Ministère de l'enseignement supérieur et de la recherche scientifique
Ecole Nationale Polytechnique
Département du génie Eléctrique

En vue de l'obtention du diplôme
de Magistère en Automatique
Option : Robotique et Productique

Thème :

Approche par coopération inter machine pour un ordonnancement distribué

Réalisé par :

DJELLOULI sabrina

Proposé et dirigé par :

M.BAKALEM

M.TADJINE

Année Universitaire 2006/2007

A mes parents,

à mon époux,

car il faut savoir choisir ses gros cailloux

Remerciements

Mes remerciements les plus sincères ;

A monsieur le Professeur BOUDJEMA Fares, pour avoir accepté de présider mon jury.

A monsieur HEMICI Boualem, et Monsieur SAARI Hamoud qui ont accepté de participer à la valorisation de ce travail.

A Mademoiselle OURARI Samia, pour le soutien qu'elle m'a apportée pour la réalisation de ce modeste travail, ainsi que pour ses conseils précieux.

A mes deux promoteurs M. BAKALEM, et M. TADJINE, pour la totale confiance qu'ils m'ont accordée.

A Monsieur BOUZOUIA Mohamed, Directeur de recherche au centre de developpement des technologies avancées (CDTA) d'Alger pour avoir accepter de m'acceuillir au sein de son laboratoire.

A Monsieur KOUIDER Ahmed chercheur au centre de developpement des technologies avancées (CDTA), pour son aide précieuse.

A toutes les personnes qui malgrès tout, ont su m'apprecier à ma juste valeur : A Monsieur OULD-ALI Hamid responsable de recherche et developpement de OneAccess (Paris), A monsieur CHABAANE El-Hachemi Gérant de VMD-software (Alger), A VMD-Software...

Merci à tous.

Table des matières

Table des figures

Liste des tableaux

Introduction générale et problématique

L'ordonnancement joue un rôle essentiel dans de nombreux secteurs d'activités : la conception (de bâtiments, de produits, de systèmes,...), l'administration (gestion d'emplois du temps, gestion du personnel), l'industrie (gestion de production), l'informatique (ordonnancement de processus, ordonnancement de réseaux). Il s'agit de définir la localisation temporelle des tâches de sorte à respecter, d'une part, les contraintes de précédence liées aux gammes opératoires, d'autre part, les contraintes de disjonction (une même ressource ne pouvant réaliser simultanément deux tâches).

Les méthodes d'ordonnancement foisonnent dans la littérature. Elles se différencient par la nature du problème considéré (nombre de ressources, structure particulière du problème), la nature des contraintes prises en compte, les objectifs à satisfaire (minimisation des coûts, de la durée totale de mise en oeuvre) et la nature de l'approche de résolution adoptée (heuristiques, méthodes exactes, méta heuristiques, approches par contraintes).

La plupart de ces méthodes opèrent sous l'hypothèse classique que les données du problème d'ordonnancement sont parfaitement connues : l'ensemble des tâches à ordonnancer est fini et déterminé a priori, les durées opératoires, les dates de début au plus tôt et de fin au plus tard des tâches sont fixées et stables et les capacités des ressources sont connues à tout instant. Cette hypothèse permet en effet de traiter un problème de façon déterministe, ce qui simplifie considérablement la résolution, celle-ci demeurant toutefois ardue dans la plupart des cas.

Hors, compte tenu du caractère incertain de l'environnement d'une entreprise, la probabilité qu'un ordonnancement prévisionnel soit exécuté comme prévu est faible [Ess03]. En effet, presque toutes les données déterministes associées à un problème d'ordonnancement sont en réalité sujettes à fluctuation. Parmi les sources d'incertitudes citons : les durées d'exécution des tâches variables, les arrivées aléatoires de nouvelles tâches, les modifications de dates de disponibilité et de dates échues, les indisponibilités de ressources.

Ce constat a justifié l'émergence d'une nouvelle thématique de recherche appelée ordonnancement robuste. Il s'agit toujours d'organiser dans le temps l'exécution de tâches soumises à des contraintes de temps et de ressources, tout en satisfaisant au mieux un ou plusieurs objectifs, mais en s'interdisant cette fois de négliger le caractère perturbé de l'environnement de mise en oeuvre. La solution produite doit alors non seulement résister aux perturbations de l'environnement, c'est à dire être toujours utilisable en cas d'imprévus, mais aussi posséder une qualité stable face à ces perturbations (solution flexible).

De façon classique, la résolution en-ligne ou hors-ligne [Ess03] de problèmes ordonnancement avec prise en compte des incertitudes est classiquement assimilée à un problème de décision global car la fonction ordonnancement gère l'organisation de la totalité des ressources, chacune devant respecter la solution établie. 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 d'une autonomie de décision dont il est fondamental de tenir compte. Il est alors nécessaire d'adopter une démarche coopérative entre les différents acteurs de manière à converger vers un compromis satisfaisant les exigences de performance globale et celles de performance locale.

Le champ d'application de l'ordonnancement considéré dans ce manuscrit est celui des systèmes de production automatisés. Le problème job shop à plusieurs machines, noté $J_n||C_{max}$, consiste à ordonnancer un ensemble T de travaux sur un ensemble M de m machines $M = M_1, \cdots M_m$. Chaque travail est caractérisé par une gamme opératoire et est constitué d'un ensemble de tâches devant chacune s'exécuter sur une des m machines dans l'ordre défini par la gamme. On note que chaque tâche i est caractérisée par une durée opératoire p_i et est réalisée par une et une seule ressource disponible en un exemplaire unique.

L'objectif de l'heuristique présentée dans ce travail, est de proposer une approche se basant sur la coopération inter-machines dans le but de distribuer le pouvoir décisionnel entre elles de manière homogène (sans faire intervenir aucune entité superviseuse), et de réaliser un ordonnancement distribué dans le cadre de la minimisation de la durée totale d'exécution, appelée makespan et notée C_{max}.

Ce problème étant définit NP-difficile [LK78], il reste cependant possible de le décomposer en m sous problèmes à une machine interdépendants, où chaque tâche est caractérisée par une fenêtre d'exécution $[r_i, d_i]$, pour lesquels on cherche à minimiser le retard algébrique (problème noté $1|r_i|L_{max}$).

Chaque machine sera assimilée à un centre de décision, sur lequel un ordonnancement local est calculé indépendament du reste des machines. Le problème $1|r_i|L_{max}$ est résolu en utilisant le théorème des pyramides qui permet de définir un ensemble de séquences dominantes, et donc de la fléxibilité séquentielle.

L'ensemble des machines représentera alors, un réseau de centres de décisions interdépendantes , les décisions d'un centre étant contraintes par celles des centres situés en amont. Nous faisons l'hypothèse que la coopération entre centre de décision est organisée de façon distribuée et que le pouvoir décisionnel y est réparti de façon homogène. La solution globale résulterait donc, d'une coopération entre les différents centres de décisions, chacun gérant son propre ordonnancement robuste local.

Pour faire coopérer les différents centres décisionnels, et de proposer un protocole de négociation, nous nous sommes inspirés de la loigique multi-agent. La solution à laquelle on aboutit devrait être cohérente et doit jouir d'une certaine flexibilité afin de permettre la prise en charge des jobs arrivant en temps réel. Pour ce, nous allons dé-

finir les critères qui permettent de classer une solution en solution acceptable ou pas [BS04, Ess03, BLE03, DB00].

Le présent manuscrit développe les principes évoqués précédemment concernant la définition et la mise en uvre par une approche coopérative pour un ordonnancement distribué de l'atelier. Ce manuscrit est organisé, en quatre chapitres dont une brève description est donnée ci-dessous :

Le premier chapitre,présente une revue de la littérature sur les domaines de la gestion de la production, de l'ordonnancement, et de la coopération ainsi que sur les systèmes développés à la suite du rapprochement de ces deux derniers domaines. Nous étudions dans un premier temps les principaux concepts relatifs à ces domaines de recherche, pour nous intéresser ensuite aux principaux travaux qui ont en découlé notament en dans le contexte de l'ordonnancement distribué.

Dans le deuxième chapitre, il sera question de décrire le théorème de dominance qui permet de définir un ensemble flexible de solutions pour résoudre le problème $1|r_i|L_{max}$. Ce deuxième chapitre propose aussi une méthode pour le calcul des marges libres présentes dans la séquence d'execution d'une tâche donnée, dans le but d'amortir l'impact de l'insertion d'une nouvelle tâche arrivant en temps réel.

Dans le troisième chapitre, on présentera une approche de résolution par coopération inter machine pour la résolution du problème job Shop à plusieurs machines avec minimisation du C_{max}. L'heuristique developée se base sur un protocole de négociation entre les différentes ressources assimilées à des centres de décision. Pour finir, les résultats de l'implémentation de l'approche pour le problème $J_n||C_{max}$ avec n=4 machines, sont présentés
.

Le quatrième chapitre illustre la mise en uvre de la méthode d'ordonnancement coopératif et distribué en simulant un cas pratique traitant du problème $J_4||C_{max}$,et en utilisant l'heuristique developpée tout au long de ce travail. Nous décrivons tout d'abord le modèle du centre décisionnel retenu et son architecture en détaillant les différents modules qui le composent et l'intéraction qui existe entre eux, Nous précisons ensuite larchitecture informatique du logiciel developpé, nous décrirons enfin le cas détude retenu pour effectuer une simulation du protocole de négociation. Nous terminons ce chapitre en présentant et en analysant les résultats obtenus par cette simulation.

Enfin, une conclusion clôturera le présent travail, en établissant un bilan des travaux présentés dans les chapitres précédents et en ouvrant un certain nombre de perspectives de recherche.

Chapitre 1

Généralités et état de l'art

Après le rappel de quelques concepts de bases sur les systèmes de gestion et de la fonction ordonnancement, ce chapitre s'intéresse, entre autre à la gestion des incertitudes en ordonnancement et en particulier, à l'ordonnancement distribué robuste. Il décrit plus précisément les sources et les types d'incertitudes à prendre en compte et comment celles-ci sont généralement modélisées, puis gérées. Un état de l'art et une synthèse des différentes approches d'ordonnancement robustes est présenté. Pour finir, nous évoquerons les concepts relatifs à l'ordonnancement distribué et à la coopération.

