \in O_i, \forall i \in T$$

On remarque que $[d_{i,n_i}] = [r_{i,n_{i+1}}]$ est l'intervalle de livraison contracté entre la ressource réalisant la dernière opération $n_i \in O_i$ du travail $i \in T$, et le client demandeur du produit i.

Nous supposons que les intervalles $[r_{uv}]$ et $[d_{uv}]$ négociés entre les centres sont contractuels et qu'ils correspondent à un engagement mutuel de production/consommation, i.e., le CDD amont s'engage à achever l'opération qu'il a en charge sur un produit dans une fenêtre temporelle fixée et, réciproquement, le CDD aval s'engage à commencer l'exécution de la prochaine opération que le produit doit subir dans cette même fenêtre, ou tout au moins à prendre le produit en charge dans cette fenêtre (cf. Figure 7.2).

Notons que les dates de lancement r_{uv} et les dates d'échéance d_{uv} choisies par le CDDi pour les opérations qu'il prend en charge prennent leur valeurs dans les intervalles $[r_{uv}]$ et $[d_{uv}]$. Ces dates doivent être cohérentes avec les intervalles contractés i.e. :

$$\begin{cases} r_{uv}^{min} \leq r_{uv} \leq r_{uv}^{max} \\ \\ d_{uv}^{min} \leq d_{uv} \leq d_{uv}^{max} \end{cases} \tag{7.1}$$

Se situant dans un contexte d'ordonnancement sous incertitudes, nous signalons que fixer des intervalles de livraison des produits (au lieu d'une date fixe) entre deux CDDs, permet de donner d'avantage de flexibilité à chaque centre pour la gestion de son organisation locale et pour le respect de ses engagements dans un contexte perturbé. Cette flexibilité est donc à considérer en plus de la flexibilité séquentielle dont dispose chaque centre de décision.

Enfin, remarquons que les intervalles contractés ne sont pas figés. Au fil du temps et en faisant face aux perturbations qui peuvent surgir, le besoin de remettre en cause les

engagements passés peut s'imposer. Cela se traduit alors, comme nous le verrons plus tard, par une renégociation des intervalles.

Nous avons, dans cette section, mis l'accent sur la notion d'intervalle de consommation/livraison, servant d'objet de négociation entre les CDDs. Dans ce qui suit, nous décrivons les caractéristique d'un ordonnancement local d'un CDD.

7.3 Ordonnancement local robuste

En considérant que chaque centre de décision gère un ordonnancement local, nous supposons comme énoncé précédemment, que cet ordonnancement est construit selon l'approche décrite dans le Chapitre 6. L'ordonnancement est alors robuste. Étant donné la structure d'intervalles (fenêtres de lancement et d'achèvement) caractérisant le CDD, chaque opération o_{uv} de V_i réalisée par le CDD_i débute sa réalisation dans l'intervalle $[lst^w_{uv}, est^w_{uv}]$ (cf. Section 7.3). Cet intervalle peut évoluer au fil du temps en fonction de la dynamique du centre, sa limite inférieure étant est^b_j, i.e., la date de début au plus tôt de j dans la situation la plus favorable. Pour une raison de clarté, nous mentionnons, dans le reste du document, cet intervalle $[lst^w_{uv}, est^w_{uv}]$ par $[s_j]$ pour $[\underline{s}_j, \overline{s}_j]$.

7.4 Notion de cohérence et de flexibilité

Nous savons que chaque centre gère un ordonnancement local selon la structure d'intervalles associée aux fenêtres de lancement et d'échéance des différentes opérations qu'il doit exécuter. Aussi, les différents centres dépendent les uns des autres. Lors de la coopération, les conversations engagées entre CDDs doivent aboutir à des propositions (intervalles contractés) devant être cohérentes du point de vue local et global, pour une éventuelle prise de décision. Elles doivent aussi permettre d'assurer une organisation flexible afin que les centres de décision puissent anticiper les imprévus. La suite de cette section est consacrée à la définition des notions de cohérence et de flexibilité.

7.4.1 Cohérence locale

Sous les hypothèses définies dans la partie précédente, un CDD doit définir un ordonnancement local qui soit cohérent avec les fenêtres $[r_{uv}]$ de disponibilité (ou lancement) et les fenêtres $[d_{uv}]$ de livraison contractées entre les différents acteurs.

La cohérence de l'intervalle $[s_{uv}] = [lst^w_{uv}, est^w_{uv}]$ notée aussi $[\underline{s}_{uv}, \overline{s}_{uv}]$ des dates de début de réalisation (resp. $[f_{uv}]$ des date de fin de réalisation) avec les intervalles de lancement

$[r_{uv}]$ (resp. d'échéances $[d_{uv}]$) négociés, peut s'exprimer alors par les inégalités suivantes :

$$\begin{cases} r_{uv}^{min} \leq \underline{s}_{uv} \leq r_{uv}^{max} \\ \\ d_{uv}^{min} \leq \overline{s}_{uv} + p_{uv} = \overline{f}_{uv} \leq d_{uv}^{max} \end{cases} \tag{7.2}$$

Sachant qu'une opération ne peut jamais commencer son exécution avant sa date de disponibilité, le système d'inégalités (7.2) impose, d'une part, qu'un CDD ne planifie jamais le commencement "au pire et au plus tard" d'une tâche sur un produit avant que celui-ci ne soit disponible (i.e. $r_{uv}^{min} \leq \underline{s}_{uv}$) et, d'autre part, le produit soit délivré "au pire et au plus tôt" à temps (i.e. $\overline{s}_{uv} + p_{uv} \leq d_{uv}^{max}$). Les deux autres inégalités permettent d'éviter la situation de sur-autonomie où un CDD proposerait à un CDD amont d'achever au pire des cas le produit plus tôt que nécessaire (i.e. $r_{uv}^{max} < \underline{s}_{uv}$) et celle où un CDD proposerait au CDD aval de lui livrer un produit, au pire des cas, plus tôt que nécessaire (i.e. $\overline{s}_{uv} + p_{uv} < d_{uv}^{min}$).

7.4.2 Cohérence globale

Du point de vue d'un centre de décision, et comme mentionné plus haut, la contrainte de cohérence locale donnée par le système d'inégalités (7.2) permet d'assurer que les ordonnancements locaux possibles soient cohérents avec les fenêtres de disponibilité et de livraison contractés entre les différents acteurs.

Si on se place maintenant du point de vue de tout le réseau de centres de décision, les ordonnancements locaux doivent être définis de manière à ce qu'ils soient cohérents entre eux. Signalons que l'interdépendance entre les ordonnancements locaux vient du fait que deux opérations d'un même travail réalisées par deux centres de décision différents sont liées par une contrainte de précédence (au sens de la gamme opératoire). La contrainte de cohérence globale peut alors s'exprimer par les inégalités suivantes :

$$\begin{cases} \underline{s}_{uv} \geq \underline{s}_{u,v-1} + p_{u,v-1} = \underline{f}_{u,v-1} \\ \\ \overline{s}_{uv} \geq \overline{s}_{u,v-1} + p_{u,v-1} = \overline{f}_{u,v-1} \end{cases} \tag{7.3}$$

Le système d'inégalité (7.3) impose qu'une opération ne peut commencer son exécution au pire et au plus tôt (resp. au plus tard) avant la date de fin au pire et au plus tôt (resp. au plus tard) de l'opération qui la précède selon la gamme opératoire.

7.4.3 Risque d'incohérence

Nous avons montré à travers le système d'inégalité (7.3) comment assurer la cohérence entre les dates au pire et au plus tôt (resp. au plus tard) de deux opérations consécutives

d'un même travail. Toutefois, la situation où la date de début d'une opération serait inférieure à la date de fin de l'opération qui la précède, i.e., $\underline{s}_{uv} \leq \overline{f}_{u,v-1}$ ou bien $\overline{f}_{uv} \geq \underline{s}_{u,v+1}$ est envisageable. La comparaison de l'intervalle $[s_{uv}]$ avec $[f_{u,v-1}]$ et l'intervalle $[f_{uv}]$ avec $[s_{u,v+1}]$ fait alors intervenir la notion de risque d'incohérence.

La Figure 7.3, représente le cas idéal où le risque d'incohérence est nul (les intervalles $[f_{u,v-1}]$ et $[s_{uv}]$ (resp. $[f_{uv}]$ et $[s_{u,v+1}]$) ne se chevauchent pas. En effet, même dans le cas où le produit serait délivré "le plus tard" (au pire et au plus tôt) possible pour un CDD, cela resterait compatible avec la date de début "au plus tôt" (au pire au plus tôt) de la prochaine opération sur le CDD suivant.

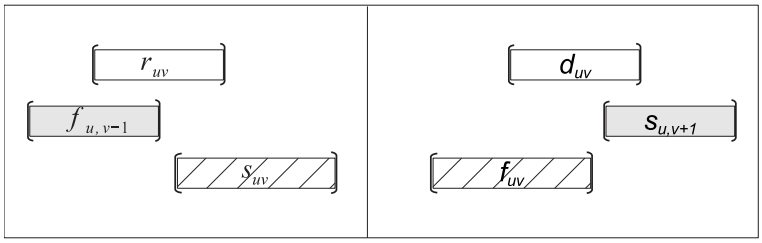

FIG. 7.3: Risque d'incoherence nul

En général, on peut aussi admettre que les intervalles $[s_{uv}]$ et $[f_{u,v-1}]$ ou bien $[f_{uv}]$ et $[s_{u,v+1}]$ se chevauchent (cf. Figure 7.4). Dans ce cas, plus le chevauchement est grand et plus le risque qu'un CDD amont livre un produit trop tard (étant donnée la date de début de la prochaine opération) est important.

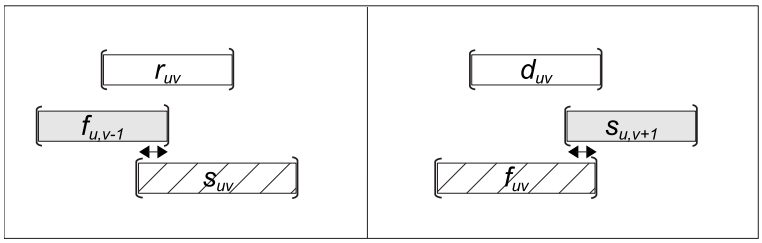

FIG. 7.4: Risque potentiel d'incoherence

En considérant que les contraintes de cohérence locale sont respectées lors de la négociation des intervalles entre les CDDs, on peut facilement remarquer que le risque d'incohérence est étroitement lié à la largeur de l'intervalle $[r_{uv}]$ (resp. $[d_{uv}]$) négocié. En effet, plus l'intervalle est grand, et plus le centre réalisant o_{uv} fait confiance au centre amont réalisant $o_{u,v-1}$, mais plus le risque d'incohérence peut être grand. Inversement, plus l'intervalle $[r_{uv}]$ est petit, plus le risque d'incohérence est petit, mais plus le centre réalisant o_{uv} doit se protéger contre une mauvaise performance du centre amont réalisant $o_{u,v-1}$. Le désir d'aboutir à une solution harmonieuse peut alors être guidé par le besoin de minimiser le risque d'incohérence. Nous reviendrons sur cette notion dans le chapitre 8.

7.4.4 Flexibilité

Nous avons mentionné dans cette étude que, pour la gestion de l'ordonnancement local d'un centre, ce dernier dispose d'une autonomie décisionnelle, i.e. d'une flexibilité décisionnelle, pour résister aux perturbations éventuelles et anticiper l'arrivée de nouvelles opérations. En effet, plus l'organisation est flexible et plus le centre a le temps, en dépit des perturbations, de réaliser les opérations selon ses engagements. Dans notre cas, cette flexibilité décisionnelle peut être reliée d'une part à la flexibilité temporelle propre au centre. Cette flexibilité est alors directement liée au temps libre dont dispose le CDD sur l'horizon de l'ordonnancement. Ainsi, plus les marges de réalisation des opérations sont larges, plus l'organisation du centre est sûre, et plus le CDD est confiant. Inversement, plus les marges de réalisation se resserrent et plus l'organisation du centre est fragile. Cette flexibilité temporelle peut par exemple être exprimée par :

$$Flex_i = \max_{o_{uv} \in V_i} d_{uv} - \min_{o_{uv} \in V_i} r_{uv} - \sum_{o_{uv} \in V_i} p_{uv} \tag{7.4}$$

D'un autre côté, la flexibilité décisionnelle d'un CDD peut aussi être reliée à la flexibilité séquentielle qu'offre la structure pyramidale caractérisant les données des opérations gérées par le centre. Cette flexibilité séquentielle est perçue à travers le nombre de séquences dominantes caractérisées au niveau du centre par application du Théorème des pyramides (cf. Chapitre 6). En effet, il est possible de dénombrer facilement les séquences dominantes caractérisées par le théorème, pour caractériser la flexibilité séquentielle du CDD, grâce à la formule :

$$Flex'_i = \Pi_{q=1}^{N} (q+1)^{n_q} \tag{7.5}$$

où n_q désigne le nombre d'opérations non sommets appartenant exactement à q pyramides et N, le nombre total de pyramides dans la structure d'intervalles caractérisant le CDD_i.

7.5 Conclusion

Nous avons, dans ce chapitre, défini la problématique d'ordonnancement coopératif pour la gestion d'un atelier en présence de perturbations. En supposant que chaque ressource est assimilée à un centre de décision possédant sa propre autonomie décisionnelle, nous considérons que la fonction ordonnancement est distribuée entre les différents acteurs et, qu'elle s'adapte de façon réactive à l'évolution de l'atelier (arrivée dynamique des commandes). Les décisions d'ordonnancement sont alors progressivement négociées entre les acteurs. Nous avons ensuite défini les mécanismes de coopération entre les centres pour la négociation des décisions d'ordonnancement. Ainsi, les conversations engagées pour la coopération ont pour objectif d'amener les acteurs à expliciter les marges de sécurité que

chacun se réserve pour dimensionner l'intervalle de livraison/consommation d'un produit négocié. L'intervalle de décision représente un engagement de mise à disposition du produit par un centre auprès de son client (centre aval) dans la fenêtre temporelle délimitée par l'intervalle. Notons, que cet intervalle est défini localement en se basant sur la méthode d'ordonnancement robuste à une machine. Afin d'assurer la pertinence des décisions d'ordonnancement du point de vue local et global, nous avons montré sous forme de contraintes que les intervalles contractés doivent être cohérents du point de vue local et global. Aussi, dans la perspective de satisfaire les objectifs internes des centres, une notion de risque d'incohérence liée à l'insensibilité des décisions d'ordonnancement aux perturbations touchant le CDD, ainsi qu'une notion de flexibilité caractérisant l'organisation du CDD, sont introduites.

Dans le chapitre qui suit nous proposons une formalisation de la problématique de négociation et mettons au point un algorithme distribué pour la coopération entre les centres de décision.

Chapitre 8

Une méthode pour l'ordonnancement coopératif et robuste

Ce chapitre décrit l'approche de résolution adoptée pour la résolution du problème d'ordonnancement sous incertitudes. La solution est construite de manière distribuée par coopération entre les acteurs. Les fonctions inhérentes à la coopération sont la négociation, la coordination et la renégociation. Un algorithme distribué et dynamique de résolution est proposé permettant la construction des décisions d'ordonnancement : des décisions évoluant au fur et à mesure des changements d'état du système de production.

8.1 Formalisation d'un processus de coopération inter-machines

Comme dans [Despontin-Monsarrat 04], nous considérons que les différentes fonctions inhérentes à la coopération sont la négociation, la coordination et la renégociation. L'objectif de la négociation est d'amener un couple d'acteurs à s'entendre pour la première fois sur un intervalle de livraison/consommation à contracter. Le processus de coordination doit permettre aux ressources de se synchroniser pour respecter les intervalles contractés. Le processus de renégociation a pour objectif quant à lui de rendre les décisions cohérentes, si elles ne le sont plus.

8.1.1 Processus de négociation

Nous distinguons deux modes de négociation : la négociation amont et la négociation aval. Un processus de négociation est initié lorsqu'un centre de décision émet une proposition. Cette proposition peut émaner d'un CDD demandant à un autre CDD amont de réaliser une opération sur un produit, correspondant à l'opération précédente de la gamme (négociation amont), ou suggérant à un CDD aval de réaliser une opération sur un produit,

correspondant à l'opération suivante de la gamme (négociation aval). De tels processus sont amorcés au moment de l'arrivée d'un nouveau travail. Il s'agit de déterminer des intervalles de consommation $[r_{uv}]$ et des intervalles de livraison $[d_{uv}]$ pour chaque opération o_{uv} du nouveau travail u. Nous supposons qu'un intervalle de délai $[d_u]$ est associé à la commande (cet intervalle pouvant éventuellement être réduit à un point). Le but des différentes négociations est de définir les intervalles $[r_{uv}]$ et $[d_{uv}]$ des opérations du nouveau produit, et éventuellement de nouveaux délais de livraison/ consommation des produits déjà négociés, de sorte à respecter les contraintes de cohérence locales (7.2) et globales (7.3) données dans le chapitre précédent, d'optimiser le risque d'incohérence ainsi que la flexibilité de chaque CDD et de satisfaire dans la mesure du possible l'intervalle de délai $[d_u]$.