1.1 Production et gestion de la production : Concepts de base

1.1.1 La production

La production est une activité opérationnelle utilisant des ressources acquises (matériels et humains) en vue de créer des biens matériels ou d'assurer des services d'utilité moyennant une ou plusieurs transformations. Ces transformations sont caractérisées par un ensemble d'opérations, qui peuvent consister, soit dans une modification de la forme des pièces, soit dans la combinaison de plusieurs pièces. L'ensemble de ces transformations constituant la production seront soumises à des critères impératifs de coût, délai, quantité et qualité [Bén98].

1.1.2 La gestion de production

La gestion de production a pour objet la recherche d'une organisation efficace dans l'espace et dans le temps, de toutes les activités relatives à la production, afin d'atteindre les objectifs de l'entreprise [Gia88].

Pour Javel, [Jav97] et Dupont [Dup97], la gestion de production consiste à produire en temps voulu, les quantités demandées par les clients dans des conditions de coût de revient et de qualité déterminées en optimisant les ressources de l'entreprise de façon à assurer sa pérennité, son développement et sa compétitivité.

1.1.3 Le système de production

Un système de production est un ensemble de ressources réalisant une activité de production. Une des ses fonctions importantes est la fabrication elle-même du produit fini, mais son bon déroulement nécessite la mise en oeuvre de fonctions additionnelles telles que les achats des composants et matières premières, la distribution du produit fini, la gestion de la qualité des composants et du produit, ainsi que la maintenance des différentes ressources.

Dans la littérature, différentes décompositions du système de production sont proposées. On en retient celle proposée par Doumeinghts et al., dans [Dou84], et reprise par [Our01].

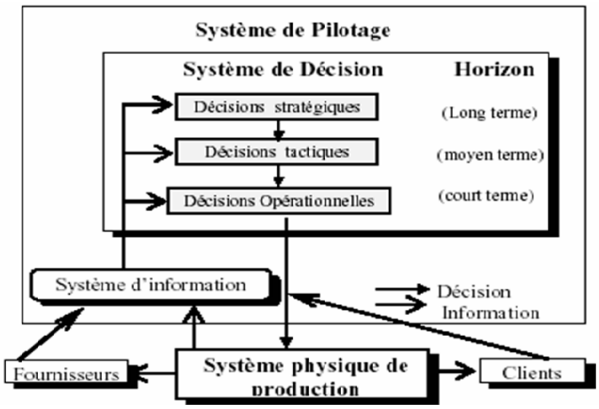


FIG. 1.1 – Vision systémique d'un système de production [Our01]

Comme le représente la figure 1.1, le système physique de production transforme les matières premières en produits finis. Il est constitué de ressources humaines et physiques. Ses activités sont déclenchées et vérifiées par le système de gestion de production .

Le système de pilotage est destiné à commander le système physique de production, tandis que le système de décision le contrôle.

Le système d'information collecte, stocke, traite et transmet les informations. Il intervient comme une interface entre les systèmes de décision et de production. Il intervient aussi à l'intérieur même du système de décision, pour la gestion des informations utilisées lors de prises de décisions, et à l'intérieur du système physique de production, pour la création et le stockage des informations de suivi.

L'association des parties des systèmes de décision et d'information concernant uniquement la production, constitue le système de gestion de production.

1.1.4 Relation de la gestion de production avec les fonctions de l'entreprise

La gestion de production présente des relations avec plusieurs fonctions différentes au sein de l'entreprise. On peut citer : la planification, les approvisionnements, les achats, et la gestion des ressources humaines et techniques. L'une des méthodes de gestion de production permettant de prendre en compte la complexité des produits (nomenclatures) et des organisations des industries manufacturières est la MRP (Manufacturing Resource Planning : Planification des Ressources de Production).

Cette méthode présente l'avantage de respecter notamment la forte hiérarchisation des prises de décision dans les entreprises. Les différentes étapes qui composent la MRP sont présentées sur la figure 1.2.

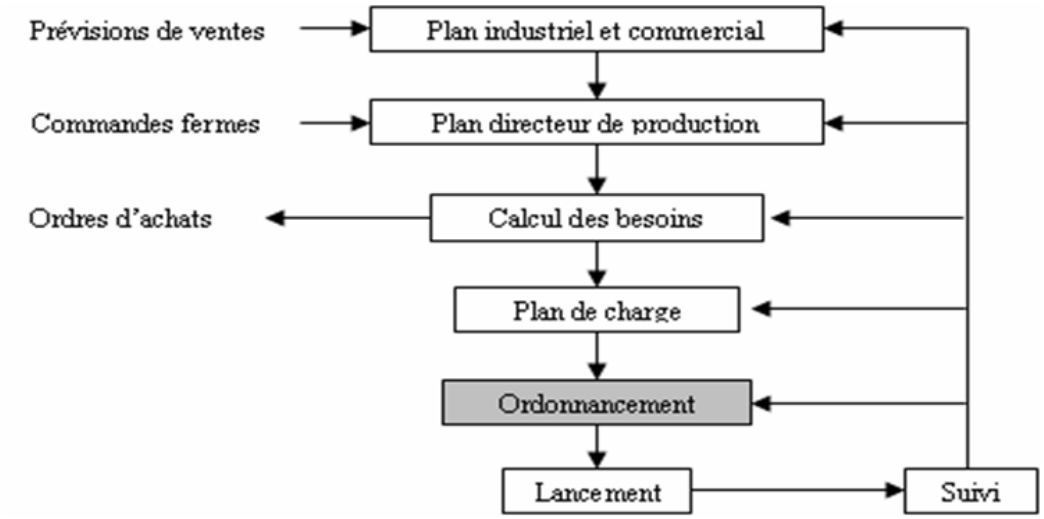


FIG. 1.2 – Les fonctions de la MRP [Let01]

Au sommet de la MRP, le plan industriel et commercial (PIC) intègre mois par mois et par famille de produits les prévisions de vente des produits et les objectifs de production, sur un horizon assez long, de l'ordre d'une année. Les prévisions commerciales du PIC sont ensuite détaillées sur chaque produit et réévaluées sur un horizon plus court (de l'ordre de quelques mois) dans le plan directeur de production (PDP). Celui-ci recense toutes les commandes fermes et prévisionnelles et en déduit un échéancier des quantités à fabriquer par produit et par période. On détermine à ce niveau, si les capacités globales de l'entreprise en hommes et en machines critiques vont être suffisantes ou s'il est nécessaire d'investir dans de nouveaux matériels ou d'embaucher davantage de personnel.

A partir de cet échéancier, le calcul des besoins détermine les quantités de chacun des composants des produits à fabriquer et les dates de fabrication, en se basant sur la nomenclature des produits et les délais de fabrication. A l'issue du calcul des besoins, on dispose des quantités de composants élémentaires à fabriquer (ordres de fabrication) et de matières premières et de composants à acheter (ordres d'achat), accompagnées de

leurs dates de besoin. Le jalonnement de ces ordres de fabrication permet de déterminer la charge par machine et par période. On peut donc vérifier si les commandes sont réalisables ou pas, et prendre d'éventuelles mesures correctives telles que des heures supplémentaires pour les opérateurs, la sous-traitance d'une partie du travail ou le lissage de la charge. Lorsque l'adéquation charge/capacité est faite, les opérations sont ordonnancées [Let01].

L'ordonnancement correspond à l'affectation des opérations sur les ressources [1]. Cette fonction de gestion de production étant l'objet principal de notre étude, elle sera détaillée dans la section suivante.

1.2 Présentation des problèmes d'ordonnancement

1.2.1 Définition

On rencontre des problèmes d'ordonnancement dans de très nombreux types de systèmes dès lors qu'on est emmené à organiser des tâches à réaliser, à répartir des ressources, à planifier l'exécution des différentes parties élémentaires d'un projet complexe. Depuis une cinquantaine d'années, les études traitant des problèmes d'ordonnancement ont été très nombreuses, que ce soit concernant les problèmes académiques ou bien des problèmes liés à des applications dans le domaine de la production, dans les systèmes informatiques, dans la gestion des emplois du temps ou encore la planification des ressources humaines. La littérature est maintenant si dense qu'on trouve de nombreuses définitions du terme ordonnancement. Carlier et Chrétienne donnent, dans [CC88], la définition suivante :

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

Plusieurs autres définitions peuvent être citées :

Le problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, ...) et de contraintes portant sur l'utilisation et la disponibilité de ressources requises [EL99].

Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début [GOT93].

Dans le domaine de la gestion de la production, la fonction ordonnancement a pour rôle d'organiser dans le temps l'exécution des opérations sur les machines et de réagir à tout changement non prévu tel que, panne des machines, manque de matières, ordres de fabrication non prévus, et ceci afin de réaliser le plan de production.

Le point de vue *temps réel* se traduit par l'acquisition des informations sur le déroulement de la production et la réaction aux aléas par la modification du programme de

[1] Ressources techniques en général, mais de plus en plus on se soucie de l'affectation des ressources humaines.

production résultant de l'opération d'ordonnancement.

1.2.2 Les éléments fondamentaux d'un problème d'ordonnancement

Le problème d'ordonnancement est défini en se basant sur les concepts de tâche, ressource, contrainte et objectif :

(a) La tâche

Une tâche i, est une entité élémentaire de travail, localisée dans le temps par une date de disponibilité r_i (ready time ou release date) avant laquelle l'exécution de la tâche i ne peut débuter, une date d'achèvement dite date échue (due date) notée d_i, et une durée opératoire (processing time) notée p_i. On note $[r_i\ d_i]$, l'intervalle temporel dans lequel la tâche i doit s'exécuter.

La réalisation de la tâche i nécessite l'utilisation d'une ou plusieurs ressources $k \in R$, où R est l'ensemble de ressources. Les ressources concernées par la réalisation de la tâche sont généralement supposées connues a priori.

Si les tâches peuvent être exécutées par morceaux, alors le problème est dit *préemptif*. Dans ce cas la durée p_i d'une tâche ne peut être déterminée qu'une fois l'ordonnancement construit, sinon il est dit *non préemptif* .

(b) La ressource

Une ressource k est un moyen technique ou humain, disponible en quantité limitée, destinée à être utilisé pour la réalisation de plusieurs tâches [EL99].

La disponibilité est généralement exprimée par une capacité propre à chaque ressource k, notée Q_k ($Q_k \geq 1$).

Une ressource est dite renouvelable si, après avoir été utilisée par une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines ...). Dans le cas contraire, elle est dite consommable (matières premières, budget ...).

Par ailleurs, nous distinguons dans le cas des ressources renouvelables : les ressources disjonctives (ou non partageables) qui ne peuvent exécuter qu'une seule opération à la fois, et les ressources cumulatives (ou partageables) qui peuvent être utilisées par plusieurs tâches à la fois.

Les problèmes non préemptifs à ressources renouvelables disjonctives couvrent une classe importante d'applications : les problèmes d'atelier [2][EL99].

[2]Ordonnancement du passage des travaux sur les machines dans l'atelier, il est souvent dit ordonnancement des fabrications. Parallèlement, l'ordonnancement de projet sert à connaître le déroulement du projet, et à prévoir l'enchaînement des opérations.

(c)La notion de contrainte

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décision. On en distingue [EL99] :

- ***Les contraintes cumulatives*** sont liées à l'utilisation simultanée d'une même ressource par plusieurs tâches et indiquent que leur capacité est limitée ;

- ***Les contraintes disjonctives*** spécifient que le traitement d'un ensemble de tâches sur la ressource disjonctive ne peut se faire simultanément au même instant ;

- ***Les contraintes physiques*** spécifient les caractéristiques limitant le fonctionnement de l'atelier, dues à l'organisation physique des moyens de production ;

- ***Les contraintes temporelles*** regroupent les contraintes de précédences et de localisation temporelle.
  - Les contraintes de précédences entre les tâches prennent en compte la succession des opérations de la gamme opératoire. Ces contraintes indiquant que la date de début d'une tâche j doit commencer après la date de fin de réalisation de la tâche i, est représentée par une inégalité de potentiels :

    $$t_j - t_i \geq p_i$$

    Où t_i dénote la date de début d'exécution de la tâche i.

  - Les contraintes de localisation temporelle définissent l'intervalle de temps sur lequel la tâche doit être traitée. Cet intervalle est limité par les valeurs des dates de début au plutôt et de fin au plus tard .

(d) Objectifs

Les critères les plus souvent rencontrés dans la littérature pour l'évaluation d'une solution d'un problème d'ordonnancement sont l'utilisation efficace des ressources, le délai global, la minimisation des encours et le respect des dates échues. Ces critères sont fonction des variables de décision suivantes, associées aux tâches à ordonnancer [EL99] :

- La date de fin ou d'achèvement C_i de la tâche i, (Completion time) ;
- La durée de séjour $F_i = C_i - r_i$ de la tâche i dans l'atelier,$F_i = C_i - r_i$ Flow time ;
- Le retard algébrique $L_i = C_i - d_i$ (Lateness), il dénote l'écart par rapport au délai souhaité avec avance favorable et retard défavorable ;
- Le retard vrai $T_i = max(0, C_i - d_i)$ (Tardiness) ;
- L'indicateur de retard $U_i (U_i = 0\ si\ C_i \leq d_i, U_i = 1\ sinon)$;
- L'avance $E_i = max(0, d_i - C_i)$(Earliness).

Les mesures de performance usuelles sont [EL99] :

- La durée totale de l'ordonnancement $C_{max} = \max C_i$(make-span) ;
- Le plus grand retard algébrique $L_{max} = \max L_i$;

- La plus grande durée de séjour $F_{max} = \max F_i$;
- La durée moyenne de séjour$\overline{F} = \frac{1}{n}\sum_1^n F_i$;
- Le plus grand retard vrai $T_{max} = \max T_i$;
- Le retard moyen $\overline{T} = \frac{1}{n}\sum_1^n T_i$;
- Le nombre de tâches en retard $N_T = \sum U_i$.

A partir de ces mesures, des critères d'appréciation ou d'évaluation des solutions du problème d'ordonnancement sont construits.

Pour les mesures que nous avons présentées, l'objectif consiste dans tous les cas à minimiser ces mesures de performance. On remarque que tous les critères sont basés sur l'inconnue C_i à l'exception du critère qui est aussi basé sur les dates échues d_i .

(e) Propriété des critères

Un critère est dit *régulier* quand sa valeur décroît si et seulement si l'un au moins des C_i décroît [EL99](les autres restants fixes).

Un sous-ensemble de solution est dit *dominant* pour un critère donné s'il contient au moins un optimum relativement à ce critère [EL99]. Les critères présentés ci-dessus sont tous réguliers.

1.2.3 Classifications des problèmes d'ordonnancement

Dans le domaine de l'ordonnancement d'atelier, la notion structurante de gamme opératoire est souvent utilisée comme critère de classification. En effet, une gamme opératoire impose un ordre de passage des produits sur les machines, et donc un ordre des opérations associées à chaque travail.

Mac Carthy et Liu décrivent dans [ML93] une typologie de problèmes d'ordonnancement qui permet de les classifier en fonction de la nature des ressources et des gammes des travaux à réaliser, suivant sept types :

- *le problème job shop* où chaque travail a sa gamme opératoire propre ;
- *le problème flow shop* où chaque travail a une gamme identique ;
- *le problème open shop* où l'ordre de passage sur les machines est libre pour chaque travail ;
- *le problème flow shop de permutation* où chaque travail à une gamme identique et où l'ordre de passage des travaux est le même pour chaque machine ;
- *le problème à une machine* où chaque travail est assimilé à une opération unique, exécutée par une seule et même machine m de capacité $Q_m = 1$;
- *le problème à machines parallèles* où chaque travail est assimilé à une opération unique, exécutée par une machine à sélectionner dans un ensemble ;
- *le problème job shop à machines dupliquées* où chaque travail a sa gamme opératoire propre et où chaque opération est réalisée par une machine à sélectionner dans un

ensemble.

La figure 1.3 synthétise et généralise cette typologie comme proposée dans [Bil99] :

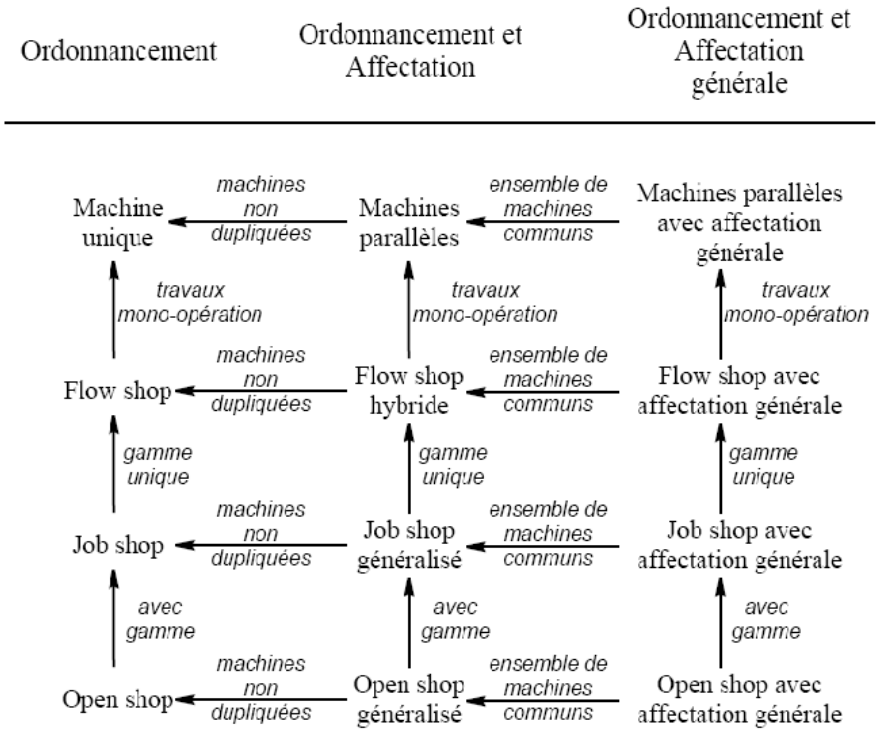


FIG. 1.3 – une typologie des problèmes d'ordonnancement [Bil99]

Dans le présent travail, on s'intéresse plus particulièrement aux problèmes du type Job Shop. Nous nous proposons de les traiter, avec de plus amples détails, dans le prochain chapitre.

S'inspirant de cette typologie, la notation proposée dans [GLLK79], puis reprise dans [JEP+96], s'est rapidement imposée comme faisant référence.

La classification de Graham et al. [GLLK79], se base sur la notation $\alpha/\beta/\gamma$ et permet de caractériser un problème d'ordonnancement de manière précise. Le champ α décrit la structure du problème et se décompose généralement en deux sous champs α_1 et α_2, le premier indiquant la nature du problème (job shop, flow shop, etc.), le second précisant le nombre de machines. Le champ β décrit les types de contraintes prises en compte. Enfin, le champ γ indique la fonction objectif considérée. À titre indicatif, quelques valeurs classiques de $\alpha_1/\beta/\gamma$ sont décrites dans l'annexe A.

A titre d'exemple la notation $J_3/p_i = 1/C_{max}$, indique qu'il s'agit d'un problème d'ordonnancement de type Job shop l, à 3 machines, dont les tâches ont toutes des durées opératoires unitaires, avec comme objectif la minimisation du Make span.

1.2.4 Complexité des problèmes d'ordonnancement

La théorie de la complexité a été crée dans le but de fournir des informations sur la difficulté théorique des problèmes que l'on peut être amené à résoudre. Concrètement, on cherche à déterminer " *mathématiquement* ", si le problème étudié est plutôt " *facile* " ou bien s'il est plutôt " *difficile* ". Des classes de problèmes ont été définies, et certaines propriétés qui sont associées à ces classes aident à répondre à de telles questions. Pratiquement la connaissance de telles propriétés permet souvent de guider dans le choix de l'algorithme à développer [Ess03].