Un processus de négociation correspond à une conversation. Il est réalisé par échange de requêtes entre paire de CDDs. Le centre émettant une proposition d'intervalle $[r_{uv}]$ ou $[d_{uv}]$ est l'initiateur du processus de coopération. Le destinataire de la proposition répond à la proposition. Il peut aussi émettre une contre proposition. Une conversation correspond donc à une suite de propositions et contre-propositions réalisée par échange de requêtes.

8.1.2 Processus de coordination

La phase de coordination peut avoir lieu dès qu'un engagement ferme de consommation et production est contracté entre un couple de centres de décision. Nous savons que les ordonnancements locaux des différents CDDs évoluent dans le temps. En effet, l'arrivée de nouveaux travaux ou l'occurrence d'aléas, impose à chaque CDD de mettre à jour l'ensemble des séquences dominantes qu'il gère. Il est donc nécessaire que les CDDs se coordonnent au fur et à mesure qu'ils évoluent en s'échangeant des informations sur les valeurs des dates de début et de fin au pire de chaque opération, afin de se synchroniser.

Ainsi que l'illustre la Figure 8.1, la modification d'une date de fin d'une opération (resp. de début) au niveau d'un CDD, e.g. $\underline{f}_{u,v-1} = lft^w_{u,v-1}$ (resp. $\overline{s}_{u,v+1} = est^w_{u,v+1}$), peut induire un réordonnancement sur le CDD réalisant l'opération suivante (resp. précédente) en augmentant (resp. diminuant) $\underline{s}_{uv} = lst^w_{uv}$ (resp. $\overline{f}_{uv} = eft^w_{uv}$) de manière à avoir $\underline{s}_{uv} \geq \underline{f}_{u,v-1}$ (resp. $\overline{f}_{uv} \leq \overline{s}_{u,v+1}$), c'est-à-dire, l'adaptation de l'ensemble dominant de séquences. Notons que la coordination pour la synchronisation des dates revient à respecter les contraintes de cohérence globale définies dans la section 7.4.2.

Des informations entre CDDs peuvent être échangées, de manière périodique ou asynchrone, à chaque fois que des dates de début ou de fin sont modifiées. Cet échange d'information permet aux différents CDDs d'avoir une vision commune et réaliste des données, et de mettre à jour l'ensemble de séquences dominantes qu'ils gèrent.

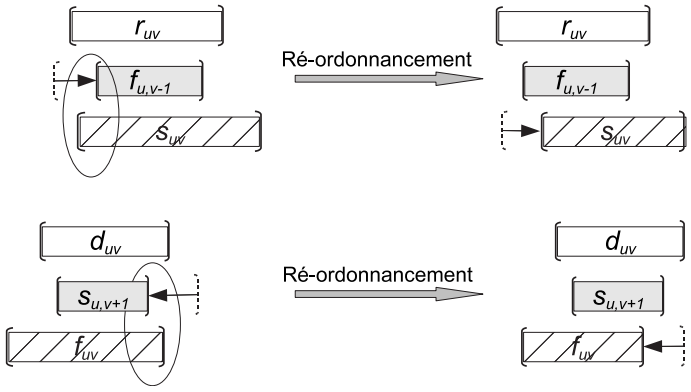

FIG. 8.1: Situations nécessitant un réordonnancement

8.1.3 Processus de renégociation

Un processus de renégociation est initié lorsqu'un des centres de décision souhaite modifier un intervalle de livaison/consommation contracté. En effet, au cours des négociations menées lors de l'insertion de nouvelles opérations, renégocier certains intervalles $[r_i]$ ou $[d_i]$ déjà contractés pour des opérations existantes peut s'avérer pertinent pour améliorer la performance globale. On parle alors de renégociation. Un processus de renégociation se distingue d'un processus de négociation par le fait que la première proposition se réfère à un intervalle déjà existant.

Notons qu'une renégociation peut également avoir lieu si la modification d'un intervalle $[s_{uv}]$ ou $[f_{uv}]$ d'un travail, suite à un aléa surgissant sur un CDD, rend l'intervalle localement incohérent avec les valeurs courantes $[r_{uv}]$ et $[d_{uv}]$. Le processus de renégociation a alors pour but de recouvrer la cohérence.

Signalons que le cas de la Figure 8.1 donnée dans la section précédente, illustre des modifications des dates des opérations d'un centre, induisant un ré-ordonnancement au niveau d'un autre centre (amont ou aval) et ce dans le but d'assurer de nouveau le respect de la contrainte de cohérence globale. Dans ce cas, les modifications sont cohérentes avec les intervalles contractés et aucune renégociation n'est alors nécessaire. C'est dans le cas où le ré-ordonnancement entraîne une violation des contraintes de cohérence locale données par le système d'inéquation (7.2), que la renégociation des intervalles contractés s'impose.

Les cas où une violation de la contrainte locale entraîne une renégociation des intervalles sont illustrés sur la Figure 8.2 : ce sont les cas où un CDD demanderait, à un CDD amont d'achever une opération plus tôt que prévu ou, à un CDD aval de lui livrer le produit plus tard que prévu, ainsi que les situations de sur-autonomie où un CDD pouvant, soit commencer l'exécution d'une opération plus tard, et renégocie ainsi avec l'amont la possibilité que l'opération lui soit livré plus tard ou bien, livrer une opération plus tôt que

prévu et renégocie avec l'aval la possibilité qu'il livre l'opération plus tôt que prévu.

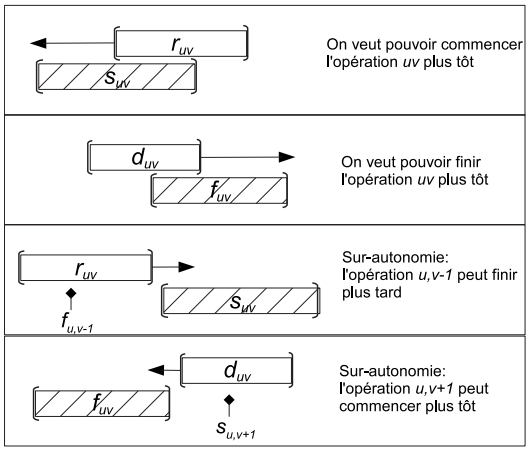

FIG. 8.2: Situations de renégotiation

Comme pour le processus de négociation, l'initiateur du processus de renégociation émet des propositions, le destinataire de la proposition émettant des contre propositions. La renégociation correspond, tout comme la négociation, à une conversation par échange de requêtes.

Émettre des propositions et des contre-propositions cohérentes les unes avec les autres, lors des processus de négociation et renégociation, n'est pas trivial. Dans la suite, nous développons cet aspect et proposons un algorithme distribué pour la négociation/renégociation des intervalles à contracter, et mettons au point des modèles de programmation permettant de fixer, lors de la coopération entre CDDs, les décisions d'ordonnancement.

La section suivante est consacrée à la description comportementale d'un CDD vis-à-vis de ses fournisseurs amont et de ses clients aval pour la prise des décisions d'ordonnancement. Ce comportement est formalisé par un algorithme.

8.2 Comportement d'un centre de décision

Nous avons vu qu'un centre de décision a pour objectif de déterminer et proposer, à l'occurrence d'un événement, des intervalles de consommation ou de disponibilité aux centres amont, et des intervalles de livraison ou d'échéance aux centres aval, pour les opérations qu'il doit réaliser et pour lesquelles il est lié (au sens de la gamme) avec les autres centres. Comme nous le verrons dans la Section 8.3, ces intervalles $[r_i]$ et $[d_i]$ sont déterminés respectivement par utilisation d'un modèle amont et d'un modèle aval que nous proposons.

L'idée de base du fonctionnement comportemental d'un centre de décision est le suivant (voir Figure 8.3) : à l'occurrence d'un événement, e.g. aléa, nouvelle opération suite à l'arrivée d'une nouvelle commande dans le système ou une nouvelle requête émanant d'un autre CDD, etc., un centre CDD_u détermine, à son niveau, les dates de début des opérations à exécuter. Il identifie ensuite les opérations incohérentes i.e., celles dont les contraintes de cohérence locales sont violées. Puis pour chacune d'elles, il négocie avec le centre amont la possibilité qu'il soit livré plus tôt en lui proposant une nouvelle fenêtre de livraison. Le centre amont émet alors, après traitement de la requête, une réponse précisant le meilleur intervalle de livraison qu'il est capable d'assurer, étant donné ses objectifs propres. Remarquons que le centre sollicité, peut être amené, avant de répondre, de renégocier à son tour avec ses centre amont des intervalles de livraison d'autres opérations. Dans la situation où le centre CDD_u ne peut négocier davantage avec les centres amont l'avancement des dates des travaux incohérents, il détermine à son niveau les dates de fin des opérations qu'il doit exécuter. De la même manière, nous supposons que le CDD_u a fait de son mieux, et que les nouvelles opérations incohérentes (livrées plus tard que prévu dans le contrat) ne feront alors pas objet d'une négociation avec le centre aval mais d'une contractualisation. Les décisions d'ordonnancement devenant fermes, le centre aval accepte alors et s'engage à prendre en compte le nouveau intervalle de livraison même si les dates ont été retardées.

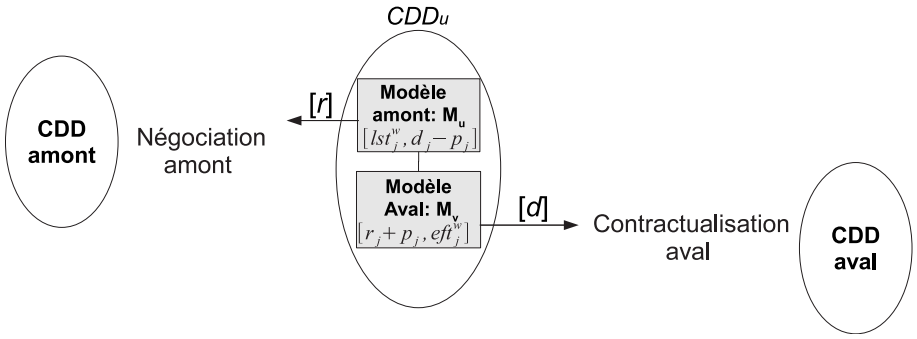

FIG. 8.3: Comportement d'un centre de décision

8.2.1 Négocier amont/ Contracter aval

Nous nous focalisons sur un centre de décision CDD_u, et nous distinguons les mécanismes mis en œuvre lors des processus de (re)négociation, permettant de déterminer les intervalles de consommation/livraison.

Un processus de négociation est initié lorsqu'un centre de décision (ou client) demande à un autre centre amont, la prise en charge d'une nouvelle opération sur un produit correspondant à l'opération précédente au sens de la gamme, en spécifiant un intervalle de livraison souhaité, ou bien de revoir un délai de livraison d'une opération déjà contracté avec le centre amont. Dans le premier cas, on négocie un délai souhaité avec le centre amont

alors que dans le second cas, on renégocie l'intervalle de livraison déjà contracté avec le centre. La prise en charge d'une nouvelle commande pour un produit donné à laquelle est associée un délai de livraison est initialement sollicitée par un acteur client.

Le centre CDD_u ayant reçu la requête prend en charge la demande. Qu'il s'agisse d'une négociation ou d'une renégociation, CDD_u définit selon son organisation interne (à travers un modèle amont qu'on note M_u et décrit dans la Section), tout en respectant ses engagements avec les centres aval et le centre initiateur de la requête, les nouveaux délais de livraison pour ses centres amont, soit $[d_{j-1}] = [r_j] = [lst_j^w, d_j - p_j]$, et ce pour toute opération j gérée par le centre en question (nous reviendrons en détail sur les définitions de cet intervalle et du modèle amont dans la Section 8.3). Ces délais peuvent alors être incohérents avec ceux contractés avec les centres amont (contrainte de cohérence locale violée, voir Figure 8.4). Dans cette situation, le centre voulant être livré plus tôt, négocie de nouveau à son tour avec les centres amont les nouveaux intervalles incohérents en faisant des propositions de renégociations par émission de nouvelles requêtes, et le processus se propage en arrière, et dans la mesure du possible, tant qu'un centre déduit un délai de livraison incohérent avec celui contracté.

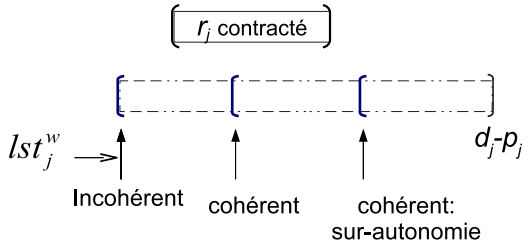

FIG. 8.4: Cohérence et incohérence de $[s_j]$ avec $[r_j]$

Notons que lors d'une prise en charge d'une requête par un centre, tout un processus de renégociation entre CDDs peut être amorcé. Aussi, lorsqu'un centre CDD_u reçoit la réponse à la requête qu'il a émise à un centre CDD_v, nous supposons que CDD_v a fait de son mieux et que l'intervalle de livraison proposé pour CDD_u devient ferme pour ce dernier. CDD_u n'a donc plus la possibilité de rénégocier cet intervalle, du moins tant que son état interne n'aura pas évolué (aucun aléa n'est venu perturber son organisation). Les nouveaux intervalles de livraisons sont alors imposés, i.e., la contrainte de cohérence locale doit obligatoirement être respectée dans le futur et pour les réponses que l'on a reçues. Ceci sera alors traduit par le respect d'une contrainte temporelle donnée par la formule 8.1 suivante :

$$lst_x^w \geq r_x^{min} \quad \forall x \in X \tag{8.1}$$

X étant l'ensemble des travaux pour lesquels des requêtes ont été formulées.

Cette précaution est nécessaire pour éviter une situation instable où le CDD referait, au cours d'une autre négociation, une requête pour la même opération, avec un intervalle de consommation différent, alors qu'il n'a pas encore reçu la réponse à sa requête précédente. Nous imposons donc le respect de certains intervalles de consommation (celles pour lesquelles des requêtes sont en cours de traitement), et autorisons la violation des intervalles des autres opérations.

Lorsqu'un centre reçoit toutes les réponses à ses sollicitations, il vérifie la cohérence de son organisation interne. De nouveaux délais de livraison pour les centres amont correspondant aux délais de consommation pour le centre en question sont détérminés. Dans la situation où la cohérence locale est violée pour les opérations autres que celles pour lesquelles des requêtes ont déjà été émises, de nouvelles requêtes sont envoyées aux CDDs amont. Par contre, si les intervalles calculés sont cohérents avec ceux contractés, ou bien s'il est impossible de trouver une solution satisfaisante, la renégociation avec les centres amont s'achève, et le centre calcule, étant données les nouvelles informations, les nouveaux délais de livraison des opérations qu'il a en charge.

Ainsi, tout en respectant les engagements avec les centres amont, le CDD détermine selon son organisation interne (via le modèle aval qu'on note M_D) les nouveaux délais de consommation des opérations sur les produits correspondants, au sens de la gamme, aux opérations successeurs pour les centres aval, soit : $[r_j + p_j, eft_j^w] = [d_j] = [r_{j+1}]$ pour toute opération j exécutée par le centre en question (nous reviendrons aussi en détail sur la définition de cet intervalle et le modèle aval dans la Section 8.3). Notons que ces délais peuvent être incohérents avec ceux contractés avec le centre aval (une contrainte de cohérence locale est violée ; cf. Figure 8.5). Dans cette situation, le CDD livrera forcément plus tard que convenu le centre aval. De ce fait, nous considérons que la coopération avec le centre aval ne fera pas objet d'une négociation, mais d'une contractualisation : le CDD aval devant obligatoirement intégrer le nouveau délai de consommation, ce dernier étant ferme.

En dehors de ces phases de négociation et renégociation, on suppose que les CDDs se coordonnent en se communiquant les valeurs des dates de disponibilité et d'échéance de leurs opération, eft_j^w et lst_j^w, que chacun se fixe. Ces échanges peuvent se faire périodiquement ou de façon asynchrone, à l'occurrence de modifications. Lorsqu'un centre reçoit une telle information, il doit vérifier que son organisation demeure cohérente.

8.2.2 Proposition d'un algorithme pour la coopération inter-CDDs

La formalisation de la coopération entre CDDs passe par l'élaboration, de façon cohérente avec l'ensemble des informations disponibles, de propositions entre acteurs pour la prise de décision d'ordonnancement. Les mécanismes de coopération ayant été décrits,

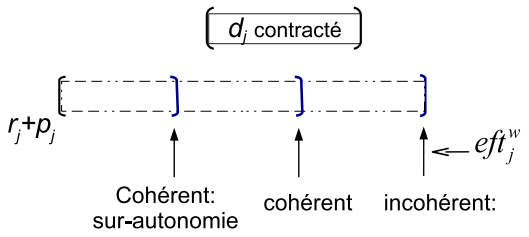

FIG. 8.5: Cohérence et incohérence de $[f_j]$ avec $[d_j]$

nous proposons, dans ce qui suit, un algorithme dynamique régissant le comportement des CDDs et, décrivant plus formellement les processus de négociation et renégociation entre les centres.

Nous considérons lors du processus de négociation/renégociation que les CDDs communiquent pour la prise de décision sur le dimensionnement des intervalles de consommation/livraison. Les conversations entres CDDs sont constituées de messages. Chaque type de message identifie une action particulière de l'émetteur auprès du destinataire.