Dans la théorie de la complexité, la difficulté d'un problème est mesurée par la complexité du meilleur programme résolvant ce problème de façon optimale.

Un algorithme est dit polynomial si son temps d'exécution est borné par une fonction polynomiale de la taille de l'instance. On dit qu'il a une complexité en O(p(n)), ou p est un polynôme et n la taille de l'instance d'entrée. Dans le cas contraire, il est dit exponentiel.

Pour des problèmes d'optimisation, on parle d'un problème P lorsque il existe un algorithme polynomial permettent de le résoudre. Un problème est dit NP lorsqu'il existe un algorithme non déterministe polynomial pour le résoudre. Un problème est NP-difficile lorsqu'aucun algorithme de complexité O(p(n)), ne peut le résoudre. Cette classe de problèmes est divisée en deux :

- Les problèmes NP-difficile au sens faible, que l'on peut résoudre avec un algorithme de résolution en O(p(n,l)), ou p est un polynôme, n la taille de l'instance d'entrée, et l le plus grand nombre de l'instance. ;
- Les problèmes NP-difficile au sens fort, qu'aucun algorithme en O(p(n,l) peut résoudre.

La plupart des problèmes d'ordonnancement sont des problèmes ***NP-difficile au sens fort***. Le problème Job Shop est classé dans la théorie de la complexité parmi les problèmes ***NP-Difficile au sens fort***. Nous pouvons trouver la démonstration de ce résultat dans [LK78].

1.3 Approches classiques de résolution des problèmes d'ordonnancement

Les problèmes d'optimisation combinatoire en général, et les problèmes d'ordonnancement, en particulier, ont donné lieu au développement de nombreux types de méthodes de résolution [Ess03]. Selon les besoins, et selon la classe de complexité du problème, on peut être emmené à développer des algorithmes exacts ou bien des algorithmes approchés. Dans le premier cas, on a la garantie d'obtenir la solution optimale du problème, alors que dans le second, on perd en complétude de la résolution pour gagner en efficacité [Gaz03].

1.3.1 Les méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Le temps de calcul de ces méthodes est généralement exponentiel ce qui explique qu'elles ne sont utilisables que sur des problèmes de moyenne ou de petite tailles. Parmi ces méthodes, on peut citer :

- ***Les Procédures par Séparation et Évaluation (PSE)*** qui énumèrent par une recherche arborescente un ensemble de solutions, en éliminant les branches de l'arbre de recherche montrées non optimales afin d'éviter la numération exhaustive des solutions (utilisation de borne inférieure et supérieure du critère). Les méthodes de ce type sont généralement de complexité exponentielle.

- ***Les méthodes basées sur la Programmation Linéaire (PL)***, modélisant les critères et les contraintes comme des fonctions linéaires des variables, et utilisant pour la résolution la méthode du Simplexe couplée à des méthodes de points intérieurs. En pratique, la méthode du simplexe est performante, bien que sa complexité théorique soit exponentielle. Le meilleur exemple d'une méthode de points intérieurs est l'algorithme de Karmarkar qui résout les programmes linéaires en un temps polynomial [LR01].

- ***Les méthodes basées sur la Programmation Dynamique (PD),*** introduite par Bellman dans les années 50 [Bel57], elle consiste en une décomposition du problème principal de dimension n, en sous problèmes de dimension n-1, que l'on résout en tenant compte à chaque étape des informations issues de la résolution du sous problème précédent. La complexité de cette méthode est soit polynomiale, pseudo polynomiale ou exponentielle, selon le type du problème résolu.

1.3.2 Les méthodes approchées (heuristiques)

Elles sont souvent utilisées pour traiter les problèmes que les méthodes optimales n'arrivent pas à résoudre en un temps acceptable [Gaz03]. Elles produisent généralement une solution faisable de bonne qualité en un temps relativement court. La qualité d'une heuristique doit être évaluée sur plusieurs jeux d'instance de taille variable afin de mesurer d'une part, la déviation moyenne du critère par rapport à sa valeur optimale, et d'autre part, l'évolution du temps de calcul en fonction de la taille ou de la structure du problème. Ces heuristiques sont classiquement classifiées en trois grands types :

- ***Les algorithmes gloutons*** [CC88] dans lesquels les décisions d'ordonnancement sont prises progressivement, à temps croissant, au fur et à mesure que les ressources se libèrent, grâce à des règles de priorité simples de type SPT (Shortest Processing Time), EDD (Earliest deadline) ;
- ***les méthodes de recherche locale [CC88]*** (tabou, recuit simulé, algorithmes gé-

nétiques, etc.) qui, partant d'une solution initiale, définissent un voisinage, qui est ensuite exploré pour trouver des solutions meilleures ;
- ***les méthodes de recherche arborescente tronquée [CC88]***, proches des PSE, excepté que l'arbre de recherche est volontairement restreint, quitte à perdre des solutions optimales, afin de gagner en temps de calcul.

1.4 Gestion des incertitudes en ordonnancement

Que ce soit en gestion de production ou en gestion de projet, une caractéristique importante de l'environnement dans lequel s'applique un ordonnancement réside dans son caractère incertain. En effet, l'ordonnancement dépend de paramètres internes, propres au système assurant sa réalisation (capacité des ressources, durées opératoires, . . .), et de paramètres externes, correspondant aux informations données par les fournisseurs et les sous-traitants participant indirectement à sa mise en oeuvre (délais, volumes de production,...). Or, lors de la construction de l'ordonnancement, ces paramètres sont souvent supposés donnés et constants alors qu'en réalité, ils sont mal connus et susceptibles de varier dans le temps de façons plus ou moins prévisibles.

Avant de voir les différentes sources d'incertitudes, commençons par définir clairement ce que nous entendons par incertitude. En particulier, nous insistons sur la distinction que nous faisons entre*incertitude* et *aléa* :

" *On parle d'incertitude pour désigner les modifications potentielles des données d'un problème d'ordonnancement qui peuvent intervenir entre le calcul d'un ordonnancement et la fin de sa mise en oeuvre réelle dans son environnement* "[Ess03].

" *On parle d'aléas pour désigner les types particuliers d'incertitudes qui concernent les données structurelles du problème, et non pas les données chiffrées* "[Ess03].

Ainsi, une durée opératoire plus longue que prévu est considérée comme une incertitude, tandis que l'arrivée d'un nouveau travail, ou le changement du nombre des machines disponibles sont considérés comme des aléas.

1.4.1 Processus global d'ordonnancement

En prenant en compte l'environnement dynamique de mise en oeuvre de l'ordonnancement, le problème globale d'ordonnancement peut se décomposer en deux phases [Ess03] :

1. ***La phase statique hors ligne***, ou une solution est produite en se basant sur des hypothèses plus au moins restrictives et en n'ayant qu'une vision inévitablement imparfaite du problème à résoudre ;

2. ***La phase dynamique en ligne***, ou la solution disponible à la fin de la phase statique est effectivement mise en oeuvre dans un environnement dynamique.

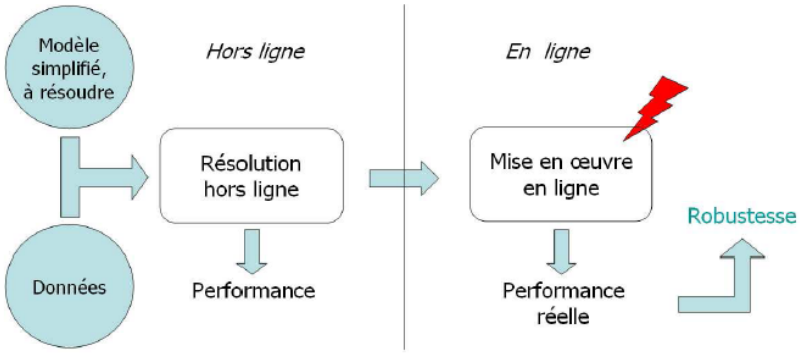


FIG. 1.4 – Processus global dordonnancement [Ess03]

Proposer une méthode d'ordonnancement revient donc à proposer, idéalement, à la fois un ordonnancement statique et un ordonnancement dynamique. Pourtant plusieurs travaux ne portent que sur la phase hors ligne, et négligent l'aspect dynamique de l'ordonnancement [Ess03]. De nos jours nous savons pertinemment, que cette hypothèse est fortement discutable, et de manière générale il faut prendre en compte les incertitudes et les aléas pour répondre de manière efficace à l'aspect indiscutablement dynamique de l'environnement de l'application. Cette prise en compte se fait en développant des méthodes d'ordonnancement robustes.

1.4.2 Notion de robustesse en ordonnancement

La littérature présente plusieurs définitions du terme robustesse. Dans ce qui suit nous citons celles qui nous ont le plus inspiré :

" *Un ordonnancement robuste est un ordonnancement capable d'absorber certaine quantité d'événements inattendus sans avoir à être réordonnancé.* " [DB00].

" *Nous utilisons le terme de robustesse pour caractériser la performance d'un algorithme ou plutôt d'un processus complet de construction d'un ordonnancement en présence d'aléas.* " [GOT02].

" *... Un ordonnancement robuste est un ordonnancement qui est insensible à des perturbations imprévues de l'atelier de production, étant donnée une politique de contrôle a priori.* " [LWS94]

" *... un ordonnancement robuste est un ordonnancement dont les performances sont acceptables en présence d'incertitudes.*" [Ess03]

Cependant ,plusieurs auteurs [BS04, Ess03, BLE03] mettent en évidence que la robustesse a un prix : plus un résultat est robuste (insensible aux incertitudes), et plus en contrepartie sa performance est mauvaise, et inversement. Les auteurs constatent que, pour un niveau d'incertitude donné, plus on souhaite que la robustesse de solution soit importante (c'est à dire que la probabilité d'infaisabilité soit faible), et plus il faut accepter en contrepartie une dégradation de qualité importante.