Les messages échangés entre les centres de décisions sont de type (cf. Figure 8.6) :

- **Nouveau job (N)** *New* : Un client demande la réalisation d'un nouveau produit en spécifiant un délai de livraison souhaité ;
- **Requête (R)** *Request* : Un centre initiateur (émetteur du message) formule une requête au centre amont en lui proposant un intervalle de livraison donné (cible), correspondant à une demande d'autorisation pour commencer une opération déjà existante plus tôt que prévu ;
- **Réponse (A)** *Answer* : Le centre répond à la requête R du centre aval en indiquant un intervalle de livraison correspondant à ce qu'il peut faire de mieux. La réponse, pouvant être cohérente ou pas avec la requête reçue, est ferme et n'est plus négociable ;
- **Information (I)** : Si après traitement d'une requête par un CDD, d'autres opérations autres que celle ayant fait l'objet de la requête, voient leur délai changer sans devenir toutefois incohérents avec l'intervalle contracté, le centre envoie une information *(I)* sur le nouveau délai aux centres amont ou aval ;
- **Retard (D)** *Delay* : Dans le cas où les délais de livraison sont retardés, le centre envoie une information D au centre aval pour indiquer la valeur du retard de livraison.

Un processus de négociation ou renégociation peut être déclenché lors de la réception, par un centre CDD_u, d'un des messages cités précédemment hormis le message de type I, où une renégociation a lieu qu'en cas d'incohérence. Cette notion de processus de négociation est importante, et chacun des messages est lié à un processus distinct. En effet, le CDD peut devoir lancer un processus de négociation s'il reçoit, du centre initiateur, le message

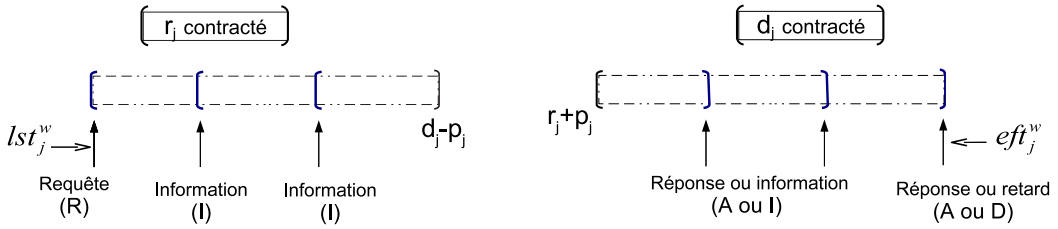

FIG. 8.6: Types de messages

N, R, A ou D. Dans la situation où il reçoit le message I, cela permet la coordination entre les centres, et le centre doit lui même initier le processus pour prendre en compte les nouvelles informations et opérer des changements à son niveau (si l'information reçue rend son état incohérent). Ainsi, nous notons par p, la variable qui définit un processus de négociation. Les attributs de cette variable ainsi que ceux de R, A, N, D et I sont donnés sur la Figure 8.7 suivante. Nous pouvons ainsi lire les attributs caractérisant un processus de négociation qui sont : son identifiant id, l'identifiant du centre source s l'ayant initié ainsi que l'opération j faisant objet de la négociation, et la flexibilité $Flex$ du centre initiateur.

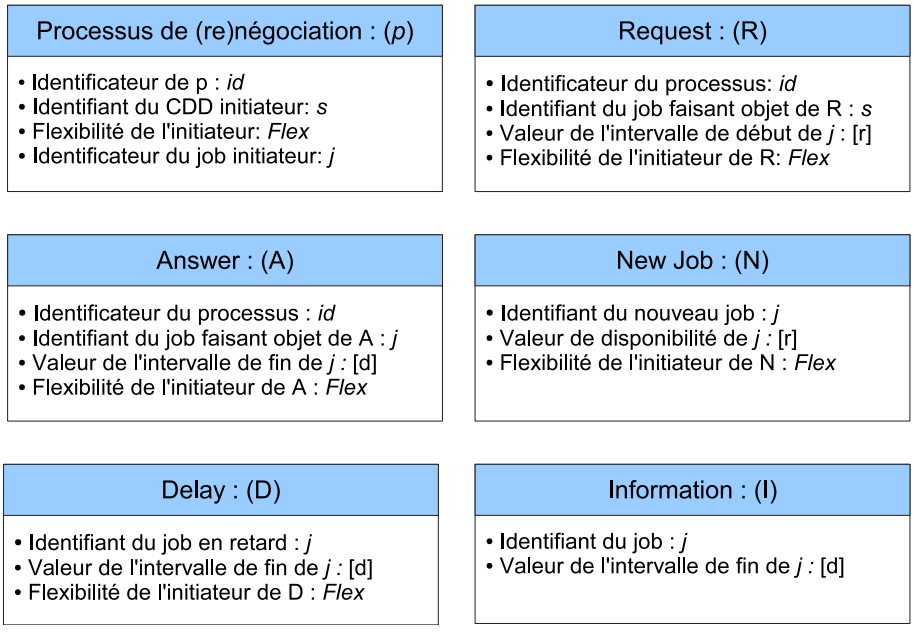

FIG. 8.7: Attributs des variables

Un centre de décision peut donc recevoir une proposition pour la réalisation d'une nouvelle opération (N) ou pour la modification d'un intervalle de livraison déjà contracté (R), tout comme il doit envoyer, suite à la prise en charge d'une des requêtes (N) ou

(R), la réponse (A) au CDD initiateur, ou informer dans le cas où des modifications sur des délais de livraison ont été opérées à son niveau, en envoyant un message D en cas de retard (ou le message I si le délai demeure cohérent) aux centres concernés. (cf. Figure 8.8).

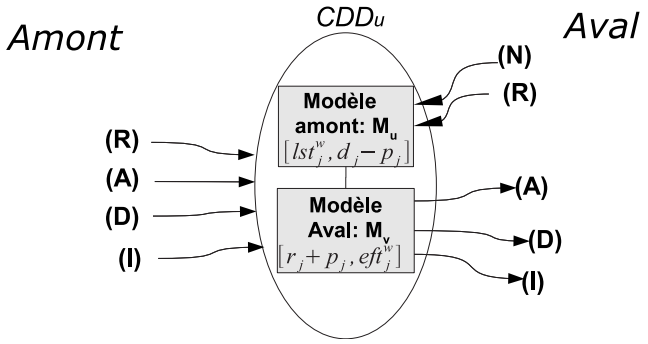

FIG. 8.8: Les différents messages reçus ou émis par un CDD

En notant j une opération, nous désignons par $(j+1)$, l'opération successeur de j et par $(j-1)$, l'opération prédécesseur de j.

Tout CDD de l'atelier est dynamique et évolue au fur et à mesure des négociations engagées dans la prise des décisions d'ordonnancement. Des messages transitent d'un centre vers d'autres centres amont ou aval. Ces messages, une fois reçus par un centre de décision, permettent de l'activer. Le comportement d'un CDD est décrit en détail par l'Algorithme 1.

Pour ne pas surcharger l'algorithme, nous avons volontairement ici négligé la gestion des messages purement informatifs (de type I) qui permettent la coordination des agents. Ces messages sont émis par un centre soit de façon périodique, soit de façon asynchrone, lorsque, lors des négociations, les valeurs des $[d_j]$ ou des $[r_j]$ des opérations qu'il prend en charge sont modifiées, sans que cela entraîne une incohérence (i.e., $d_j^{max} \leq eft_j^w = \overline{f}_j$ et $r_j^{min} \geq lst_{j-1}^w = \underline{f}_{j-1}$). À la réception d'un message de type I, un agent doit vérifier que son état reste cohérent (utilisation de la procédure BackwardSolve) et si, tel n'est pas le cas, engager une renégociation des intervalles caractéristiques de ses opérations. Aussi, la gestion des messages de type N, correspondant à une requête pour une nouvelle opération, est équivalente à la gestion des messages de type R, et n'est pas considérée dans l'algorithme.

Si un nouveau message est reçu (ligne 3), deux cas sont distingués selon si le message est de type R ou D (ligne 4) ou alors de type A (ligne 13). Dans le premier cas les lignes 4-12 de l'algorithme décrivent les traitements faits. Tout d'abord, la variable $Check$ est mise à vrai car l'agent va devoir vérifier la cohérence de son état. Un nouveau processus de négociation p est ensuite créé. Pour un message de type R, on considère que l'initiateur $p.s$ du processus est l'agent ayant émis la requête et on mémorise dans $p.j$ la tâcheopération

Algorithme 1 Comportement d'un centre de décision CDD_u

```
1: repeat
2:   msg ← Getmsg() ;
3:   if msg ≠ Null then
4:     if msg.type = R ∨ msg.type = D then
5:       Check ← true ;
6:       p ← CreateProcess(u) ;
7:       if msg.type = R then
8:         p.s ← msg.s ;
9:         p.j ← msg.j ;
10:      else
11:        p.s ← p.id ;
12:      end if
13:    else if msg.type = A then
14:      p ← SearchProcess(msg.s) ;
15:      AddAnswer(p, msg) ;
16:      UpdateConstraint((j + 1), [r]) ;
17:      Check ← AnswerEnd(p) ;
18:    end if
19:    if Check then
20:      BackwardSolve(M_u, p, state, U) ;
21:      if state ≠ Infeasible ∧ U ≠ ∅ then
22:        for all j ∈ U do
23:          Send(R, p.id, p.s, (j), [r]) ;
24:          AddRequest(p.id, j, [r]) ;
25:          AddExtraConstraint(j, [r]) ;
26:        end for
27:      else
28:        RemoveExtraConstraints(p.id) ;
29:        ForwardSolve(M_D, p, U) ;
30:        for all j ∈ V_u do
31:          if p.s ≠ p.id then
32:            Send(A, p.id, p.s, j, [d]) ;
33:          else if j ∈ U then
34:            UpdateConstraint(j, [d]) ;
35:            Send(D, p.id, p.s, u, j, [d]) ;
36:          end if
37:        end for
38:        RemoveProcess(p) ;
39:      end if
40:    end if
41:  end if
42: until False
```

$msg.j$ pour laquelle il faudra produire une réponse. Pour un message de type D, l'initiateur du processus est l'agent lui même, i.e., $p.s \leftarrow p.id$.

Dans le cas où le message reçu est de type A (lignes 13 à 18), il s'agit de la réponse à une requête que CDD_u avait précédemment émise dans le cadre d'un processus de négociation précédent. Dans ce cas, la procédure SearchProcess est appelée pour retrouver le processus correspondant parmi les processus actifs. On ajoute ensuite le message à la liste des réponses déjà reçues (procédure AddAnswer) par le processus. Dans le cas où toutes les réponses correspondant aux requêtes précédemment émises en amont ont été reçues (i.e., AnswerEnd$(p)= True$), CDD_u va de nouveau pouvoir vérifier la cohérence de son état et émettre si nécessaire de nouvelles requêtes.

Le déclenchement d'un processus de négociation amène le CDD_u à vérifier la cohérence de son organisation interne vis-à-vis des ses engagement passés. La vérification de la cohérence correspond aux lignes 19-40 de l'algorithme et se fait en deux phases : une phase de négociation amont et une phase de propagation aval. Lors de la première phase, la procédure BackwardSolve est appelée (son fonctionnement est tributaire d'un modèle amont noté M_u et décrit dans la Section 8.3). Cette procédure établit,dans la situation où la solution retournée est faisable, la liste $\mathcal{U}$ des opérations j pour lesquelles des requêtes de modification des intervalles de consommation $[r_j]$ des opérations $j \in \mathcal{U}$ doivent être envoyées aux agents amont. L'envoi des requêtes est réalisé aux lignes 22-26. Pour chaque envoi, on mémorise la requête dans la liste des requêtes en cours attachée au processus (appel à AddRequest) et on impose de façon temporaire que, dans les négociations futures, l'intervalle de consommation demandé à l'agent amont soit respecté i.e. les contraintes temporelles 8.1 soient respectées (appel à AddExtraConstraint). Cette précaution est nécessaire pour éviter une situation instable où l'agent referait, au cours d'une autre négociation, une requête pour la même opération, avec un intervalle de fin de réalisation différent, alors qu'il n'a pas encore reçu la réponse à sa requête précédente. La procédure BackwardSolve doit donc permettre de respecter les intervalles de fin de réalisation de certaines opérations amont (celles pour lesquelles des requêtes sont en cours de traitement) mais est libre de violer les intervalles des autres opérations.

La phase de propagation aval est amorcée lorsque la procédure BackwardSolve ne trouve pas de solution ($state = infeasible$) ou lorsque la liste $\mathcal{U}$ est vide i.e., les intervalles sont cohérents vis-à-vis des engagements contractés avec l'amont. Dans le premier cas, il n'est plus possible de continuer à négocier en amont en respectant les contraintes temporaires et les contraintes de positivité des variables de décision. Dans le second cas, il n'est plus nécessaire de négocier en amont puisque l'on respecte les intervalles de fin de réalisation de toutes les opérations amont. Dans les deux cas, la négociation avec l'amont s'achève et la coopération avec l'aval est engagée en appelant la procédure ForwardSolve (son fonc-

tionnement est tributaire d'un modèle aval M_D décrit dans la Section 8.3), après avoir levé les contraintes temporaires portant sur les intervalles de consommation $[r_j]$ imposées précédemment (appel à RemoveExtraConstraints). Celle-ci établit la liste $\mathcal{U}$ des opérations j pour lesquelles des retards sur les intervalles de fin de réalisation $[d_j]$ doivent être signalés auprès des agents aval. L'envoi des messages de retard est réalisé aux lignes 33-35 et on mémorise les valeurs des nouveaux intervalles. Si le processus p a été amorcé dans le cadre d'une requête d'un agent amont ($p.s \neq p.id$) concernant l'opération $p.j$, on envoie à cet agent la réponse à sa requête. S'il existe des opérations de $\mathcal{U} \subset V_u$, leurs délais de livraison étant retardés, une information est alors mise à jour (UpdateModelConstraint) et est signalée (D) auprès des centres aval réalisant les opérations successeurs ; Notons dans le cas où les délais de livraison ont été modifiés tout en demeurant cohérents, une information (I) est envoyée aux centres concernés. Enfin, une fois la propagation aval achevée, le processus p peut alors être détruit (ligne 38).

Notons que le comportement d'un centre autorise d'avoir plusieurs processus de négociation en cours. Il existe une cohérence dans la gestion, par les différents centres, des processus de négociation. En effet, étant donné que les valeurs des intervalles de début de réalisation indiquées dans les requêtes émises à l'amont sont provisoirement imposées pour les futures négociations, il est certain que les prochaines requêtes émises par le centre seront compatibles avec celles déjà émises. Cependant, les décisions seront dépendantes de l'ordre de traitement des requêtes reçues.

On peut également affirmer que le comportement des centres de décisions converge vers une solution globale d'ordonnancement. Comme nous l'avons mentionné au début de cette section, le principe de la coopération entre les acteurs consiste à avancer si possible les dates de début de réalisation de certaines opérations, et au cas contraire, de retarder les dates de fin de réalisation de certaines opérations. Ainsi, l'algorithme général proposé converge nécessairement vers une solution. En effet, les intervalles de de début de réalisation étant progressivement gelés lors de la négociation amont, on est sûr qu'à un moment donné l'état où ($state = Infeasible \vee \mathcal{U} = \emptyset$) sera atteint. La négociation avec l'amont s'achève et on passe dans une phase où les décisions sont imposées vers l'aval sans possibilité de recours.

Illustrons un exemple de processus de négociation à travers le diagramme de séquences donné par la Figure 8.9. Le centre initiateur CDD_z émet un intervalle cible de début de réalisation $[r_i]$ d'une opération i au centre amont CDD_u réalisant l'opération prédécesseur $i-1$. CDD_u ne pouvant localement absorber la modification de l'intervalle de fin de réalisation $[d_{i-1} = [r_i]]$, tout en conservant un niveau acceptable de flexibilité et de cohérence, identifie les opérations incohérentes $\mathcal{U} = \{i-1, j\}$ à travers la procédure *BackwardSolve*. Ceci induit alors, afin de répondre à la requête du CDD_z, une phase de négociation amont en émettant des requêtes aux centres CDD_x et CDD_y concernant respectivement les dates de fin de réalisation, des opérations $i-2$ et $j-1$. La phase de négociation amont étant

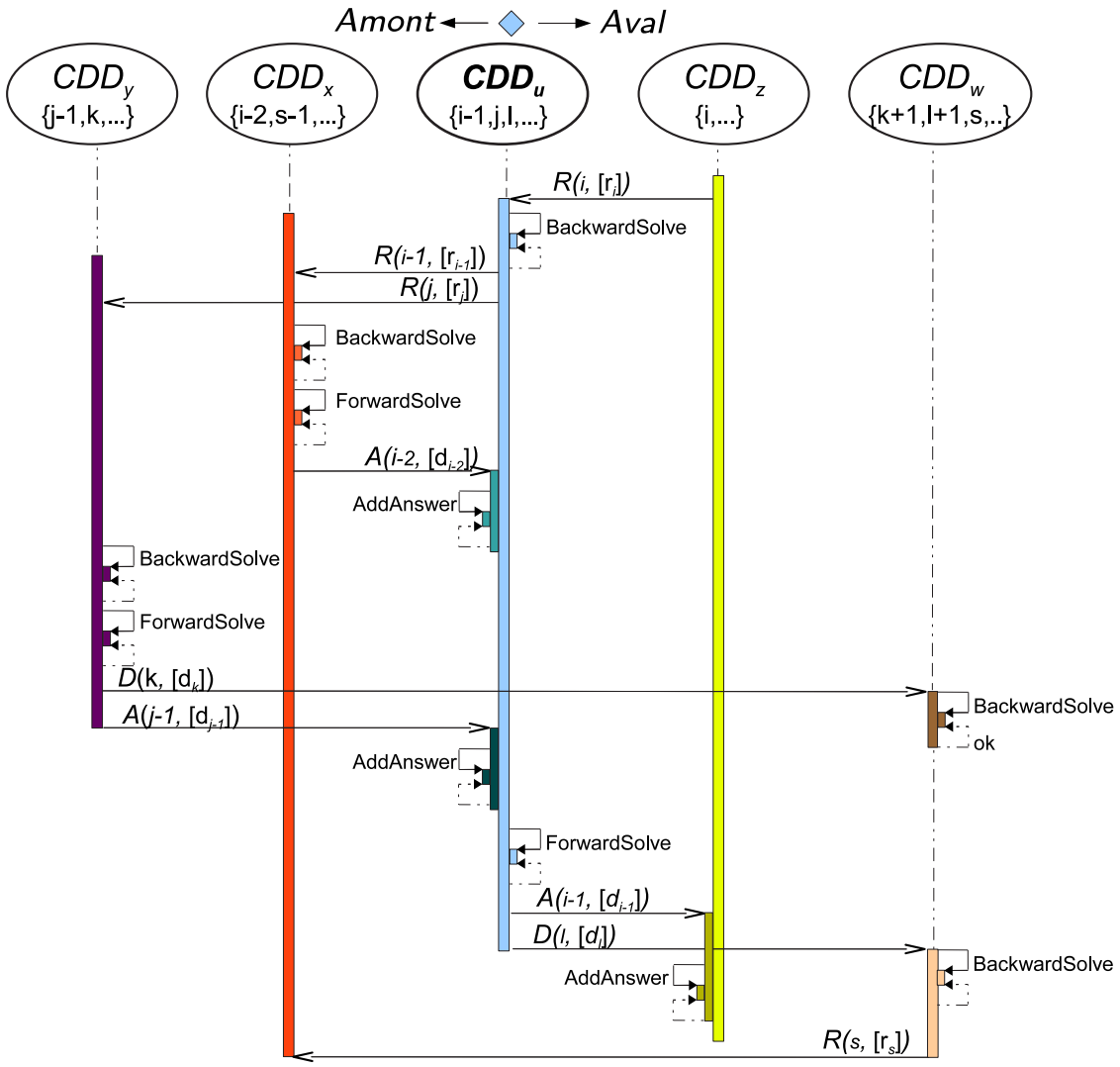

FIG. 8.9: Diagramme de séquences illustrant un processus de négociation

achevée, i.e., il n'est plus possible d'absorber par l'amont les modifications induites par l'émission de la toute première requête sans augmenter certains intervalles d'achèvement au pire des opérations existantes, une propagation aval des retards est alors engagée au fur et à mesure de la réception des éventuels messages A, D ou I. Comme on peut le lire sur la Figure 8.9, la phase de contractualisation aval commence lorsque le CDD_u reçoit toutes les réponses à ses requêtes, pour ensuite répondre via la procédure *ForwardSolve*, à la requête du centre CDD_z concernant l'opération i. Notons que cela contraint le CDD_u à retarder le délai d'une opération l à livrer au centre aval CDD_w. Cette même situation apparaît lorsque CDD_y, en réponse à la requête de CDD_u, retarde la fin de réalisation de l'opération k à livrer au CDD_w. Notons toutefois, que le centre CDD_w ayant reçu une information sur un retard de livraison concernant l'opération l, et ne pouvant plus assurer ses engagements de délais, initie à son tour une nouvelle phase de négociation amont avec le centre CDD_x concernant l'opération s. Signalons, que le CDD_x, traitant la requête concernant l'opération $s-1$, impose le respect du délai de livraison de l'opération $i-2$ contracté avec le CDD_u lors du processus de négociation.

8.3 Modèles de négociation/renégociation

Nous avons vu dans la section précédente, qu'à la réception par un CDD d'une proposition de négociation ou renégociation sur un intervalle cible, le centre doit répondre en fonction de son organisation interne et de manière à assurer au mieux la consistance des décisions déjà prises avec les autres CDDs auxquels il est lié. Ceci est réalisé grâce aux procédures *BackwardSolve* et *ForwardSolve*. Nous allons, dans cette section, décrire en détail les modèles qui régissent ces procédures.

Nous rappelons que l'ordonnancement local réalisé au niveau de chaque CDD est un ordonnancement robuste basé sur la caractérisation d'un ensemble de solutions possibles. Le Chapitre 6 a été consacré à la description de la technique d'ordonnancement robuste qui, étant donné un ordre relatif entre les dates de disponibilité et les dates d'échéance des opérations gérées par un CDD, fournit un ensemble de solutions dominantes dont les dates de début au pire et au mieux de chaque opération peuvent être calculées en temps polynomial.

Dans le cadre d'un atelier job-shop à ressources autonomes, la caractérisation d'un ensemble dominant de séquences nécessite de définir un ordre total entre les dates de disponibilité r_{uv} et d'échéances d_{uv} des opérations o_{uv} réalisées par chaque acteur. En effet, cet ordre est requis (i.e., la connaissance des valeurs de r_{uv} et d_{uv}) pour pouvoir appliquer le théorème des pyramides régissant la technique d'ordonnancement robuste et déduire les date de début des opérations. Dans la mesure où les négociations entre CDDs définissent des intervalles $[r_{uv}] = [r_{uv}^{min}\ r_{uv}^{max}]$ et $[d_{uv}] = [d_{uv}^{min}\ d_{uv}^{max}]$, les valeurs r_{uv} et d_{uv} qui doivent

être choisies dans ces intervalles ne sont pas fixées, et il existe donc plusieurs ordres totaux possibles. De plus, à chaque ordre total correspond un ensemble de séquences dominantes, i.e., un intervalle de date de début au mieux et au pire $[s_{uv}]$ différent, pour chaque opération o_{uv} appartenant à une séquence dominante donnée.

La détermination sur un CDD d'un ordre total pertinent entre les dates de disponibilité r_{uv} et d'échéance d_{uv} des opérations doit alors être guidée par l'objectif de respecter les contraintes de cohérence, et de minimiser le risque d'incohérence (des notions introduites dans la Section 7.4). Ces objectifs ne sont pas les seuls à devoir être pris en compte. En effet, afin qu'un CDD soit capable de résister aux perturbations éventuelles, il est également envisageable de vouloir de maximiser sa flexibilité séquentielle ou temporelle (cf. Section 7.4.4).

Minimiser le risque d'incohérence et maximiser la flexibilité peuvent être des objectifs contradictoires. En effet, pour la flexibilité séquentielle, plus le nombre de séquences dominantes est grand et plus les intervalles $[s_{uv}]$ des opérations sont larges et donc plus le risque d'incohérence est grand. Inversement, si on souhaite réduire le risque d'incohérence, il faut accepter de perdre de la flexibilité séquentielle pour pouvoir réduire la largeur des intervalles. Cette dernière remarque est également à relier à la performance globale du système. Dans le cas d'un job shop ou l'objectif global est la minimisation du makespan, l'ordonnancement global sera d'autant plus performant que les intervalles $[s_{uv}]$ seront petits.

Pour simplifier, nous supposons qu'au niveau de chaque centre gérant un ensemble d'opérations, la structure pyramidale associée est définie de manière à ce que les sommets soient conservés dans le temps (pas de nouveau sommet) tout en autorisant une modification des valeurs des dates de disponibilité et d'échéance des opérations sommet et non-sommet, et de sorte à assurer que toute opération non sommet n'appartienne pas à une autre pyramide autre que celle à laquelle elle appartenait avant. Pour un souci de clarté, nous notons également par j l'indice d'une opération non sommet et par k celui d'un sommet.

La définition de nouveaux intervalles de consommation/livraison passe par la définition d'un ordre total entre les dates de disponibilité r_j, r_k et les dates d'échéance d_j, d_k des opérations. En effet, c'est grâce à la connaissance de cet ordre que l'on peut déterminer les valeurs des intervalles de début et de fin au mieux et au pire $[s_j]$, $[s_k]$ et $[f_j], [f_k]$ associés aux opérations et servant de base pour la (re)négociation. Les dates de disponibilité et d'échéance définissant l'ordre total sont des variables qui prennent leurs valeurs, comme nous le verrons plus loin, dans un intervalle de départ. La suite est consacrée à la définition de ces variables et aux contraintes qui les unies, puis à la proposition de modèles de négociation amont et aval, régissant respectivement les procédures *BackwardSolve* et *ForwardSolve*, permettant de résoudre localement et pour chaque centre le problème de

décision.

8.3.1 Choix d'un ordre total et caractérisation de séquences dominantes

Pour déterminer le domaine de variation des variables r_i et d_i des opérations, on se propose tout d'abord de définir des intervalles de dates de disponibilité et d'échéance $[\widetilde{r}_i^{min}\ \widetilde{r}_i^{max}]$ et $[\widetilde{d}_i^{min}\ \widetilde{d}_i^{max}]$ en fonction de ceux contractés $[r_i^{min}\ r_i^{max}]$ et $[d_i^{min}\ d_i^{max}]$ tout en assurant le respect des contraintes de cohérence globales (respect des gammes opératoires) données dans le chapitre précédent par le système d'inéquations 7.3, soit :

$$\begin{cases} \widetilde{r}_i^{min} = \max\ (r_i^{min},\ \underline{f}_{i-1}) \\ \\ \widetilde{r}_i^{max} = \min\ (r_i^{max},\ \overline{f}_{i-1}) \end{cases} \tag{8.2}$$

et

$$\begin{cases} \widetilde{d}_i^{min} = \max\ (d_i^{min},\ \underline{s}_{i+1}) \\ \\ \widetilde{d}_i^{max} = \min\ (d_i^{max},\ \overline{s}_{i+1}) \end{cases} \tag{8.3}$$

De là, nous nous proposons de définir les valeurs des dates de disponibilité r_i^0 et d'échéance d_i^0 et de les faire varier linéairement en fonction du temps selon les formules suivantes :

$$\begin{cases} r_i^0 = \widetilde{r}_i^{max} - \varepsilon_i^r\ (\widetilde{r}_i^{max} - \widetilde{r}_i^{min}) \\ \\ d_i^0 = \widetilde{d}_i^{min} + \varepsilon_i^d\ (\widetilde{d}_i^{max} - \widetilde{d}_i^{min}) \end{cases} \tag{8.4}$$

avec

$$\begin{cases} \varepsilon_i^r = \frac{\widetilde{r}_i^{min} + \widetilde{r}_i^{max}}{2H} \\ \\ \varepsilon_i^d = \frac{\widetilde{d}_i^{min} + \widetilde{d}_i^{max}}{2H} \end{cases} \tag{8.5}$$

Où, $H = \max_i(d_i^{max}, t)$, t étant l'instant courant, désigne l'horizon de l'ordonnancement et ϵ_i^r (resp. ϵ_i^d), une variable variant de 1 à 0. Ainsi, au fur et à mesure que l'instant courant se rapproche de $\widetilde{r}_i^{max}$ (resp. $\widetilde{d}_i^{min}$), i.e., plus l'instant courant sera éloigné de la fenêtre d'exécution (resp. achèvement), plus ϵ_i^r (resp. ϵ_i^d) sera grand et inversement, plus l'exécution (resp. achèvement) de l'opération devient éminente et plus ϵ_i^r (resp. ϵ_i^d) sera faible. En d'autres termes, ϵ_i^r est important quand les travaux peuvent être réalisés loin dans le temps. On se trouve alors dans une situation optimiste (en admettant que le produit attendu soit livré au plus tôt), où un risque d'incohérence élevé peut être toléré.

Inversement, quand ϵ_i^r est faible, les travaux sont proches dans le temps. On adopte alors une attitude pessimiste où la protection contre les aléas (mauvaise performance de l'amont) doit être maximale, le centre considérant que le produit lui sera livré au plus tard, d'où l'intérêt d'un risque d'incohérence faible dans cette situation (cf. Figure 8.10).

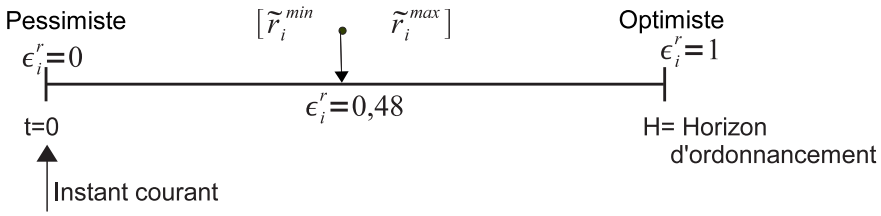

FIG. 8.10: Différentes situations selon certaines valeurs de ϵ_i^r

Partant des intervalles de départ $[r_i^0,\ d_i^0]$, et sous l'hypothèse qu'un sommet garde son statut de sommet et qu'une opération j non sommet est affectée au moins à une pyramide P_k avec $u(j) \leq k \leq v(j)$, nous considérons r_j, r_k, d_j et d_k, des variables de décision entières appartenant aux domaines donnés ci-après, et comme indiqués sur la Figure 8.11 : la fenêtre de réalisation de tout sommet $[r_k, d_k]$ est choisie dans la fenêtre initiale $[r_k^o, d_k^o]$ alors que la fenêtre de toute opération non sommet est choisie entre $[r_j^o, d_j^o]$.

$$r_k \in [r_k^o,\ d_k^o] \quad k = 1...m$$

$$d_k \in [r_k^o,\ d_k^o] \quad k = 1...m$$

$$r_j \in [r_j^o,\ d_{v(j)}^o - 1] \quad j = 1...n$$

$$d_j \in [r_{u(j)}^o + 1,\ \ d_j^o] \quad j = 1...n$$

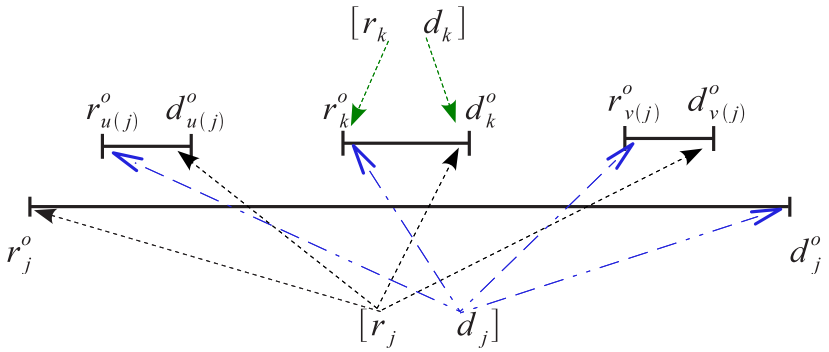

FIG. 8.11: Domaine de variation des variables r et d

Afin de modéliser le problème, on introduit les variables de décisions suivantes :

$$\begin{cases} x_{jk} = 1 \quad si \quad j \in P_k \\ 0 \quad sinon \end{cases}$$

$$\begin{cases} y_{jk} = 1 \quad si \quad j \notin P_k \ \wedge \ j \in P_{k'} / \ k' < k \\ 0 \quad sinon \end{cases}$$

et

$$\begin{cases} z_{jk} = 1 \quad si \quad j \notin P_k \ \wedge \ j \in P_{k'} / \ k' > k \\ 0 \quad sinon \end{cases}$$

Les contraintes qui lient les différentes variables sont alors les suivantes :

$$\begin{cases} (d_k + 1) - d_j \leq (d_k^o - r_{u(j)}^o)(1 - x_{jk}) \ \ \forall j = 1...n, \ \ \forall k | u(j) \leq k \leq v(j) & (c1) \\ r_j - (r_k - 1) \leq (d_{v(j)}^o - r_k^o)(1 - x_{jk}) \quad \forall j = 1...n, \ \ \forall k | u(j) \leq k \leq v(j) & (c2) \\ d_j - d_k \quad \leq (d_j^o - r_k^o)(1 - y_{jk}) \qquad \forall j = 1...n, \ \ \forall k | u(j) \leq k \leq v(j) & (c3) \\ r_k - r_j \quad \leq (d_k^o - r_j^o)(1 - z_{jk}) \qquad \forall j = 1...n, \ \ \forall k | u(j) \leq k \leq v(j) & (c4) \\ x_{jk} + y_{jk} + z_{jk} = 1 \qquad \forall j = 1...n, \ \ \forall k | u(j) \leq k \leq v(j) & (c5) \\ \sum_{k=u(j)}^{v(j)} x_{jk} \geq 1 \qquad \forall j = 1...n & (c6) \\ r_k \leq d_k; \ \ r_k \leq r_{k+1}; \ \ d_k \leq d_{k+1} \qquad \forall k = 1...m \end{cases} \quad (8.6)$$

Les contraintes (8.6) permettent de caractériser à partir de la structure d'intervalles initiale donnée par les valeurs des dates de disponibilité r_i^o et des dates d'échéances d_i^o des opérations, une nouvelle structure d'intervalles caractérisée par les variables r_i et d_i, telle que les sommets demeurent sommets et que les autres opérations non sommets appartiennent au moins à une pyramide. Plus explicitement, les contraintes $(c1)$ et $(c2)$ permettent d'assurer l'appartenance d'une opération j ($x_{jk} = 1$ ssi $r_j \leq r_k - 1$ et $d_j \geq d_k + 1$) ou sa non appartenance ($x_{jk} = 0$ et inégalités vraies) à la pyramide P_k. Si l'opération j est positionnée à gauche de la pyramide P_k alors la contrainte $(c3)$ est active, et si elle est placée à droite de P_k, alors la contrainte $(c4)$ est active ($c3$ est désactivée). La contrainte $(c5)$ impose une position unique pour l'opération j, i.e., soit dans P_k, ou bien à droite ou à gauche de P_k. Enfin la dernière contrainte $(c6)$ impose que j appartienne au moins à une pyramide.