1.4.3 Notion de flexibilité

Les termes de robustesse et de flexibilité sont très souvent indissociables. En effet, comme il est difficile de proposer une définition unique et universelle de la robustesse, définir le mot flexibilité n'est pas trivial.

Selon [Ess03], *la flexibilité est la liberté dont on dispose durant la phase de mise en oeuvre d'un ordonnancement.*

Selon [ET88], *la flexibilité d'un objet, d'un système, est une propriété qui recouvre deux aspects complémentaires mais distincts* :

- *Un aspect interne* lié à une capacité de changement, de déformation, à une variété d'états possibles. Ce premier type de flexibilité est lié à la nature même de l'objet, les auteurs utilisent aussi le terme de flexibilité statique pour le désigner ;

- *un aspect externe* lié à une capacité d'adaptation à des modifications de l'environnement, à des perturbations. Ce second concerne les interactions avec son environnement, et sa fonction au sein d'un ensemble plus vaste. Cette flexibilité est aussi appelée flexibilité dynamique.

Nous remarquons, que si la capacité de changement (la flexibilité statique) peut être mesurée indépendamment de l'environnement de l'objet, sa capacité d'adaptation (la flexibilité dynamique) dépend du type de perturbations qu'il doit être capable d'absorber.

Dans [GOT02], pour le domaine de l'ordonnancement, les auteurs indiquent que :

"La flexibilité peut être exprimée comme l'existence de modifications possibles dans un ordonnancement, calculé hors-ligne, entraînant une perte de performance acceptable. Elle peut aussi être associée à un voisinage de solutions pouvant être examiné lors de l'exécution, ou à une famille d'ordonnancements, sans privilégier un ordonnancement en particulier".

Il s'agit donc là d'une flexibilité statique pouvant avoir plusieurs formes [BMS04]. Nous distinguons :

- ***la flexibilité temporelle***, concernant les dates de début des tâches, et autorisant sous certaines limites une dérive de ces dernières dans le temps ;

- ***la flexibilité séquentielle***, qui est parfois appelée flexibilité sur les ordres. Il s'agit cette fois d'autoriser des modifications de l'ordonnancement au niveau des séquences d'opérations sur les machines. l'introduction de flexibilité séquentielle impose de manipuler, non pas un seul ordonnancement, mais une famille d'ordonnancements ;

- ***La flexibilité sur les affectations***, qui comme son nom l'indique, est présente lorsqu'on permet à une opération d'être exécutée sur une ou plusieurs ressources auxquelles elle n'avait pas été affectée initialement ;

- ***La flexibilité sur les modes d'exécution***,ce quatrième type de flexibilité est nécessaire si on veut pouvoir modifier le mode d'exécution d'une opération. Par exemple, il peut être profitable de préempter une opération, de modifier une gamme opératoire, Cette flexibilité est certainement la plus poussée et la plus osée de toutes les flexibilités qu'on vient de voir puisqu'elle suppose un changement dans les données même du problème initial .

1.5 Une classification des approches d'ordonnancement robuste

Depuis maintenant plusieurs années, la littérature ne cesse de s'enrichir en méthodes d'ordonnancement robustes. En effet, plusieurs classifications de ces méthodes peuvent être envisagées, la plus courante serait de les classer par les phases au cours desquelles sont prises en compte les incertitudes. Davenport et Beck [DB00] proposent la classification suivante :

Les approches proactives tentent de prendre en compte l'incertain lors de la phase d'ordonnancement hors-ligne uniquement .

Les approches réactives prennent en compte l'incertain lors de la phase d'ordonnancement dynamique uniquement.

Les approches proactives-réactives tentent de combiner avantageusement les deux techniques précédentes .

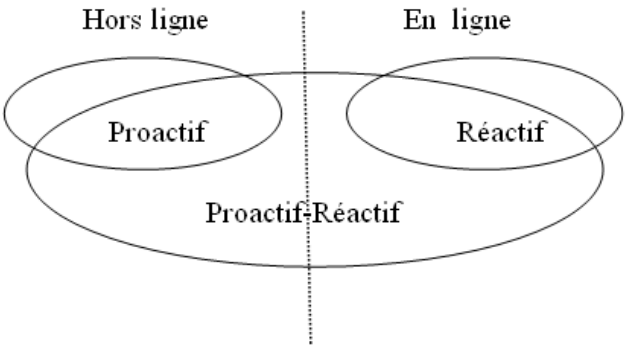


FIG. 1.5 – Positionnement des différentes approches dordonnancement en présence dincertitudes dans les différentes phases du processus global dordonnancement [Ess03]

1.5.1 Les approches Proactives

La proactivité désignent la volonté d'anticiper les perturbations avant qu'elles ne se produisent [Tru05].

Plusieurs manières de présenter les approches proactives existent dans la littérature concernant l'ordonnancement en présence d'incertitudes. A titre d'exemple, on peut citer les approches basées sur la redondance, les approches stochastiques, ou bien les approches générant une famille d'ordonnancement. Afin de souligner l'apport en flexibilité qu'offre chaque méthode, plusieurs auteurs [Ess03, Tru05], classent ces méthodes en deux sous-classes selon si :

- l'anticipation a lieu lors de la phase hors-ligne et consiste à construire une seule solution robuste, ces méthodes sont connues dans la littérature sous le nom de ***méthodes non flexibles*** [DB00] ;

- l'anticipation a lieu lors de la phase hors-ligne et vise à construire une famille flexible de solutions. Ce sont ***les méthodes flexibles*** [DB00] .

1.5.2 Les approches Réactives

Contrairement aux approches proactives qui se proposent de prendre en considération les incertitudes lors de la phase statique du processus d'ordonnancement global, les méthodes réactive procèdent de manière symétrique, en ne prenant en compte les incertitudes que lors de la phase dynamique du processus de résolution .

Ces approches sont souvent utilisées dans des environnements fortement perturbés, où les incertitudes sont fréquentes et de fortes amplitudes. Dans de tels environnements, un ordonnancement de référence peut rapidement s'avérer de mauvaise qualité ou même infaisable. Par conséquent, toutes les décisions sont prises en temps réel, en utilisant des stratégies qui privilégient la rapidité des décisions sur leur qualité .

Généralement, les méthodes réactives se basent sur l'application des règles de priorité. Par exemple on commencera par l'exécution de la tâche ayant la durée opératoire la plus longue (" Longest Processing Time first"), ou bien celle correspondant au travail dont l'échéance est la plus proche (" Earliest Due Date first"), parmi les opérations disponibles pour l'exécution. Une extension naturelle des méthodes de ré-ordonnancement par règles de priorité est de proposer un choix dynamique de ces règles permettant ainsi au système de modifier la règle utilisée selon le contexte [PGQ98].

Dans le cadre des méthodes issues de l'intelligence artificielle, Lim et Zhang dans [LZ02], proposent une approche réactive basée sur la coopération inter ressources. Pour faciliter le processus d'affectation des tâches et manipuler la négociation entre les agents, les auteurs proposent un mécanisme d'attracteur itératif. Ce mécanisme permet de créer des plans de production de manière concurrente. Le point fort de cette approche est la grande liberté dont dispose chaque agent, mais la difficulté se situe au niveau de la

définition des objectifs de chaque agent, ceux-ci devant être ajustés pour assurer une bonne performance globale du système.

1.5.3 Les approches Proactives -réactives

Ce sont des approches où les incertitudes sont prises en compte dans la phase statique et dans la phase dynamique du processus de résolution. Selon [DB00], un système qui est capable de prendre en compte les incertitudes en ordonnancement est un système qui prend en compte les incertitudes durant la phase de génération de la solution statique, et qui peut réagir, durant l'exécution, à l'arrivée d'événement inattendus.

Plusieurs approches proactives réactives proposées dans la littérature se basent sur l'utilisation du concept de groupes d'opérations permutables [Dem77, Tho80, Gal89, Bil93, Art97]. Ce concept a été développé au LAAS-CNRS (Laboratoire d'Analyse et d'Architecture des Systèmes), il y a plus de vingt cinq ans. L'idée générale de cette approche est de construire, durant la phase hors ligne, un ordonnancement partiel en déterminant pour chaque ressource, une séquence de groupes de tâches, les tâches d'un même groupe étant totalement permutables. Un avantage de cet ordre partiel réside dans la possibilité de pouvoir évaluer la performance au pire de l'ordonnancement (c'est à dire la permutation la plus défavorable vis-à-vis de la minimisation du plus grand retard algébrique). La méthode utilise une heuristique qui construit une séquence de groupes initiale. Cet ordonnancement est ensuite exploité en temps réel : le séquencement des tâches d'un même groupe est défini, au fur et à mesure de l'exécution de l'ordonnancement, par un décideur qui dispose d'indicateurs quant à la qualité de chaque choix vis-à-vis de la performance.

Les approches basées sur le concept de groupe d'opérations permutables autorisent uniquement la permutation des tâches placées de manière contiguë sur la ressource. Pour dépasser cette limite, Trung dans [Tru05] et en se basant sur un concept de dominance définit un ordre partiel exhibant davantage de flexibilité comparativement à la flexibilité dégagée en utilisant le concept de groupe de tâches permutables. L'approche proposée vise à définir une famille flexible de solutions pour un compromis flexibilité/qualité satisfaisant. L'ensemble flexible de solutions est caractérisé par exploitation de structures d'intervalles et permet ainsi de définir des ordres sur un sous ensemble des tâches du problème. Les ordres ainsi définies et qualifiés de partiels permettent de borner la qualité au pire de l'ensemble flexible caractérisé et de mesurer sa flexibilité. De telles approches sont fortement intéressantes du fait qu'elles permettent de déterminer interactivement la solution qui correspond au compromis préféré par le décideur, d'autant plus que ces approches sont relativement insensibles aux variations des données des problèmes considérés .