À titre illustratif, considérons un exemple de 5 travaux et 3 sommets ayant une structure pyramidale initiale comme indiqué sur la figure 8.12a). On associe à chaque sommet 1, 2 et 3 ainsi qu'aux deux travaux non sommets i et j les variables x, y et z. La nouvelle structure d'intervalle donnée sur la figure 8.12b) est caractérisée par des valeurs des variables x, y et z suivantes :

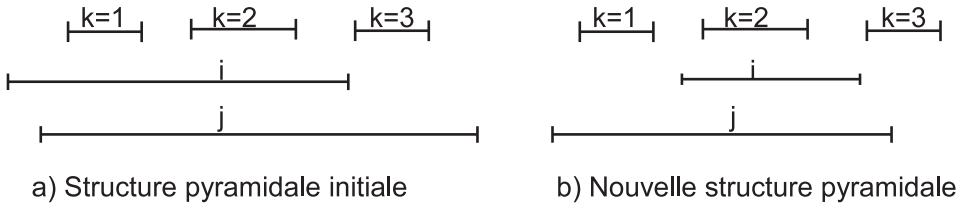

FIG. 8.12: Exemple illustratif

- $z_{i1} = 1$, $x_{i2} = 1$, $y_{i3} = 1$, i.e. le travail i appartient à la pyramide de sommet 2 ($x_{i2} = 1$) et est positionné à droite du sommet 1 ($z_{i1} = 1$) et à gauche du sommet 3 ($y_{i3} = 1$) ;
- $x_{j1} = 1$, $x_{j2} = 1$ et $y_{j3} = 1$, i.e., le travail j appartient aux deux pyramides P_1 et P_2 et est positionné à gauche du sommet 3 ($y_{j3} = 1$) ;
- le reste des variables, $x_{i1}, y_{i1}, y_{i2}, z_{i2}, x_{i3}, y_{i3}, y_{j1}, z_{j1}, y_{j2}, z_{j2}, x_{j3}$ et z_{j3} sont nulles.

Dans ce cas, on peut s'assurer, selon aussi l'ordre des dates de disponibilité r et d'échéance d initiales et nouvelles (cf. Figure 8.12), que les contraintes 8.6 sont vérifiées.

Comme évoqué précédemment, la construction par un CDD_u d'une nouvelle structure d'intervalles se fera de manière à respecter la consistance des engagements antérieurs. Nous verrons dans la suite que, selon si la négociation du CDD concerne un centre amont ou un centre aval, d'autres contraintes et un objectif spécifique sont définis. Nous distinguons alors un modèle pour la coopération amont et un modèle pour la coopération aval : ces deux modèles sont respectivement à la base des procédures *BackwardSolve* et *ForwardSolve* introduites dans la Section 8.2.2.

8.3.2 Modèle pour la coopération amont

Un centre de décision CDD_u peut, pour une opération i sur un produit, émettre une proposition de négociation au centre amont CDD_v, en lui proposant un délai de consommation de l'opération i correspondant au délai de livraison de l'opération prédécesseur $i-1$ réalisée par CDD_v. Cet intervalle correspond à $[s_i] = [lst_i^w,\ d_i - p_i]$. Nous considérons que la procédure *BackwardSolve* détermine cet intervalle en fonction des valeurs r_i^o et d_i^o calculées préalablement selon la formule 8.4. Rappelons que selon l'instant où la procédure est appelée, les valeurs choisies de r_i^o et d_i^o varient (en devenant de moins en moins risquées).

La procédure utilise aussi, la technique d'ordonnancement robuste décrite à la Section 7.3 pour déterminer la valeur de lst_i^w. Le calcul de lst_i^w repose sur la séquence dominante au pire, calant les opérations au plus tard et à la limite de leur deadline (à valeur d_i fixée). Remarquons que la limite supérieur de $[s_i]$ est donnée par $d_i - p_i$ du fait que le raisonnement établi pour le calcul des dates de début des opérations se fait à la base de la séquence au plus tard, et qu'au "pire" l'opération commence son exécution à l'instant $d_i - p_i$.

La variable lst_i^w est donc déterminée à partir de la séquence au plus tard au pire que nous avons notée $Seq_l^w(i)$ et donnée par l'expression 6.4 (voir Chapitre 6). En distinguant une opération sommet k d'une opération non sommet j, et en désignant par m le nombre de sommets et par n le nombre d'opérations non sommets, nous déduisons l'expression mathématique de cette variable s'écrivant comme suit :

$$\underline{s}_k = lst_k^w = \min(d_k,\ lst_{k+1}^w) - p_k - \sum_{i=1}^{n} p_i x_{ik}(1 - x_{i,k-1}), \quad \forall k = 1...m \tag{8.7}$$

$$\underline{s}_j = lst_j^w \leq x_{jk} lst_k^w - \sum_{i=1}^{n} p_i x_{ik} x_{i,k-1} \quad \forall j,\ \forall k \tag{8.8}$$

Notons que ces contraintes sont non linéaires (existence de produits de variables binaires), mais peuvent facilement être linéarisées en ajoutant une nouvelle variable binaire égale au produit de $x_{ik}x_{i,k-1}$.

Concentrons nous maintenant sur la notion de flexibilité. En considérant que chaque opération i dispose d'une flexibilité temporelle relative à l'intervalle contracté et mesurée par la largeur de $[lst_i^w,\ d_i - p_i]$, nous voulons maintenir pour le CDD_u lors des différentes négociations un niveau de flexibilité ($Flex_u$) donné . Nous avons montré dans le chapitre précédent, Section 7.4.4, comment peut être mesurer unede flexibilité d'un centre de décision.

Du point de vue global, on veut éviter la situation où il n'y aurait plus qu'un et un seul ordonnancement valide pour chaque CDD, et aussi celle inverse où les centres de décisions se réserveraient trop de flexibilité au détriment de la performance globale. La coopération dans notre cas suppose que la perte de flexibilité, induite par une remise en cause de décisions d'ordonnancement, soit collectivement répartie entre les centres de décision. Il faut par exemple éviter que ce soit le centre présentant le moins de flexibilité qui ait à subir la totalité des conséquences d'un aléa (dans la mesure où ces conséquences pourraient être supportées par des ressources ayant plus de flexibilité). Pour garantir ces conditions, et dans la perspective d'assurer le niveau de flexibilité temporelle désiré au niveau de chaque centre, on impose la contrainte suivante :

$$\frac{1}{m}\sum_{k=1}^{m}(d_k - lst_k^w - p_k) + \frac{1}{n}\sum_{j=1}^{n}(d_j - lst_j^w - p_j) \geq Flex_u \tag{8.9}$$

Notons, que plus la valeur de l'expression donnée dans la Formule 8.9 est grande, plus la marge disponible dans le pire des cas pour réaliser des opérations est importante ; $Flex_u$ étant la plus grande valeur de flexibilité admissible par le CDD_u.
u

Nous avons vu dans l'algorithme distribué proposé dans la section précédente, que les propositions échangées entre centres sont accompagnées de la valeur de la flexibilité liée aux centres expéditeurs. Afin d'éviter des situations de trop grande disparité entre les valeurs de flexibilité des différents centres, nous considérons que, pour répondre à une requête, le centre CDD_u compare sa flexibilité avec celle de l'expéditeur CDD_v de la requête, et s'interdit de dégrader sa flexibilité au dela d'un certain seuil au dessous de celle du centre demandeur. Ceci permet de favoriser les comportements coopératifs, dans le mesure où un centre peut accepter de perdre de sa flexibilité afin de répondre au mieux à la requête du centre initiateur. Nous encourageons ainsi les centres à adopter des comportements équitables où les valeurs de flexibilité seraient proches les unes des autres.

Toujours dans le but de définir les contraintes qui régissent le comportement d'un centre, nous considérons lors de la prise en compte d'une perturbation, que le CDD recherche à déterminer de nouveaux intervalles $[s_i]$ des opérations, les plus cohérents possible avec ceux contractés $[r_i]$ (respect de la contrainte de cohérence locale, cf. section 7.4.1). Nous avons vu que la procédure *BackwardSolve* cherche à réaliser l'opération i dans les délais impartis, sans remettre en cause ses engagements de consommation (i.e., les délais de livraison de $i-1$). Dans le cas contraire, la procédure établit la liste $\mathcal{U}$ des opérations incohérentes (les opérations $i-1$ en retard) pour lesquelles il va falloir renégocier les intervalles de fin des opérations amont. En posant $\mathcal{U} = \{i | U_i = 1\}$, nous notons :

$$\begin{cases} U_j = 1 & si \quad lst_j^w < r_j^{min} \\ 0 & sinon \quad \forall j = 1..n \end{cases} \tag{8.10}$$

$$\begin{cases} U_k = 1 & si \quad lst_k^w < r_k^{min} \\ 0 & sinon \quad \forall k = 1..m \end{cases} \tag{8.11}$$

U_i est donc une variable binaire qui permet d'indiquer si la date de début d'une opération est cohérente ou pas avec l'intervalle contracté.

Cherchant à formaliser le problème permettant à la procédure *BackwardSolve* de fixer les décisions d'ordonnancement, i.e., des valeurs de r_i et d_i permettant de définir un ordre total et de déduire les valeurs de lst_i, nous avons jusqu'ici donné les contraintes qui lient les différentes variables et qui permettent de maintenir un niveau de flexibilité donné pour le CDD. Nous considérons à présent l'objectif à satisfaire. Il s'agit d'assurer au maximum la cohérence des dates de début de réalisation i.e., minimiser l'incohérence de ces dates tout en considérant le niveau de risque ϵ_i^r associé à chaque opération cf. Section 8.3.1.

La fonction objectif est alors donnée par l'expression suivante. Elle consiste à minimiser la somme pondérée du nombre d'incohérence, en pénalisant d'avantage les opérations dont la réalisation est proche.

$$\min z = \sum_{k=1}^{m}(1-\varepsilon_k^r)U_k + \sum_{j=1}^{n}(1-\varepsilon_j^r)U_j \tag{8.12}$$

Étant données les contraintes et la fonction objectif présentées plus-haut, et considérant qu'une fonction de produit de variables binaires peut être linéarisée, le modèle pour la négociation amont noté M_u, est un modèle de programmation linéaire en nombres entiers se présentant comme suit :

Objectif : (8.12)

s.t.
$$\begin{cases} (8.11) \\ (8.10) \\ (8.6) \\ (8.7) \\ (8.8) \\ (8.9) \\ r_k,\ d_k,\ r_j,\ d_j, lst_k^w, lst_j^w \in \mathbb{Z}_+ \quad \forall j = 1..n,\ \forall k = 1..m \\ x_{jk},\ y_{jk},\ z_{jk},\ U_j,\ U_k \in \{0,1\} \quad \forall j = 1..n,\ \forall k = 1..m \end{cases}$$

la procédure *BackwardSolve*, par le biais du modèle de coopération amont M_u, permet à l'occurrence d'un événement (une proposition émanent de l'amont) de déterminer, selon les données actualisées, la nouvelle structure d'intervalles gérée par le centre en question. Il est ainsi possible de déduire l'ensemble des opérations $\mathcal{U} = \{x | U_x = 1\}$ pour lesquelles les contraintes de cohérence locales sont violées. L'intervalle $[lst_x^w,\ d_x - p_x]$ avec $x \in \mathcal{U}$ représentent alors le nouveau intervalle de consommation pour le CDD réalisant x. Étant incohérent avec l'intervalle contracté, il est proposé au centre amont réalisant l'opération prédécesseur de x en lui adressant une requête R.

8.3.3 Modèle pour la coopération aval

De manière similaire avec le modèle de coopération amont, un centre de décision CDD_u contracte un intervalle de livraison $[f_i]$ pour une opération i qu'il gère (initiateur). Sachant qu'au "mieux" une opération i est disponible à l'instant $r_i + p_i$, on associe à l'intervalle $[f_i]$ de fin de réalisation de i l'intervalle $[r_i + p_i, eft_i^w]$. Nous considérons que la procédure

ForwardSolve détermine cet intervalle en fonction des valeurs r_i^o et d_i^o calculées préalablement, et de lst_i^w déterminé à partir de la séquence au plus tôt au pire (notée $Seq_e^w i$ donnée par l'expression 6.2 au chapitre 6), calant les opérations au plus tôt (à valeur r_i fixée). En distinguant une opération sommet k d'une opération non sommet j, les expressions mathématiques des variables eft_k^w et eft_j^w sont données comme suit :

$$\overline{f}_k = eft_k^w = \max(d_k,\ eft_{k-1}^w) + p_k + \sum_{i=1}^{n} p_i x_{ik}(1 - x_{i,k+1}), \quad \forall k = 1...m \tag{8.13}$$

$$\overline{f}_j = eft_j^w \geq x_{jk} eft_k^w + \sum_{i=1}^{n} p_i x_{ik} x_{i,k+1} \quad \forall j,\ \forall k \tag{8.14}$$

De la même manière que précédemment, pour répondre à la requête d'un centre aval et proposer un intervalle de livraison $[r_i + p_i, eft_i^w]$, on veut imposer au niveau du CDD_u un niveau de flexibilité temporelle $flex_u$ donné, par le respect de la contrainte suivante :

$$\frac{1}{m}\sum_{k=1}^{m}(eft_k^w - r_k - p_k) + \frac{1}{n}\sum_{j=1}^{n}(eft_j^w - r_j - p_j) \geq Flex_u \tag{8.15}$$

Nous considérons, lors de la contractualisation avec l'aval, que le centre CDD_u cherche à déterminer, via la procédure *ForwardSolve*, les nouveaux intervalles de fin $[f_i]$ des opérations, les plus cohérents possible avec ceux contractés $[d_i]$ (respect de la contrainte de cohérence locale, cf. section 7.4.1), et tout en maintenant un niveau de flexibilité donné. Nous avons vu que la procédure *ForwardSolve* cherche à achever l'opération i dans les délais impartis, sans remettre en cause ses engagements de livraison (i.e., les délais de consommation de $i+1$). Dans le cas contraire, la procédure établit la liste $\mathcal{U}$ des opérations incohérentes (les opérations i en retard). En posant $\mathcal{U} = \{i | U_i = 1\}$, nous notons :

$$\begin{cases} U_j = 1 & si \quad eft_j^w > d_j^{max} \\ 0 & sinon \quad \forall j = 1..n \end{cases} \tag{8.16}$$

$$\begin{cases} U_k = 1 & si \quad eft_k^w > d_k^{max} \\ 0 & sinon \quad \forall k = 1..m \end{cases} \tag{8.17}$$

La procédure *ForwardSolve* permet de fixer les décisions d'ordonnancement en définissant de nouvelle structure d'intervalles et en déduisant les valeurs de eft_i. Nous considérons que l'objectif à satisfaire est la minimisation de l'incohérence de ces dates de fin, tout en prenant en compte le niveau de risque ϵ_i^d associé à chaque opération cf. Section 8.3.1. Ceci s'exprime alors par :

$$\min z = \sum_{k=1}^{m}(1 - \varepsilon_k^d)U_k + \sum_{j=1}^{n}(1 - \varepsilon_j^d)U_j \tag{8.18}$$

Étant données les contraintes et la fonction objectif présentées plus-haut, et considérant qu'une fonction de produit de variables binaires peut être linéarisée, le modèle pour la négociation aval noté M_D, est un modèle de programmation linéaire en nombres entiers s'écrivant comme suit :

$$\text{Objectif : } (8.18)$$

$$\text{s.t.} \quad \begin{cases} (8.17) \\ (8.16) \\ (8.6) \\ (8.13) \\ (8.14) \\ (8.15) \\ r_k,\ d_k,\ r_j,\ d_j, eft_k^w, eft_j^w \in \mathbb{Z}_+ \quad \forall j = 1..n,\ \forall k = 1..m \\ x_{jk},\ y_{jk},\ z_{jk},\ U_j,\ U_k \in \{0,1\} \quad \forall j = 1..n,\ \forall k = 1..m \end{cases}$$

Le modèle de coopération aval permet de déterminer une nouvelle structure d'intervalles et de déduire l'ensemble X des opérations x pour lesquelles les contraintes de cohérence locales sont violées, i.e. $U_x = 1$. Les intervalles $[r_x + p_x, eft_x^w]$ sont les nouveaux intervalles de livraison pour le CDD réalisant X. Ces intervalles étant incohérents avec ceux contractés, ils sont proposés aux centres aval réalisant les opérations succédant les opérations de X.