1.5.4 Synthèse des différentes approches et justification de l'approche retenue

Un des facteurs essentiels qui maintiennent le large fossé entre la théorie et la pratique, est la notion générale d'incertitude. En ordonnancement, comme pour plusieurs autres domaines, de nombreux chercheurs oeuvrent afin de proposer des méthodes basées sur les

concepts de flexibilité et de robustesse. Selon le type du problème étudié, la dynamique de variation des perturbations (aléas et incertitudes) et les objectifs attendus, la littérature propose une panoplie de méthodes de résolution robustes.

Les approches proactives prennent en compte les incertitudes uniquement hors ligne et aucun algorithme d'adaptation en ligne n'est utilisé.

Les approches réactives ne fournissent au final, que de faibles performances comparativement à l'ordonnancement optimal. Cependant, elles ont l'avantage de construire un ordonnancement faisable ce qui est en soi une qualité essentielle dans un environnement fortement perturbé. Le second avantage de ces méthodes est leur besoin calculatoire très minime .

Afin de disposer d'une vision plus réaliste et moins optimiste, les approches proactives réactives tentent d'anticiper les fluctuations de l'environnement en construisant un ordonnancement qui intègre explicitement les incertitudes. Ces méthodes permettent de réagir en ligne aux événements imprévus qui apparaissent durant l'exécution, en autorisant le passage d'une solution devenue obsolète à une autre sans remise en compte de calculs déjà effectués.

Dans le cadre des travaux menés dans ce mémoire, nous nous sommes intéressés à la classe d'approches proactives réactives qui sont très adaptées pour faire face aux incertitudes. Notre intérêt est particulièrement porté sur les approches fournissant de la flexibilité séquentielle, dans une optique de recherche de compromis flexibilité / performance. En ce sens, nos travaux se basent sur ceux dévélopées par Trung [Tru05] cités plus haut.

D'un autre point de vue et comme développé dans [OBB07], nous pouvons dire que d'une façon classique, la résolution en ligne ou hors ligne de problème d'ordonnancement avec prise en compte des incertitudes est assimilée à une fonction globale car l'ordonnancement gère l'organisation de la totalité des ressource. Partant du fait que les ressources peuvent disposer d'une autonomie décisionnelle dont il est important de tenir compte, nous avons adopté dans notre travail une démarche coopérative entre les différents acteurs représentant les ressources.

Dans ce qui suit, nous nous proposons d'évoquer quelques notions concernant la distribution de la fonction de l'ordonnancement, ainsi que les principaux travaux qui en ont découlé.

1.6 Systèmes d'ordonnancement distribué

Selon Tranvouez [Tra01], un Système d'Ordonnancement (Scheduling System) est " *un programme informatique dont la fonction est de produire des solutions efficaces pour l'allocation (ou la re-allocation) de tâches sur des ressources tout en respectant certains*

objectifs et contraintes."

Cette définition souligne le fait que la dénomination *"Système d'Ordonnancement"* indique que le développement de ces systèmes ne se limite pas à implanter une méthode de résolution dans un système informatique. En effet, au-delà de l'aspect logiciel, ces systèmes prennent en compte des contraintes pratiques spécifiques ignorées des méthodes plus théoriques .

Les systèmes d'ordonnancement fonctionnent de manière très simple : pour trouver une solution réalisable, le dit système procède à une analyse des données sur le problème posé. Cette analyse dépend de la modélisation du problème, et de la base théorique retenue pour générer les solutions (méthodes exactes, heuristiques, etc.).

L'Intelligence Artificielle ***(IA)*** a joué un rôle important dans l'élaboration de ces systèmes [SR95]. En effet, il existe même une branche de l'IA qui s'occupe de manière exclusive des problèmes distribués, il s'agit de l'***IAD*** (intelligence artificielle distribuée) dédiée à la distribution des calculs en ordonnancement [Par96].

L'*intelligence qrtificielle distribuée* s'intéresse en particulier à des comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents. En effet, elle a introduit le concept de système multi-agents qui consiste à Faire coopérer un ensemble d'entités (agents) dotées d'un comportement intelligent, de manière à coordonner leurs buts et leurs plans d'actions pour résoudre un problème.

1.6.1 calcul centralisé Versus calcul distribué

Pendant longtemps, les systèmes d'ordonnancement issus de l'Intelligence Artificielle (IA) se sont appuyés sur une conception centralisée de la résolution du problème d'ordonnancement. Et ce n'est qu'au cours des années 80, que les progrès technologiques ont permis d'envisager la distribution du calcul d'ordonnancement [Tra01].

Les organisations centralisées font l'hypothèse de l'existence d'une entité qui a une vision globale des caractéristiques des flux sur l'ensemble des ressources. Cette entité supervise alors toutes les activités et prend les décisions globales devant être respectées par l'ensemble des machines [Des05]. La centralisation des décisions permet d'assurer leur cohérence et de garantir une performance globale.

Un inconvénient des approches centralisées est qu'elles requièrent une totale transparence des acteurs, ceux-ci devant communiquer à l'entité superviseur les caractéristiques de leur production. De plus, elles imposent que les acteurs respectent les décisions prises par le superviseur et n'autorisent pas de remise en cause de ces décisions. Cette hypothèse semble peu acceptable dans un contexte réel d'application, où le pouvoir de décision est réparti de façon homogène entre les différentes ressources. En plus, la centralisation des décisions augmente la complexité du problème et réduit les possibilités d'effectuer des calculs parallèles.

L'approche de résolution distribuée propose de répartir les calculs d'ordonnancement sur des systèmes logiciels autonomes et ainsi de mieux appréhender la complexité du problème d'ordonnancement. Cette approche permet de parvenir à une solution globale entre plusieurs acteurs tout en conciliant les objectifs collectifs et individuels.

Le changement de paradigme (distribué versus centralisé) consiste à aborder un problème d'ordonnancement selon plusieurs points de vue (partiels ou complets) permettant ainsi d'améliorer la qualité de sa modélisation comme l'illustre la figure 1.6.

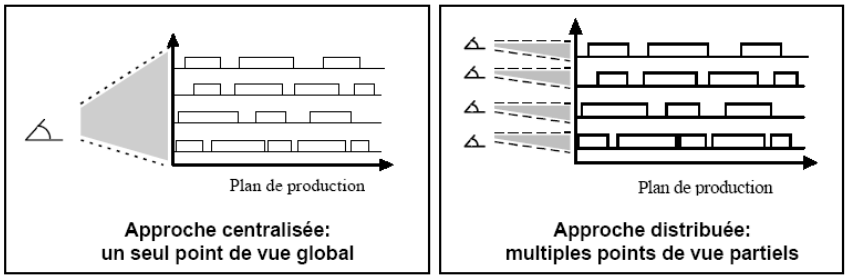


FIG. 1.6 – Différentes perceptions d'un problème d'ordonnancement [Tra01]

1.6.2 Distribution de la fonction d'ordonnancement

Distribuer c'est avant tout décomposer c'est à dire identifier dans un problème global un ensemble de sous problèmes partiels. Le processus d'identification est fortement induit par la modélisation initiale du problème global [Tra01]. Il y a, en effet, autant de modes de distribution que de possibilités de décomposer un problème d'ordonnancement et d'interpréter cette décomposition. Les éléments de chaque décomposition doivent bien sur, avoir assez de point commun pour que l'on puisse les regrouper sous le même sous ensemble. A titre d'exemple, un problème d'ordonnancement à plusieurs machines peut être décomposé en plusieurs problèmes à une machine .

Il convient ensuite de définir les unités de résolution auxquelles vont être allouées les sous problèmes identifiés (agents artificiels et/ou humain), déterminer leur degré de liberté d'action (agent dirigé ou autonome) ainsi que leur organisation sociale (hiérarchique ou décentralisée). Enfin il reste alors à préciser leurs modes d'interaction dans la construction d'une solution d'ordonnancement finale (mode de communication : coordination ou coopération, etc.).

Face au grand nombre d'options conceptuelles, il est difficile de proposer une classification qui recouvre tous les cas possibles.

En effet, plusieurs variantes de l'approche distribuée ont vu le jour. Tranvouez dans [Tra01], propose une classification des différentes approches de résolutions utilisées dans

les systèmes d'ordonnancement distribués. Cette classification comprend :

Les approches basées sur l'aide à la décision se basent sur la distribution de la charge de résolution entre deux acteurs : un acteur logiciel (le système) et au moins un acteur humain (le décideur). Le système d'ordonnancement distribué est formé d'un environnement logiciel (centralisé ou non) et d'au moins un décideur humain [Tra01].

Les approches basées sur l'allocation des ressources (AR) cherchent à trouver la ressource la plus compétente pour traiter une tâche de production. Le processus est itératif et aboutit à une diminution progressive des marges de manoeuvre des ressources. La sélection de la ressource se fait généralement par le biais d'appels d'offres. Deux modes d'attributions existent : un mode dirigé ou un mode décentralisé [Tra01]. Dans les approches AR dirigées, il existe un agent responsable de la cellule de production, qui évalue les offres. L'allocation des ressources est alors, dirigée par ce même agent.

Les approches basées sur la coordination décomposent le problème d'ordonnancement global en sous problèmes d'ordonnancement plus simples alloués à des unités de résolution fonctionnant en parallèle. Du fait des contraintes d'antériorité sur les tâches de production d'un même job, il est impossible que cette décomposition aboutisse à une partition du problème d'ordonnancement initial. Les sous problèmes étant interdépendants, des mécanismes de coordination permettent de maintenir la cohérence entre les solutions locales calculées par les unités de résolution.

La coordination en tant que mécanisme distribué de résolution nécessite une décomposition particulière du problème [Tra01]. Cette décomposition conduit soit à rajouter une entité superviseuse, soit à conditionner les différentes entités afin que leurs actions convergent vers une solution. Ceci impose une limitation des autonomies des agents si ce n'est dans leur capacité de décision (connaissances et moyens de choisir) au moins dans leur champ d'action. L'approche par coopération permet de dépasser ces limitations. Celle-ci étant l'objet même de notre étude nous lui réservons la section suivante.