Enfin, le modèle de coopération aval sert de base à la procédure *ForwardSolve* pour déterminer, afin de répondre à une proposition de l'aval et selon les données actualisées, la nouvelle structure d'intervalles gérée par le CDD. Il est ainsi possible de déduire l'ensemble des opérations $\mathcal{U} = \{x | U_x = 1\}$ pour lesquelles les contraintes de cohérence locales sont violées. Les intervalles $[r_x + p_x, eft_x^w]$ avec $x \in \mathcal{U}$ représentent alors les nouveaux intervalles de livraison pour les CDDs réalisant $\mathcal{U}$. Étant incohérents avec ceux contractés, ils sont proposés aux centres aval réalisant les opérations successeurs $x + 1$. Un message A est adressé au centre aval réalisant l'opération $x+1$ si cette dernière a fait objet d'une requête, un message D, dans le cas contraire. Notons, que les autres opérations n'appartenant pas à $\mathcal{U}$ peuvent faire objet d'un message d'information I si l'intervalle a été modifié sans qu'il soit pour autant incohérent avec celui contracté.

8.4 Conclusion

La problématique d'ordonnancement distribué sous incertitudes est définie comme étant un mécanisme de co-décision des intervalles cibles de réalisation ou d'achèvement (livraison/consommation) des opérations prises en charge par chaque centre de décision. La méthode de résolution proposée est basée sur une coopération entre les acteurs. Elle permet à un centre de solliciter ou répondre aux sollicitations, des autres centres amont et aval. Nous avons identifié plusieurs types de messages permettant aux différents acteurs de communiquer lors du processus de négociation. À l'occurrence d'un événement, si le centre ne peut absorber la perturbation sans remettre en cause les décisions antérieures, l'acteur négocie en amont la livraison de certaines opérations plus tôt que prévu, et les conversations peuvent alors se propager en arrière, de proche en proche et dans la mesure du possible, dans le réseau de centres de décision et cela de manière à conserver un certain niveau de flexibilité et de cohérence. Une fois la négociation amont achevée, une propagation en aval des retards est amorcée, et des contractualisations s'engagent et se propagent jusqu'à recouvrer un état cohérent. Un algorithme distribué régissant le comportement de chaque CDD et permettant de déterminer les décisions d'ordonnancement au fur et à mesure de l'évolution de l'atelier est proposé et garantit la convergence des négociations.

Afin de cerner les décisions d'ordonnancement, des méthodes locales sont aussi proposées. Elles sont basées sur une approche d'ordonnancement robuste. Ces méthodes utilisent des modèles de coopération amont et aval basés sur la programmation mathématique. Le modèle amont, permet à un centre de calculer, suite à la réception d'une requête sur une fenêtre de livraison, ses nouvelles dates de début. Le modèle aval permet quant à lui, de répondre à la requête d'un centre amont, en proposant un intervalle de livraison, le meilleur que le centre soit capable d'assurer.

CONCLUSION ET PERSPECTIVES

Les problèmes d'ordonnancement, largement abordés dans la littérature, sont généralement réputés difficiles à résoudre. Cette difficulté est liée :

- à la taille et à la complexité de l'espace de solutions pour la recherche d'une solution optimale ;
- au fait que l'environnement d'application est par essence dynamique ; la solution d'ordonnancement proposée devenant alors rapidement non valide et devant être adaptée en temps réel.

Dans le cadre des travaux menés dans cette thèse, deux axes de recherche ont été développés. Le premier concerne la résolution exacte d'un problème d'ordonnancement déterministe à une machine, en considérant à la fois les critères d'admissibilité, le retard algébrique maximal ainsi que la minimisation du nombre de travaux en retard. Le second axe concerne un problème d'ordonnancement multi-machines évoluant dans un contexte perturbé, et la proposition d'une approche de résolution à la fois coopérative et robuste.

Le socle de cette thèse est un théorème de dominance démontré dans les années quatre-vingt pour le problème à une machine. Son avantage principal réside dans sa capacité à caractériser un ensemble de séquences dominantes, sans les énumérer, grâce à un ordre partiel, en ne considérant exclusivement que les relations d'ordre entre les dates de disponibilité et les dates d'échéance des opérations. Le théorème a été formulé dans un but premier de limiter la complexité liée à la recherche d'une solution admissible, puis a été étendu pour être exploité dans un contexte de recherche de robustesse. Dans ce dernier cas, l'ensemble de solutions flexibles caractérisé par le théorème de dominance, permet de réagir aux perturbations en basculant facilement d'une solution à une autre dans l'ensemble (flexibilité séquentielle), lors de la mise en œuvre, tout en garantissant un niveau de performance ; les performances de l'ensemble flexible pouvant être évaluées en temps polynomial.

Dans un premier volet, en se basant sur le théorème de dominance, nous avons énoncé de nouvelles conditions analytiques et numériques de dominance. Ces conditions permettent, d'identifier la séquence la plus dominante parmi toutes les séquence que le théorème ca-

ractérise et, de prouver ainsi l'existence ou l'absence d'une solution admissible. Nous nous sommes aussi intéressés à l'étude de la dominance du théorème vis-à-vis du critère de minimisation du nombre de travaux en retard. Nous avons identifié sous quelles conditions ce théorème restait dominant vis-à-vis de ce nouveau critère. Puis, à la base des résultats trouvés, nous avons proposé de nouvelles formulations mathématiques sous forme de programmation linéaire en nombres entiers pour la modélisation du problème à une machine. En considérant tout d'abord le problème d'admissibilité, un premier modèle de programmation linéaire pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité est proposé ; l'admissibilité de la solution ainsi trouvée dépend de la valeur de la fonction objectif.

Ensuite, nous avons démontré l'équivalence du modèle initial (recherche de la solution la plus dominante) au cas de résolution du problème de minimisation du plus grand retard algébrique $1|r_i|L_{max}$; la recherche de la solution la plus dominante est démontrée équivalente à la solution qui minimise L_{max}. De nombreux résultats expérimentaux, réalisés en utilisant un solveur commercial, viennent prouver l'efficacité de ce second modèle en comparaison avec la procédure Branch-and-bound de Carlier. Il ressort que le PLNE s'adapte bien à la résolution des instances de grandes tailles, particulièrement difficiles à résoudre. Il n'existe pas, à notre connaissance, d'autres modèles PLNE qui puissent être comparés, en terme d'efficacité, à la procédure de Carlier ; on peut alors conclure qu'il est possible de résoudre le problème $1|r_i|L_{max}$ en utilisant la programmation mathématique [Briand *et al.* 10a], bien qu'en général, la procédure de Carlier demeure très efficace.

Nous nous sommes aussi intéressés dans ce volet au problème de minimisation du nombre de travaux en retard $1|r_i|\sum U_i$. Nous avons montré comment adapter le modèle de programmation linéaire établi pour la recherche de la solution la plus dominante vis-à-vis de l'admissibilité pour calculer des bornes inférieure et supérieure pour le problème $1|r_i|\sum U_i$. La borne supérieure est obtenue en supposant que tous les sommets du problème sont à l'heure et que les autres travaux non-sommets peuvent être en retard, rendant ainsi la condition de dominance vraie vis-à-vis de $\sum U_i$; la borne inférieure, quant à elle, est obtenue en relâchant le problème de manière à ce que sa structure pyramidale devienne parfaite, rendant ainsi le théorème des pyramides dominant vis-à-vis de $\sum U_i$. Les expérimentations réalisées sur les instances de Baptiste et al. ont permis d'évaluer l'efficacité révélée en termes de performance et temps de calcul des modèles ayant permis de calculer les bornes [Ourari *et al.* 09a]. Il est important de préciser que nous nous sommes comparés aux méthodes ad-hoc publiées antérieurement à notre travail, et que si nous nous comparons à la méthode de [M'Hallah & Bulfin 07] publiée durant la préparation de notre travail, nos résultats se révèlent moins performants.

Pour conclure ce premier volet de notre étude, nous précisons que les modèles de programmation linéaire en nombres entiers que nous proposons sont originaux. De plus, nous avons utilisé une méthode générique. En effet, le développement de modèles PLNE pour la résolution des problèmes d'ordonnancement à une machine est toujours intéressant dans la

mesure où il est possible de les adapter pour prendre en compte de nouvelles contraintes ou de nouveaux critères de performance. D'autres perspectives de travail sont aussi envisageables. En effet, exploitant les nouvelles conditions de dominance que nous avons énoncées, il nous semble qu'une bonne perspective de travail consisterait à aborder l'étude de problèmes plus simples : le cas du problème $1|p_i = p, r_i|\sum U_i$, et le cas du problème à une seule pyramide parfaite. Pour le problème général de minimisation de travaux en retard que nous avons traité, il serait envisageable d'exploiter les modèles qui déterminent les bornes supérieure et inférieure, dans le cadre d'une recherche arborescente afin de déterminer la solution optimale de manière exacte. L'idée serait d'utiliser la séquence correspondant à la borne supérieure pour guider la recherche arborescente, et d'appliquer une règle de séparation sur les positions des sommets en retard ; la séparation serait binaire, i.e., les deux nœuds créés considèrent respectivement : le cas où le sommet correspondant à la position choisie est effectivement en retard (autorisant ainsi la suppression de l'intervalle du nœud père pour obtenir alors une nouvelle structure pyramidale avec éventuellement un nombre de sommets plus important si le sommet en question est générateur) ; et le second cas où on considérerait que le sommet soit à l'heure et imposerait ainsi la nouvelle contrainte dans les modèles du nœud père.

Enfin, de manière globale, les modèles proposés présentent d'autres intérêts, résidant dans la possibilité de les étendre pour considérer des problèmes d'ordonnancement plus complexes (e.g. le job shop ou le problème à machines parallèles)

Le travail abordé dans la première partie du manuscrit est une contribution théorique à la modélisation des problèmes d'ordonnancement déterministe à une machine en utilisant un concept de dominance. L'étude de ce type de problème est d'un intérêt omniprésent dans le domaine de la recherche en raison de la nature fortement combinatoire de ces problèmes. Toutefois, au vue de la réalité de l'environnement d'application de l'ordonnancement, à la fois non déterministe et incertain, il parait important d'aborder le problème de gestion des incertitudes en ordonnancement dans le but de réduire l'écart entre la théorie (ordonnancement prévisionnel) et le domaine pratique (l'ordonnancement réellement mis en œuvre). L'étude présentée dans le second volet de cette thèse traite de cette problématique et propose une nouvelle approche de résolution. Nous sommes partis du constat que la majorité des systèmes d'ordonnancement, avec ou sans prise en compte des incertitudes, s'appuie sur une politique centralisée de résolution et suppose donc l'existence d'une entité qui prend les décisions globales devant être respectées par l'ensemble des acteurs du système. Hors, nous pensons qu'il n'est pas toujours réaliste de supposer qu'une entité dispose d'une vision globale de tout le système et qu'il est davantage judicieux de tenir compte de l'autonomie de décision et des objectifs propres de chaque acteur. Nous avons alors considéré l'ordonnancement sous l'hypothèse d'une fonction distribuée où la solution globale résulte d'une coopération entre les différents acteurs.

Après avoir passé en revue les techniques d'ordonnancement sous incertitudes existantes et, donné un aperçu des approches distribuées en ordonnancement, nous avons, dans un premier temps, rappelé une méthode d'ordonnancement robuste à une machine que nous avons utilisée dans le cadre de la problématique abordée dans la dernière partie du manuscrit. C'est une approche intéressante pour prendre en compte les incertitudes. Elle consiste à fournir durant la phase hors ligne de résolution un ensemble flexible de solutions, et d'utiliser cette flexibilité pour réagir, en ligne, aux aléas. La flexibilité produite est séquentielle et utilise le concept de dominance, la performance au mieux et au pire de l'ensemble flexible de solutions caractérisé étant calculable en temps polynomial.

Ensuite, considérant le cas particulier du problème d'atelier job shop, nous avons défini la problématique d'ordonnancement en environnement dynamique. Il s'agit d'ordonnancer un ensemble d'opérations impliquant plusieurs ressources assimilées à des centres de décision, possédant chacune une autonomie décisionnelle, et prenant chacune en charge l'exécution d'un sous-ensemble d'opérations. Les centres, évoluant dans un contexte perturbé, se communiquent des intervalles de livraison/consommation (dates de fin ou de début) des opérations que chacun gère ; ils sont alors libres dans leur organisation interne tant qu'ils garantissent l'achèvement (consommation) des opérations dans les intervalles contractés. Les valeurs des intervalles sont négociées de manière à atteindre un compromis satisfaisant l'ensemble des acteurs engagés dans la négociation. Le but recherché est la cohérence des décisions prises collectivement pour chaque acteur tout en maintenant un niveau donné de flexibilité décisionnelle qui soit équitablement réparti entre les différents acteurs constituant l'atelier de production.

Dans un dernier temps, nous avons proposé une nouvelle approche de résolution à la fois robuste et coopérative. L'approche permet de retourner une solution au problème en adoptant une démarche distribuée où les centres de décision négocient des intervalles de livraison ou consommation des opérations qu'ils gèrent et par lesquelles ils sont liés les uns avec les autres. Nous avons considéré que les centres communiquent de façon asynchrone à travers plusieurs types de messages. Le comportement de chaque centre, obéissant à des règles de coopération adoptées collectivement, est décrit par un algorithme distribué. L'idée de base consiste à absorber, si possible, les modifications des intervalles par négociation amont en anticipant les dates de début, sinon, retarder certaines dates de fin par propagation aval. Les décisions d'ordonnancement prises au niveau de chaque centre sont fixées par des méthodes locales : ce sont des modèles de programmation mathématique que nous avons proposés et décrits.

Le travail réalisé, dans ce second volet de recherche, est axé essentiellement sur la formalisation de la problématique d'ordonnancement robuste et coopératif en milieu perturbé. Les bases et les concepts de la résolution distribuée du problème ont été posés. Toutefois dans la perspective de valider l'approche de résolution, il reste nécessaire de l'implémenter, en vue de la tester, et sans doute, de l'améliorer. Aussi, il semblerait naturel, afin d'évaluer les performances de l'approche distribuée de résolution proposée de procéder à des compa-

raisons ; une perspective de recherche serait donc de développer une approche centralisée de résolution pour comparer les performances obtenues dans les cas distribué et centralisé.

Pour terminer, notons que dans la problématique d'ordonnancement distribué abordée dans ce manuscrit, la notion de réseau de centres de décisions autonomes négociant les décisions d'ordonnancement par coopération peut être rattachée à la notion d'agents. En effet, considérer une ressource comme une entité autonome capable de résoudre un problème revient à effectuer une *"agentification"* de l'atelier job shop. La notion d'agent a souvent été au cœur des méthodes recourant à la distribution des capacités décisionnelles sur un processus de résolution en interaction : des méthodes basées sur le paradigme des systèmes multi-agents (SMA). Il en découle que notre travail s'inscrit dans un spectre plus large qui est celui des SMA, et que les bases et les concepts que nous avons posés peuvent être réutilisés dans une perspective d'application SMA [Briand *et al.* 10b].

Bibliographie

[Adams *et al.* 88] J. Adams, E. Balas & D. Zawack. *The shifting bottleneck procedure for job shop scheduling.* Management Science, vol. 34, no. 3, pages 391–401, 1988.

[Albino *et al.* 05] V. Albino, N. Carbonara & I. Giannoccaro. *Supply chain cooperation in industrial district : A simulation analysis.* European Journal of Operational Research, vol. 177, no. 1, pages 261–280, 2005.

[Allen 91] J.F. Allen. *Time and time again : The many ways to represent time.* International Journal of Intelligent Systems, vol. 6, no. 4, pages 341–355, 1991.

[Aloulou *et al.* 02] M.A. Aloulou, M-C. Portmann & A. Vignier. *Predictive-Reactive Scheduling for the Single Machine Problem.* Dans Eighth International Workshop on Project Management and Scheduling, pages 39–42, Valencia, Spain, 2002.

[Aloulou *et al.* 07] M.A. Aloulou, M.Y. Kovalyov & M-C. Portmann. *Evaluating flexible solutions in single machine scheduling via objective function maximization : the study of computational complexity.* RAIRO Opererations Research, vol. 41, no. 1, pages 1–18, 2007.

[Aloulou 02] M.A. Aloulou. *Structure flexible d'ordonnancements à performances contrôlées pour le pilotage d'atelier en présence de perturbations.* thèse de doctorat, Institut national de lorraine, Lorraine, France, 2002.

[Archimede & Coudert 01] B. Archimede & T. Coudert. *Reactive scheduling using a multi-agent model : the SCEP framwork.* Engineering Application of Artificiel Intelligence, vol. 14, no. 5, pages 667–683, 2001.

[Artigues *et al.* 02] C. Artigues, C. Briand, M-C. Portmann & F. Roubellat. *Pilotage d'atelier basé sur un ordonnancement flexible.* Dans Méthodes du pilotage des systèmes de production sous la direction de P. Pujo et J-P. Kiefer, pages 61–97. Hermès, 2002.

[Artigues *et al.* 05] C. Artigues, J-C. Billaut & C. Esswein. *Maximisation of solution flexibility for robust shop scheduling.* European Journal of Operational Research, vol. 165, no. 12, pages 314–328, 2005.

[Artigues 97] C. Artigues. *Ordonnancement en temps réel d'ateliers avec temps de préparation des ressources.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 1997.

[Baker 74] K.R. Baker. Introduction of sequencing and scheduling. Wiley, New York, Etats unis, 1974.

[Baptiste *et al.* 03] P. Baptiste, L. Peridy & E. Pinson. *A Banch and Bound to Minimize the Number of late Jobs on a Single Machine with Release Time Constraints.* European Journal of Operations Research, vol. 144, no. 1, pages 1–11, 2003.

[Baptiste 99a] P. Baptiste. *A $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs.* Operations Research Letters, vol. 24, pages 175–180, 1999.

[Baptiste 99b] P. Baptiste. *Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times.* Journal of Scheduling, vol. 2, pages 245–252, 1999.

[Barnard 38] C-J. Barnard. *The functions of the executive.* Dans Harvard Univ.Press, Cambridge, 1938.

[Bertsimas & Sim 04] D. Bertsimas & M. Sim. *The price of robustness.* Operations Research, vol. 52, no. 1, pages 35–53, 2004.

[Bessiere *et al.* 01] C. Bessiere, A. Maestre & P. Messeguer. *Distributed dynamic backtracking.* Dans Workshop on Distributed Constraint of Seventeenth International Joint Conference on Artificial Intelligence, Washington, USA, 2001.

[Billaut *et al.* 05] J.C. Billaut, A. Moukrim & E. Sanlaville. chapitre Introduction à la flexibilité et à la robustesse en ordonnancement, pages 15–34. Hermès, 2005.

[Billaut 93] J-C. Billaut. *Prise en compte de ressources multiples et des temps de préparation dans les problèmes d'ordonnancement en temps réel.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 1993.

[Billaut 99] J-C. Billaut. *Recherche Opérationnelle et aide à la décision pour les problèmes d'ordonnancement.* Habilitation à diriger des recherche, Laboratoire d'Informatique, E3i, Université François Rabelais, Tours, France, 1999.

[Blackstone *et al.* 82] J.H. Blackstone, D.T. Phillips & G.L. Hogg. *A state-of-the-art survey of dispatching rules for manufacturing job shop operations.* International Journal of Production Research, vol. 20, no. 1, pages 27–47, 1982.

[Blazewicz *et al.* 96] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt & J. Weglarz. Scheduling in computer and manufacturing processes. Springer Verlag, 1996.

[Boujut *et al.* 02] J.F. Boujut, J.B. Cavaillé & A. Jeantet. *Instrumentation de la coopération.* Dans Coopération et connaissance dans les systèmes industriels, pages 91–109. Hermès, 2002.

[Briand & La 03] C. Briand & H.T. La. *Une procédure par séparation et évaluation progressive pour l'ordonnancement robuste de problèmes à une machine.* Dans Recherche Informatique Vietnam & Francophonie 2003 (RIVF2003), pages 11–16, Hanoi, Vietnam, 2003.

[Briand & Ourari 08] C. Briand & S. Ourari. *A cooperative approach for Job Shop Scheduling under Uncertainties.* Dans Collaborative Decision Making : Perspective and challengs, pages 5–15. P. Zaraté et al. (Eds.) IOS Press, 2008.

[Briand & Ourari 09] C. Briand & S. Ourari. *Une formulation PLNE efficace pour* $1|r_i|L_{max}$. Dans ROADEF09, Nancy-France, 10-12 Février 2009.

[Briand *et al.* 03a] C. Briand, H.T. La & J. Erschler. *Ordonnancement de problèmes à une machine : une aide à la décision pour un compromis flexibilité vs performance.* Dans 5ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'03), pages 146–147, Avignon, France, 2003.

[Briand *et al.* 03b] C. Briand, H.T. La & J. Erschler. *Une approche pour l'ordonnancement robuste de tâches sur une machine.* Dans 4ème Conférence Francophone de MOdélisation et SIMulation (MOSIM'03), pages 205–211, Toulouse, France, 2003.

[Briand *et al.* 05] C. Briand, M-J. Huguet, H.T. La & P. Lopez. *Approches par contraintes pour l'ordonnancement robuste.* Dans Flexibilité et Robustesse en Ordonnancement sous la direction de J-C. Billaut et A. Moukrim et E. Sanlaville, pages 181–208. Hermès, 2005.

[Briand *et al.* 06a] C. Briand, H.T. La & J. Erschler. *A new sufficient condition of optimality for the two-machine flowshop problems.* European Journal of Operational Research, vol. 169, no. 3, pages 712–722, 2006.

[Briand *et al.* 06b] C. Briand, S. Ourari & B. Bouzouia. *Minimizing the number of late jobs in single machine scheduling with nested execution intervals.* Dans IEEE/SSSM, Troyes, France, 2006.

[Briand *et al.* 07] C. Briand, S. Ourari & B. Bouzouia. *A new dominance condition for mono-pyramidal single machine scheduling problems.* Dans MAPSP, Istanbul, Turkey, 2007.

[Briand *et al.* 08] C. Briand, S. Ourari & B. Bouzouia. *Une approche coopérative pour l'ordonnancement sous incertitudes.* Dans 7ème conférence internationale de MOdélisation et SIMulation (MOSIM08), Paris-France, 2008.

[Briand *et al.* 10a] C. Briand, S. Ourari & B. Bouzouia. *An efficient ILP formulation for the single machine scheduling problem.* RAIRO-Operation Research, vol. 44, no. 1, pages 61–71, 2010.

[Briand *et al.* 10b] C. Briand, S. Ourari & B. Bouzouia. *Un algorithme coopératif pour un problème d'atelier job shop multi-agent.* Dans 8ème conférence internationale de MOdélisation et SIMulation (MOSIM10), Hammamet-Tunisie, 2010.

[Brito *et al.* 09] I. Brito, A. Meisels, P. Meseguer & R. Zivan. *Distributed constraint satisfaction with partially known constraints.* Constraints, vol. 14, no. 2, pages 199–234, 2009.

[Camalot *et al.* 97] J-P. Camalot, P. Esquirol, M-J. Huguet & J. Erschler. *Aide à la décision et à la négociation dans un problème de gestion de production distribué.* Journée du Groupement pour la Recherche en Productique, pages 159–180, 97.

[Camalot 00] J-P. Camalot. *Aide à la décision et à la coopération en gestion du temps et des ressources.* Thèse de doctorat, Institu National des sciences appliquées de Toulouse, 2000.

[Campagne & Sénéchal 02] J. Campagne & O. Sénéchal. *Les nouvelles exigences de la coopération.* Dans Coopération et connaissance dans les systèmes industriels sous la direction de R. Soënen et J. Perrin, Lavoisier, pages 51–67. Hermes Science, 2002.

[Carlier & Chrétienne 88] J. Carlier & P. Chrétienne. Problèmes d'ordonnancement : modélisation/complexité/algorithmes. Masson, 1988.

[Carlier 81] J. Carlier. *Problèmes d'ordonnancement à durées égales.* QUESTIO, vol. 5, no. 4, pages 219–228, 1981.

[Carlier 82] J. Carlier. *The one-machine sequencing problem.* European Journal of Operational Research, vol. 11, pages 42–47, 1982.

[Cesta *et al.* 98] A. Cesta, A. Oddi & S.F. Smith. *Profile Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems.* Dans the 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98), page 214Ű223, 1998.

[Chauvet 99] F. Chauvet. *Ordonnancement en temps réel dans les problèmes à encours limités.* thèse de doctorat, Université de Metz, Metz, France, 1999.

[Cheng *et al.* 02] J. Cheng, G. Steiner & P. Stephenson. *Fast algorithms to minimize makespan or maximum lateness in the two-machine flow shop with release times.* Journal of Scheduling, vol. 5, pages 71–92, 2002.

[Chiang & Fox 90] W-Y. Chiang & M.S. Fox. *Protection againts uncertainty in a deterministic schedule.* Dans Fourth International Conference on Expert Systems in Production and Operations Management, South California, USA, 1990.

[Chrobak *et al.* 06] M. Chrobak, C.Dürr, W. Jawor, L. Kowalik & M. Kurowski. *A Note on Scheduling Equal-Length Jobs to Maximize Throughput.* Journal of Scheduling, vol. 9, no. 1, pages 71–73, 2006.

[Chrétienne *et al.* 95] P. Chrétienne, E.G. Coffman & Z. Liu J.K. Lenstra. Scheduling theory and its applications. Wiley, New York, Etats unis, 1995.

[Chu & Proth 96] C. Chu & J.M. Proth. L'ordonnancement et ses applications. Masson, Paris, France, 1996.

[Dantzig & Wolfe 60] G.B. Dantzig & P. Wolfe. *Decomposition principle for linear programs.* Operations Research, vol. 8, pages 101–111, 1960.

[Dauzère-Pérès & Sevaux 04] S. Dauzère-Pérès & M. Sevaux. *An exact method to minimize the number of tardy jobs in single machine scheduling.* Journal of Scheduling, vol. 9, no. 6, pages 405–420, 2004.

[Dauzère-Pérès 95] S. Dauzère-Pérès. *Minimizing late jobs in the general one machine scheduling problem.* European Journal of Operational Research, vol. 9, no. 81, pages 134–142, 1995.

[Davenport & Beck 00] A.J. Davenport & J.C. Beck. *A survey of techniques for scheduling with uncertainty*, http ://www.eil.utoronto.ca/chris/chris.papers.html, 2000.

[Davenport *et al.* 01] A.J. Davenport, C. Gefflot & J.C. Beck. *Slack-based Techniques for Robust Schedules.* Dans Six European Conference on Planning (ECP-2001), Toledo, Spain, 2001.

[Demmou 77] R. Demmou. *Etude de familles remarquables d'ordonnancements en vue d'une aide à la décision.* Thèse de troisième cycle, Université Paul Sabatier, Toulouse, France, 1977.

[Despontin-Monsarrat 04] E. Despontin-Monsarrat. *Aide à la décision pour une coopération inter-entreprises dans le cadre de la production la production à la commande.* Thèse de troisième cycle, Université Paul Sabatier, Toulouse, France, 2004.

[Dillenbourg *et al.* 96] P. Dillenbourg, M.J. Baker, A. Blaye & C. O'Malley. *The evolution of research on collaborative learning.* Dans Dans Learning in Humans and Machines : Towards an Interdisciplinary Learning Science, pages 189–211. P. Reimann and H. Spada (éds.), 1996.

[Dorigo 92] M. Dorigo. *Optimisation, Learning and Natural Algorithms.* Politechnico di milano, Université Paul Sabatier, 1992.

[Eisenberg 03] C. Eisenberg. *Distributed Constraint Satisfaction for coordinating and integrating a large-scale, heterogeneous entreprise.* Thèse de Doctorat, École Polytechnique Fédérale de Lausanne, Suisse, 2003.

[Erschler *et al.* 80] J. Erschler, F. Roubellat & J. Vernhes. *Characterizing the set of feasible sequences for n jobs to be carried out on a single machine.* European Journal of Operational Research, vol. 144, pages 1–11, 1980.

[Erschler *et al.* 83] J. Erschler, G. Fontan, C. Merce & F. Roubellat. *A New Dominance Concept in Scheduling n Jobs on a Single Machine with Ready Times and Due Dates.* Operations Research, vol. 31, no. 1, pages 114–127, 1983.

[Erschler *et al.* 85] J. Erschler, G. Fontan & C. Merce. *Un nouveau concept de dominance pour l'ordonnancement des travaux sur une machine.* RAIRO Recherche Opérationnelle/Operations Research, vol. 19, no. 1, pages 1–13, 1985.

[Erschler *et al.* 96] J. Erschler, M-J. Huguet & G. de Terssac. *Décision distribuée en gestion de production : Exploitation et régulation de l'autonomie.* Dans Dans Concepts et Outils pour les Systèmes de Production, pages 109–132. Cépaduès-Editions, 1996.

[Erschler 76] J. Erschler. *Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement.* Thèse de Doctorat d'Etat, Université Paul Sabatier, Toulouse, 1976.

[Erschler 96] J. Erschler. *Approches par contraintes pour l'aide à la décision et à la coopération.* Dans Dans Coopération et Conception, pages 137–147. Octares Editions, 1996.

[Esquirol & Lopez 99] P. Esquirol & P. Lopez. L'ordonnancement. Economica, 1999.

[Esswein *et al.* 05] C. Esswein, J.-C. Billaut & C. Artigues. *Une approche multi-critères pour un apport de flexibilité séquentielle.* Dans Dans Flexibilité et robustesse en ordonnancement sous la direction de J-C. Billaut et A. Moukrim et E. Sanlaville, pages 209–232. Hermès, 2005.

[Esswein 03] C. Esswein. *Un apport de flexibilité séquentielle pour l'ordonnancement robuste.* thèse de doctorat, Université François Rabelais, Tours, France, Décembre 2003.

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

[French 82] S. French. Sequencing and scheduling : An introduction to the mathematics of the job-shop. Ellis Horwood, Chichester, 1982.

[Gao *et al.* 95] H. Gao, M.S. Fox, W-Y. Chiang & S. Hikita. Building robust schedules - an empirical study of single machine scheduling with uncertainty. 1995.

[Glover 89] F. Glover. *Tabu search- part I.* ORSA Journal on computing, vol. 1, pages 190–206, 1989.

[Glover 90] F. Glover. *Tabu search- part II.* ORSA Journal on computing, vol. 2, pages 4–32, 1990.

[GOThA 02] GOThA. *Flexibilité et Robustesse en Ordonnancement.* Le bulletin de la ROADEF, vol. 8, pages 10–12, 2002.

[Graham *et al.* 79] R.L. Graham, E.L. Lawler, J.K. Lenstra & A.H.G. Rinnooy Kan. *Optimization and approximation in deterministic sequencing and scheduling : a survey.* Annals of Discrete Mathematics, vol. 5, pages 287–326, 1979.

[Hall & Posner 04] N.G. Hall & M.E. Posner. *Sensitivity analysis for scheduling problems.* Journal of Scheduling, vol. 7, pages 49–83, 2004.

[Hariri & Potts 83] A.M. Hariri & C.N. Potts. *An algorithm for single machine sequencing with release dates to minimize total weighted completion time.* Discrete Applied Mathematics, vol. 5, pages 99–109, 1983.

[Haspeslagh & Causmaecker 07] S. Haspeslagh & P. Causmaecker. *Distributed decision making in hospital wide nurse rostering problems.* Dans 3rd Multidisciplinary International Conference on Scheduling : Theory and Application, MISTA'07, pages 192–199, Paris, France, 2007.

[Hatchuel 96] A. Hatchuel. *Coopération et conception collective : Variété et crises des rapports de prescription.* Dans G. De Terssac and E. Friedberg (Eds.), Coopération et Conception, pages 101–121, Toulouse, 1996. Octarès Editions.

[Herroelen & Leus 02] W. Herroelen & R. Leus. *Project scheduling under uncertainty. Survey and Research Potentials.* Dans Eighth International Workshop on Project Management and Scheduling (PMS'2002), Valencia, Spain, 2002.

[Herroelen & Leus 04a] W. Herroelen & R. Leus. *The construction of stable project baseline schedules.* European Journal of Operational Research, vol. 156, pages 550–565, 2004.

[Herroelen & Leus 04b] W. Herroelen & R. Leus. *Robust and reactive project scheduling : a review and classification of procedures.* International Journal of Production Research, vol. 42, no. 8, pages 1599–1620, 2004.

[Holhaus & Rajendran 00] O. Holhaus & C. Rajendran. *Efficient jobshop dispatching rules : further developments.* Production Planning & Control, vol. 11, no. 2, pages 171–178, 2000.

[Holland 75] J.H. Holland. *Adaptation in natural and artificial systems.* Rapport technique, Ann Arbor, University of Michigan, 1975.

[Homberger 07] J. Homberger. *A multi-agent system for the decentralized ressource-constrained multi-projet scheduling problem.* International Transactions in Operational Research, vol. 14, pages 565–589, July 2007.

[Huguet 94] M-J. Huguet. *Approche par contarintes pour l'aide à la décision et à la coopération en gestion de production.* Thèse de Doctorat, Institu National des Sciences Appliquée, Toulouse, France, Décembre 1994.

[Johnson *et al.* 00] E. L. Johnson, G. L. Nemhauser & M.W.P. Savelsbergh. *Progress in linear programming-base algorithms for integer programming : an exposition.* INFORMS Journal on Computing, vol. 12, no. 1, pages 2–23, 2000.

[Johnson 54] S.M. Johnson. *Optimal two and three stage production schedules with set-up times included.* Naval research Logistics Quaterly, vol. 1, pages 61–68, 1954.

[Kaplansky & Meisels 07] E. Kaplansky & A. Meisels. *Distributed personnel scheduling negotiation among scheduling agents.* Annals of Operations Research, vol. 155, no. 1, pages 227–255, 2007.

[Karmarkar 84] N. Karmarkar. *A new polynomial time algorithm for linear programming.* Combinatorica, vol. 4, pages 373–395, 1984.

[Khachian 79] L.G. Khachian. *A polynomial algorithm in linear programming.* Soviet Mathematics Doklady, vol. 20, pages 191–194, 1979.

[Kirkpatrick *et al.* 83] C. Kirkpatrick, C.D. Gellat & M.P. Vecchi. *Optimization by simulated annealing.* Science, vol. 220, page 671680, 1983.

[Kise *et al.* 78] H. Kise, T. Ibaraki & H. Mine H. *A solvable case of the one-machine scheduling problem with ready and due times.* Operations Research, vol. 26, pages 121–126, 1978.