1.6.3 Approche d'ordonnancement distribué basée sur la Coopération des ressources

Kallel et al. dans [KPB85], considèrent que l'organisation purement hiérarchique des systèmes de production est un frein à leur rapidité de réaction face aux aléas. Ils lui opposent une approche décentralisée dans laquelle des centres de décision autonomes coopèrent pour gérer des conflits de ressource sur des tâches de production. De la même manière, ce constat a été suivi par l'approche de résolution coopérative de problèmes d'ordonnancement, et ce, pour remettre en cause l'organisation des unités de résolutions jusqu'alors fortement hiérarchisée .

L'approche coopérative s'appuie sur une certaine autonomie de ces unités de résolution et sur leur capacité à coopérer pour aboutir à une solution harmonieuse. Elle apparaît dans la plus part du temps, comme une solution aux limites de la coordination dont

l'objectif est de limiter au maximum les conflits résultants des interdépendances entre les objets de l'ordonnancement [HETL96].

Il existe plusieurs travaux antérieurs dans lesquelles l'approche distribuée coopérative est appliquée. A titre d'exemple, nous citons le travail de [Des05], dans lequel une approche par coopération inter-entreprises au sein d'une chaîne logistique est développée. En effet, les entreprises partageant des objectifs et des intérêts communs, peuvent être en situation de concurrence pour certaines activités et peuvent coopérer pour d'autres. Pour ce, seule une approche distribuée pourrai assurer à chaque entreprise l'autonomie décisionnelle dont elle a impérativement besoin.

Envisager un système d'ordonnancement distribué et coopértaif, implique de disposer d'une architecture capable de supporter la distribution et la coopération de processus des calculs d'ordonnancement concurrent. Les Systèmes Multi-Agents répondent à ce besoin en fournissant des cadres formels et applicatifs adaptés et éprouvés. De fait, il est à intéressant de constater que de nombreuses approches ont recours à une plate-forme Multi-Agents pour leur développement, parfois de manière complètement mplicite.

1.7 Aperçu sur les systèmes multi agents

Comme mentionné précédemment, une approche centralisée ne pouvait recouvrir la complexité de nombreux problèmes rencontrés en Intelligence Artificielle. Ce constat a causé l'émergence de l'Intelligence Artificielle Distribuée (IAD) [Fer95], et a conduit à distribuer les capacités décisionnelles sur des processus de résolution en interaction. Les Systèmes Multi-Agents (SMA) constituent à cet égard un sous domaine de l'IAD fondé sur l'hypothèse supplémentaire que ces processus, appelés alors "*agents*", possèdent un certain nombre de propriétés particulières. Cette section traite précisément de ce domaine d'étude.

Selon [Tra01], la mise en oeuvre d'un SMA requiert une approche de conception différente si l'on veut éviter d'aboutir à des systèmes fortement hiérarchisés aux comportements figés et peu évolutifs. En effet, la description d'un SMA nécessite la définition des agents qui le composent et des modes d'interactions qui les relient. C'est-à-dire l'identification des propriétés nécessaires d'autonomie, de capacités sociales (communication), de réactivité et de proactivité. Certains auteurs [WJK99], rajoutent même les notions de connaissances, de croyances, d'intention et d'obligation voire d'aptitudes émotionnelle.

1.7.1 La notion d'agent et de centre de décision

L'une des premières définitions du concept d'agent est donnée par J. Ferber dans [Fer95] :

"Un agent est une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le com-

portement est la conséquence de ses observations, de sa connaissance, et des interactions avec les autres agents."

Cette définition générale d'un agent conduit à distinguer deux types d'agents *réactifs* et *cognitifs*. Les systèmes d'agents réactifs s'appuient sur une conception émergente de l'intelligence. Ceux-ci sont souvent comparés aux sociétés d'insectes, ou un grand nombre d'agents de faible granularité peut présenter un comportement global cohérent et efficace. A l'opposé, les systèmes d'agents cognitifs sont constitués d'un petit nombre d'agents de forte granularité : chaque agent est apparenté à un système expert plus ou moins évolué. Les actions de ces agents seront ainsi "réfléchies" en ce sens qu'elles sont basées sur les connaissances de l'agent (sur lui-même, sur les autres et sur son environnement) et les objectifs qui le guident.

Dans plusieurs travaux, les systèmes Multi-Agents sont utilisés de manière complètement implicite. En effet, certains auteurs utilisent les notions de centre de décision [Des05], ou même d'acteur, qui sont particulièrement rattachées à la notion d'agent.

Sur ce, dans le présent travail, on assimile les ressources qui réalisent les tâches, à des centres de décision autonomes qui doivent réaliser un certain nombre de fonctions tout en prenant en compte des données extérieures. Un centre de décision doit alors disposer d'une certaine autonomie puisqu'il est capable d'accepter ou de refuser les requêtes provenant d'autres centres de décision, et de percevoir et d'agir sur son environnement.

1.7.2 Une vision générique fonctionnelle d'un centre de décision

Dans [ABPR02], les auteurs proposent une vision générique fonctionnelle d'un centre de décision. Sur la base d'un modèle, et à partir de paramètres internes ou externes (décisions prises par d'autres centres) et de contraintes locales, un centre d'une part adapte et prévoit son comportement, et d'autre part communique des décisions qui vont elles mêmes influer sur la prise de décision dans d'autres centres.

La figure 1.7, propose une architecture fonctionnelle d'un centre de décision de niveau n. Nous supposons que ce centre est connecté avec un centre de niveau (n+1), un ou plusieurs centres de niveau (n-1) et qu'il peut interagir avec les centres de même niveau.

L'architecture proposée met en évidence quatre fonctions (prévoir, adapter, suivre et décider) organisées autour de trois bases d'information (contraintes locales, paramètres et modèles). Un décideur interagit avec la fonction décider qui lui fournit une aide à la décision. Enfin, toutes les flèches orientées de la figure indiquent des échanges d'informations, celles en pointillés signalent en plus des activations de fonction.

Dans [Des05], l'auteur présente deux conditions nécessaires afin d'assurer le comportement harmonieux des centres de décisions. En effet, pour que la décision prise dans un centre soit consistante vis-à-vis du fonctionnement global d'un réseau de centres de décision, elle doit être :

- ***Cohérente*** vis-à-vis des décisions prises par un centre de décision en amont. Cela

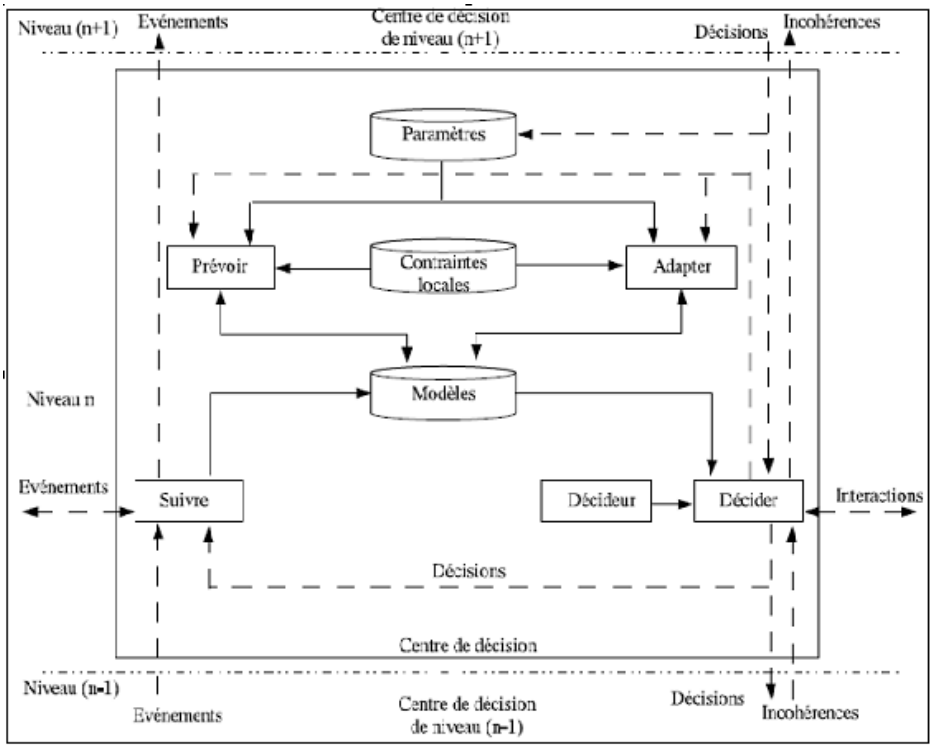


FIG. 1.7 – Une vision générique d'un centre de décision [ABPR02]

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 prises un centre de décision en aval. Cela se traduit par la nécessité de prendre des décisions qui pourront être mises en uvre par l'aval ou d'accepter que ces décisions soient remises en cause.

La difficulté pour un centre de prendre des décisions robustes est liée notamment à la représentation approchée qu'il a des centres situés en aval, ces centres étant eux-mêmes contraints par des décisions provenant d'un autre centre [Des05]. De plus, une décision robuste à un moment donné peut perdre sa qualité au cours du temps compte tenu du caractère dynamique du système dans lequel elle agit. De même, se posent des problèmes de coordination entre centres : en principe, un centre devrait connaître toute décision susceptible de peser sur ses décisions avant de les rendre. Ces problèmes là ne peuvent être réglés sans une profonde connaissance de la manière dont se fait la coopération, du rôle de chaque acteur et de la manière dont se fait l'interaction entre les différents centres de décision.

1.8 Notions relatives à la coopération

1.8.1 Définition de la coopération

Plusieurs définitions de la coopération ont apparu dans la littérature. Nous précisons ci-après celles qui nous ont le plus inspirés dans le cadre de nos travaux :

"La coopération consiste à l'élaboration et à la prise de décision collective." [BCJ02]

"La coopération est l'action collective qui apporte de la cohérence aux décisions prises par les centres de décision individuellement." [Des05]

"La coopération est le support de la mise en oeuvre de décisions entre plusieurs centres de décision."[Ers96]

De toutes ces définitions, celle qui rejoint le plus nos objectifs est celle de Despontin [Des05], car elle souligne le lien entre la coopération est la cohérence des décisions prises individuellement par les centres de décisions.