[Klee & Minty 72] V. Klee & G. J. Minty. *How good is the simplex algorithm.* Dans DAnd Shisha Os, editeur, Inequalities III, pages 159–175. Academic Press, New-York, 1972.

[Kouvelis & Yu 97] P. Kouvelis & G. Yu. Robust discrete optimisation and its applications. Kluwer Academic, 1997.

[La 04] H.T. La. *Utilisation d'ordres partiels pour la caractérisation de solutions robustes en ordonnnancement.* Thèse de doctorat, INSA, Toulouse, France, 2004.

[Lawler 82] E.L. Lawler. *Scheduling a single machine to minimize the number of late jobs.* Rapport technique, Preprint. Computer Science Division, Univ. of California Berkley, 1982.

[Lawler 90] E.L. Lawler. *A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs.* Annals of Operations Research, vol. 26, pages 125–133, 1990.

[Le Pape 91] C. Le Pape. *Constraint propagation in Planning and Scheduling.* Rapport technique, Robotics Laboratory, Department of Computer Science, Stanford, 1991.

[Lenstra *et al.* 77] J.K. Lenstra, A.H.G. Rinnooy Kan & P. Brucker. *Complexity of machine scheduling problems.* Annals of Discrete Mathematics, vol. 1, pages 343–362, 1977.

[Leon *et al.* 94] V.J. Leon, S.D. Wu & R.H. Storer. *Robustness measures and robust scheduling for job shops.* IIE Transactions, vol. 26, no. 5, pages 32–43, 1994.

[Leus 03] R. Leus. *The generation of stable projects plans complexity and exact algorithms.* thèse de doctorat, Université de Metz, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium, 2003.

[Levy 96] M-L. Levy. *Méthodes par décomposition temporelle et problèmes d'ordonnancement.* Thèse de Doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, Mars 1996.

[Lin *et al.* 08] F.r. Lin, H. c. Kuo & S m. Lin. *The enhancement of solving the distributed constraint satisfaction probleme for cooperative supply chains using multi-agent systèmes.* Decision Support System, vol. 45, no. 4, page 795810, 2008.

[Lopez & Roubelat 01] P. Lopez & F. Roubelat. Ordonnancement de la production. Hermes, Paris, France, 2001.

[Lopez *et al.* 96] P. Lopez, L. Hodaut & P. Esquirol. *Coopération homme-système en ordonnancement de production.* Dans Dans Concepts et Outils pour les Systèmes de Production, pages 161–178. Cépaduès-Editions, 1996.

[Lopez 91] P. Lopez. *Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources.* thèse de doctorat, Université Paul Sabatier, Toulouse, France, 1991.

[McMahon & Florian 75] G.B. McMahon & M. Florian. *On scheduling with ready times and due dates to minimize maximum lateness.* Operations Research, vol. 23, pages 475–482, 1975.

[Mebarki *et al.* 96] N. Mebarki, H. Pierreval & K. Kouiss. *Une approcha multi-agents pour l'ordonnancement dynamique d'un système de production flexible.* Ingénierie des Systèmes d'Information, Hermés, vol. 4, no. 5, pages 621–636, 1996.

[M'Hallah & Bulfin 07] R. M'Hallah & R.L. Bulfin. *Minimizing the weighted number of lardy jobs on a single machine with release dates.* European Journal of Operational Research, vol. 176, no. 1, pages 727–744, 2007.

[Minoux 08] M. Minoux. programmation mathématique : Théorie et algorithmes. 2 ème édition Lavoisier, 2008.

[Monma 82] C.L. Monma. *Linear time algorithms for scheduling on parallel processors.* Operations Research, vol. 30, pages 116–124, 1982.

[Monteiro 01] T. Monteiro. *Conduite distribuée d'une coopération entre entreprises : le cas de la relation donneurs d'ordres - fournisseurs.* Thèse de Doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, 2001.

[Moore 68] J.M. Moore. *An n job, one machine sequencing algorithm for minimizing the number of late jobs.* Management Science, vol. 15, no. 1, pages 102–109, 1968.

[Mortan & Pintico 93] T.E. Mortan & D. Pintico. Heuristics scheduling systems. Wiley, New York, Etats-Unis, 1993.

[Moyaux *et al.* 03] T. Moyaux, B.Chaib-draa & S.D'Amours. *Satisfaction distribuée de contraintes et son application à la génération d'un emploi du temps d'employés.* Dans 5ème Congères International de Génie Industriel (CIGI03), Quebec, 2003.

[Muscettola 02] N. Muscettola. *Computing the Envelope for Stepwise-Constant Resource Allocations.* Dans 8th International Conference in Principles and Practice of Constraint Programming, Springer, 2002.

[Noronha & Sarma 91] S.J. Noronha & V.V.S. Sarma. *Knowledge-based approach for scheduling problems : A survey.* IEEE transaction on knowledge and data engineering, vol. 3, no. 2, pages 160–171, 1991.

[Ourari & Bouzouia 03] S. Ourari & B. Bouzouia. *An Approach Based On Operation Insertion For One-Machine Real-Time Scheduling.* International Journal of Robotic and Automation, vol. 18, no. 4, pages 185–190, 2003.

[Ourari & Briand 08] S. Ourari & C. Briand. *Conditions de dominance pour le problème à une machine avec minimisation des travaux en retard.* Dans 9ème

congrès de la société française de Recherche Opérationnelle et d'aide à la décision. (ROADEF08), Clermont-Ferrand, France, 2008.

[Ourari *et al.* 01] S. Ourari, B. Bouzouia & B. Bakalem. *Single machine scheduling problem with random events.* Dans 10th IFAC Symposium on Automation in Mining, Mineral and Metal Processing (IFAC MMM'01), Tokyo, Japan, 2001.

[Ourari *et al.* 07] S. Ourari, C. Briand & B. Bouzouia. *Concept de Dominance pour l'Ordonnancement sur une Machine avec Minimisation des Travaux en Retard.* Dans CIGI07, Trois-rivières, Canada, 2007.

[Ourari *et al.* 09a] S. Ourari, C. Briand & B. Bouzouia. *A mathematical programming approach to minimize the number of late jobs for the single machine scheduling problem.* in revision for the Journal of Mathematical Modelling and Algorithms, may 2009.

[Ourari *et al.* 09b] S. Ourari, C. Briand & B. Bouzouia. *Minimizing the number of tardy jobs in single machine scheduling using MIP.* Dans MISTA09, Dublin, Irland, 2009.

[P. Baptiste & Peridy 98] C. Le Pape P. Baptiste & L. Peridy. *Global constraints for partials csps : a case study of ressource and due date constraints.* In springer-Verlag editor, Proceeding of the Fourth International Conference on Principles and Practice of Constraint Programming, pages 87–101, 1998.

[Pendharkar 07] Parag C. Pendharkar. *The teory and experiments of designing cooperative intelligent systems.* Decision Support Systems, vol. 43, no. 3, pages 1014–1030, 2007.

[Pierrevall & Mebarki 97] H. Pierrevall & M. Mebarki. *Dynamic selection of dispatching rules for manufacturing system scheduling.* International Journal of Production Research, vol. 35, no. 6, pages 1575–1591, 1997.

[Pinedo 08] M. Pinedo. Scheduling : Theory, algorithms and systems (2nd edition). Prentice Hall, 2008.

[Policella 05] N. Policella. *Scheduling with uncertainty. A proactive approach using partial order schedules.* Thèse de doctorat, Université de Rome, Sapienza, Italie, 2005.

[Portmann & Mouloua 07] M-C. Portmann & Z. Mouloua. *A window time negotiation approach at the scheduling level inside supply chains.* Dans 3rd Multidisciplinary International Conference on Scheduling : Theory and Application, MISTA'07, pages 410–417, Paris, France, 2007.

[Priore *et al.* 98] P. Priore, D.D. Garcia & I.F. Quesada. *Manufacturing systems scheduling through machine learning.* Dans Neural Computation, NC'98, pages 914–917, Vienna, Austria, 1998.

[Priore *et al.* 01] P. Priore, De la Fuente, A. Gomez & J. Puente. *A review of machine learning in dynamic scheduling of flexible manufacturing systems.* Dans AI EDAM Artificial Intelligence for Engineering Design Analysis and Manufacturing, pages 251–263, Cambrige University Press, USA, 2001.

[Pujo & Brun-Picard 02] P. Pujo & D. Brun-Picard. *Pilotage sans plan prévisionnel ni ordonnancement préalable.* Dans Méthodes du pilotage des systèmes de production sous la direction de P. Pujo et J-P. Kiefer, pages 129–162. Hèrmes, 2002.

[Ramasesh 90] R. Ramasesh. *Dynamic job shop scheduling : a survey of simulation research.* OMEGA, vol. 18, pages 43–57, 90.

[Roy & Sussmann 64] B. Roy & B. Sussmann. *Les problèmes d'ordonnancement avec contraintes disjonctives.* Dans Technical report D.S. Nř 9 bis SEMA, Montrouge, 1964.

[Roy 70] B. Roy. Algèbre moderne et théorie des graphes, volume 2. Dunod, Paris, 1970.

[Sabuncuoglu & Bayiz 09] I. Sabuncuoglu & M. Bayiz. *Analyse of reactive scheduling problems in a job shop environment.* International Journal of Computer Integrated Manufacturing, vol. 22, no. 2, pages 138–157, 2009.

[Sabuncuoglu & Goren 00] I. Sabuncuoglu & S. Goren. *Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research.* European Journal of Operational Research, vol. 126, pages 567–586, 2000.

[Sadeh *et al.* 93] E. Sadeh, S. Otsuka & R. Schelback. *Predictive and reactive Scheduling with the MicroBoss Production Scheduling and Control System.* Dans Workshop on knowledge-Based Production Planning, Scheduling and Control, pages 293–306, 1993.

[Sadki 08] A. Sadki. *Minimisation du nombre de travaux en retard sous contraintes de fenêtres d'exécution.* Master, LAAS, Toulouse, 2008.

[Sakkout *et al.* 98] El Sakkout, M.G. Wallace & E.T. Richards. *Minimal Perturbation in Dynamic Scheduling.* Dans 13th European Conference on Artificial Intelligence (ECAI-98), Brighton, UK, 1998.

[Sanlaville 05] E. Sanlaville. *Ordonnancement sous conditions changeantes : Comment prendre en compte les variations, aléas, incertitudes sur les données ?*, http ://mathinfo.univ-bpclermont.fr/sanlavil/hdr_sanlaville.pdf, 2005.

[Sardas *et al.* 02] J.C. Sardas, J. Ershler & G. De Terssac. *Coopération et organisation de l'action collective.* Dans Coopération et connaissance dans les

systèmes industriels sous la direction de R. Soënen et J. Perrin, pages 69–90. Hermès Science, 2002.

[Sevaux & Sörensen 02] M. Sevaux & K. Sörensen. *A genetic algorithm for robust schedules in a just-in-time environment.* Rapport technique, LAMIH/SP-2003-1, University of valenciennes, 2002.

[Sevaux & Sörensen 04] M. Sevaux & K. Sörensen. *A genetic algorithm for robust schedules in a just-in-time environment.* 4OR - Quaterly journal of the Belgian, French and Italian Operations Research Societies, vol. 2, no. 2, pages 129–147, 2004.

[Sevaux 98] M. Sevaux. *Étude de deux problèmes d'optimisation en planification et ordonnancement.* Thèse de doctorat, Université Paris VI, France, Décembre 1998.

[Sevaux 04] M. Sevaux. *Métaheuristiques stratégies pour l'optimisation de la production de biens et services.* Hdr, Université Valencienne et du Hainaut Cambrésis, France, Juillet 2004.

[Shen 02] W. Shen. *Distributed Manufacturing Scheduling Using Intelligent Agents.* IEEE Intelligent Systems, vol. 17, no. 1, pages 88–94, 2002.

[Telle 03] O. Telle. *Gestion des chaînes logistiques dans le domaine aéronautique : Aide à la coopération au sein d'une relation donneur d'ordre/ Fournisseur.* Thèse de doctorat, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, Janvier 2003.

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

[Thomassen & Lorenzen 00] M.A. Thomassen & M. Lorenzen. *The dynamic costs of coordination and specialization.* Dans DRUID Nelson and Winter Conference, Aalborg, Danemark, Juin 2000.

[Touati 08] N. Touati. *Amélioration des performances du schéma de la génération de colonnes : Application aux problèmes de tournées de véhicules.* Thèse de doctorat, Université Paris 13, France, 2008.

[Tranvouez 01] E. Tranvouez. *IAD et ordonnancement : Une approche coopérative de réordonnancement par système multi-agents.* Thèse de doctorat, Université d'Aix-Marseille III, France, Mai 2001.

[Urbani 06] D. Urbani. *Elaboration d'une approche hybride SMA-SIG pour la définition d'un système d'aide à la décision ; application à la gestion de l'eau.* Thèse de doctorat, Université de Corse, France, Novembre 2006.

[Wu *et al.* 93] S.D. Wu, R.H. Store & P.C. Chang. *One-machine rescheduling heuristics with efficiency and stability as criteria.* Computer and Operations Research, vol. 20, pages 1–14, 1993.

De l'ordonnancement déterministe à l'ordonnancement distribué sous incertitudes

Résumé : Ce travail présente l'étude de deux types de problèmes d'ordonnancement. Le premier concerne la résolution centralisée et exacte d'un problème à une machine, le second, la résolution distribuée et coopérative d'un problème job shop où chaque machine est assimilée à un acteur possédant sa propre autonomie décisionnelle. Pour ces deux problèmes, des conditions de dominance sont utilisées, dans le premier cas, dans le but de limiter la complexité algorithmique liée à la recherche de solutions admissibles ou optimales, dans le deuxième cas, pour accroître la capacité de chaque acteur à résister aux incertitudes liées aux fluctuations de son environnement. Dans un premier temps, un théorème proposé dans les années quatre-vingt est rappelé, qui, considérant le problème à une machine, permet de caractériser un ensemble de solutions dominantes. Sur la base de ce théorème, nous proposons ensuite de nouvelles conditions analytiques et numériques de dominance permettant de restreindre encore davantage l'ensemble des solutions dominantes. En exploitant ces résultats, des formulations mathématiques originales et efficaces sont présentées, sous forme de programmes linéaires en nombres entiers, pour la modélisation et la résolution du problème à une machine en s'intéressant tour à tour au critère de minimisation du plus grand retard algébrique, puis à celui de minimisation du nombre de travaux en retard. Dans un deuxième temps, nous étudions le problème d'ordonnancement job shop dans un environnement multi-acteur, chaque acteur gérant l'activité d'une machine. Tenant compte de l'autonomie de décision et des objectifs propres de chacun, l'ordonnancement est envisagé sous la forme d'une fonction distribuée où la solution globale résulte d'une coopération entre les différents acteurs, cette solution pouvant évoluer dans le temps au fur-et-à-mesure des prises de décision locales. Ainsi, chaque acteur construisant localement sa propre organisation et n' n'ayant qu'une connaissance partielle et incertaine de l'organisation des autres, nous proposons que les organisations locales soient construites de façon robuste. Pour cela nous montrons comment, à l'aide des résultats de dominance, maintenir au niveau de chaque acteur un ensemble dominant de solutions ayant une performance au pire bornée. Une nouvelle approche d'ordonnancement est ensuite proposée où les acteurs négocient deux à deux, de façon distribuée, de façon à converger progressivement vers des décisions assurant un compromis satisfaisant entre l'optimisation des objectifs locaux et des objectifs globaux.

Mots clés : *Ordonnancement, condition de dominance, une machine, PLNE, job shop, coopération, robustesse, algorithme distribué.*

From deterministic scheduling to distributed scheduling with uncertainties

Abstract : This work presents the study of two scheduling problems. The former concerns the exact and centralised resolution of a single machine problem, and the latter, the distributed and cooperative resolution of a job shop, each machine being viewed as an actor having its own decision autonomy. For both problems, dominance conditions are used, in the first case, in order to reduce the algorithmic complexity for seeking feasible or optimal solutions, and in the second case, to increase the ability of each actor to face uncertainties. In the first part, a theorem, stated in the early eighties, is recalled that allows to characterize a set of dominant solutions, considering a one-machine sequencing problem. On the basis of the theorem, new analytical and numerical dominance conditions are established that allow to tighten the set of dominant sequences. Then original and efficient mathematical formulations, in the form of integer linear programs, are proposed for modelling and solving single machine problems. Two kinds of criterion are considered : the minimization of the maximum lateness and the minimization of the number of tardy jobs. In the second part, the job shop scheduling problem is studied, using a multi-actor framework, assuming that each actor manages one machine. Taking into account the decisional autonomy and the own objectives of each actor, scheduling is seen as a distributed and dynamic function, where the global solution emerges from negotiations among the actors. We assume that each actor builds up its own local organisation in a robust way, having an imprecise and partial knowledge of the other actor's organisation. We particularly show how maintaining on each actor a set of dominant job sequences so that the worst performance can be bounded. Then a new scheduling approach is sketched where actors initiate point-to-point negotiation, in a distributed way, so as to progressively converge toward trade-off decisions that balance local and global objectives.

Keywords : *scheduling, dominance condition, single machine, MILP, job shop, cooperation, robustness, distributed algorithm.*