1.8.2 Les formes de la coopération

Selon les travaux de [Cam00] et [Mon01] cités dans [Des05], il existe trois formes de coopération : La coordination, la collaboration et la codécision.

La coordination vise à synchroniser les actions dans le temps en exploitant un référentiel temporel commun, et à gérer la cohérence des actions individuelles par rapport à l'ensemble des activités.La coopération a pour objet de faciliter la coordination d'activités étroitement complémentaires pour la réalisation des processus.

La collaboration signifie travailler ensemble à l'exécution d'une certaine action pour produire un résultat final. Selon [BCJ02], la collaboration implique le partage d'informations à l'intérieur d'un groupe donné, sans prise de décision collective. Le terme collaboration s'utilise à la place de coopération lorsque les actions individuelles ne sont pas différentiables.

La codécision signifie la collaboration de plusieurs acteurs en vue de prendre des décisions. Cette codécision peut être le résultat de mécanismes de négociation ou de renégociation. Ces mécanismes visent à trouver un compromis acceptable entre les objectifs locaux de chaque entité qui peuvent être contradictoires. Si aucune codécision n'a été prise au préalable, on parlera de négociation ; au contraire si l'objet de la collaboration entre partenaires est la remise en cause d'une décision passée, on parlera de renégociation.

Dans notre approche, la coopération est assimilée à une action de prise de décision collective distribuée en vue de synchroniser les actions réparties sur les différentes ressources. Comme dans [Ers96], nous sommes donc dans un contexte de codécision.

1.8.3 Notions de négociation et de protocole de négociation

La négociation est avant tout *un mécanisme de résolution de conflit* [CdM96]. Cette résolution peut alors s'effectuer dans un contexte favorable ou défavorable [CCC96]. Dans un contexte favorable, toutes les parties sont prêtes à faire les concessions nécessaires pour arriver à un accord. Dans un contexte défavorable, l'objectif principal pour au moins un des participants, est d'obtenir un accord qui le favorise même aux dépends des autres. De fait, la négociation peut être décrite comme *une coopération dont l'objectif commun est l'obtention d'un accord.*

Dans une perspective IAD (Intelligence artificielle distribuée), Davis et Smith définissent la négociation comme " *une discussion dans laquelle les parties intéressées échangent des informations et aboutissent à un accord.*" [DS88].

D'après le petit Robert, la négociation est : " *une série d'entretiens, d'échange de vue, de démarches qu'on entreprend pour parvenir à un accord ou conclure une affaire*".

Cette définition nous parait très proche du modèle que nous voulons proposer, car elle prend en compte la notion de dialogue et celle d'accord entre les différents participants.

L'une des définitions les basiques de la négociation est formulée par Bussman et Muller [BM92] :

"Tous les chercheurs s'accordent sur la finalité de la négociation, à savoir l'aboutissement à un accord commun satisfaisant. Mais la négociation est elle-même définie comme un processus. Toute la diversité des recherches en négociation provient de ce mot : processus. La négociation peut donc être vue comme une boîte noire ayant en entrée un conflit et en sortie un accord, dans le meilleur des cas. La recherche sur la négociation consiste donc à étudier les mécanismes de cette boîte noire, pour la rendre transparente."

Bussman et Muller [BM92] y indiquent notamment que le minimum requis pour un acteur négociant est la possibilité de faire et de répondre à des propositions et de pouvoir indiquer son insatisfaction avec les propositions qu'ils trouvent inacceptables. Les propositions peuvent être soit faites indépendamment des autres propositions, soit basées sur l'historique de la négociation.

Les auteurs clament également que si les différentes entités négociantes peuvent seulement accepter ou refuser les propositions, alors la négociation peut être gourmande en temps et inefficace. Pour améliorer l'efficacité du processus de négociation, le destinataire de l'offre doit être capable de fournir un feedback plus utile sur les propositions qu'il reçoit. Ce feedback peut prendre la forme d'une critique ou d'une contre-proposition. Grâce à de tels feedbacks, l'initiateur devrait être en position de générer une proposition qui est plus à même de conduire à un accord.

Selon Verrons,[Ver04] le déroulement du processus de négociation est appelé *protocole de négociation.*

selon le même auteur, *Chaque forme de négociation possède son propre protocole, c'est-à-dire les actes de langage utilisés et leur séquencement.*

Cette définition souligne deux aspects importants à savoir les messages que les différentes entités peuvent s'envoyer et la dynamique opérationnelle associée.

1.8.4 Différentes phases du processus de coopération

Compte tenu du caractère dynamique des décisions d'ordonnancement, trois phases peuvent être distinguées lors du processus de coopération :

- **la phase de négociation** au cours de laquelle le couple de ressources doit s'entendre pour la première fois afin de rendre cohérent leur ordonnancement local vis-à-vis de la réalisation de deux tâches i et j nouvelles au sein du système ;

- **la phase de coordination** au cours de laquelle le couple de ressource s'échange, au fur et à mesure de la réalisation de leur ordonnancement, des messages informatifs, relatifs aux dates de fin de i et de début de j, afin de se synchroniser ;

- **la phase de renégociation** où il s'agit de remettre en cause les décisions prises lors de la phase de négociation au sein d'un couple de ressource afin de rendre à nouveau cohérent leur ordonnancement local lorsqu'ils ne le sont plus.

La phase de négociation n'est réalisée qu'une seule fois (au moment où un nouvel ordre de fabrication apparaît). La phase de renégociation peut être réalisée autant de fois que nécessaire, à chaque fois qu'une incohérence est détectée, en prenant en compte les décisions antérieures. La phase de coordination est active dès qu'une décision de négociation ou de renégociation a été validée.

1.9 Conclusion

Dans ce chapitre, nous avons commencé par définir quelques concepts généraux rattachés à la fonction ordonnancement, afin de sensibiliser le lecteur à la complexité de cette fonction, et aussi pour faciliter la compréhension du problème d'ordonnancement robuste et distribué qui fait objet de ce mémoire. Nous avons ensuite détaillé les différentes méthodes robustes de résolution des problèmes d'ordonnancement avec présence de perturbations. En nous situant dans le cadre de notre travail, qu'est la résolution du problème Job Shop avec minimisation du C_{max} par une approche distribué basée sur la coopération inter machines, nous avons présenté certaines notions relatives au systèmes d'ordonnancement distribués, aux systèmes multi agents et à la coopération.

Dans le prochain chapitre, nous allons décomposer le problème global en plusieurs sous-problèmes à une machine et nous présenterons une méthode, basée sur un théorème de dominance, permettant de génerer un ordre partiel dominant et ensemble fléxible de

solutions vis à vis d'un critère local donné. Et ce, dans la perspective de proposer un protocole de coopération permettant de résoudre le problème globale.

Chapitre 2

une approche de résolution robuste pour l'ordonnancement Job Shop à une machine

---

Nous vous proposons dans ce chapitre, de décrire un ordre partiel dominant pour le problème à une machine. Cet ordre partiel est défini par un théorème de dominance qu'est le théorème des pyramides et permet de définir un ensemble flexible de solutions pour le problème à une machine avec minimsation du L_{max}. Pour ce, nous allons tout d'abords justifier la décomposition du problème global en plusieurs sous problèmes à une machines puis nous aborderons quelques notions relatives au théorème des pyramides, ensuite nous allons donner une mesure de la qualité au mieux et au pire de l'ensemble de solutions dominantes à savoir des dates de début et de fin au mieux et au pire de chaque tâche. Pour finir, nous allons présenter une méthode de calcul des marges libres disponibles dans la séquence au pire d'une tâche, dans la perspective de proposer, dans le prochain chapitre, une manière d'insérer dynamiquement les nouvelles commandes dans le plan de production établit hors ligne. Mais avant tous cela, nous allons aborder une méthode de modélisation graphique des problèmes Job Shop, qui sera utilisée tout au long de ce manuscrit.

2.1 Modélisation des problèmes job shop par le graphe disjonctif

La modélisation sous forme de graphe disjonctif est très certainement la plus employée pour représenter les problèmes d'ordonnancement, et en particulier le problème du job shop. Cette formulation est très souvent utilisée dans la mise au point de méthodes de résolution. Présentée pour la première fois par [RS64], elle fut reprise un peu plus tard par Balas [Bal69]. Elle est à l'origine des premières méthodes de résolution efficaces pour traiter le problème.

Le job shop est représenté par un graphe $G = (X, C \cup D)$, ou X représente l'ensemble des sommets du graphe G, C est l'ensemble des arcs conjonctifs et D est l'ensemble des

arcs disjonctifs :

- L'ensemble X des sommets de G contient n+2 éléments. Parmi ceux-ci, n sommets sont associés à chacune des n opérations i. Les deux sommets additionnels sont des tâches fictives de durée opératoire nulle, qui représente respectivement, le début et la fin de l'ordonnancement. Tous les autres sommets correspondant à des opérations réelles sont pondérés par leur temps opératoire ;

- L'ensemble C contient les arcs conjonctifs représentant les contraintes de précédence entre les opérations d'un même job (gamme opératoire). Il existe donc un arc entre les sommets i et i+1 de longueur p_i. toutes les premières tâches des gammes opératoires sont reliées à la tâche fictive de début avec un arc de longueur 0. De la même manière toutes les dernières tâches des gammes opératoires sont reliées à la tâche fictive de fin avec des arcs de longueur égale à leurs durées opératoires ;

- L'ensemble D contient les arcs disjonctifs reliant toutes les tâches exécutées par la même ressource. En effet, deux arcs (aller et retour) relient chaque couple d'opérations i et j telles que R(i)= R(j). Ces arcs disjonctifs sont issus de la contrainte qu'une machine ne peut exécuter qu'une opération à la fois ;

- Un graphe sur lequel figurent uniquement les arcs conjonctifs, est appelé graphe conjonctif.

Pour construire un ordonnancement admissible sur le graphe G, il suffit de choisir, pour chaque paire d'arcs disjonctifs, seulement un arc qui déterminera l'ordre de passage sur la machine. Pour que l'ordonnancement soit parfaitement définit, il faut que le graphe soit non cyclique. En effet, l'existence d'un circuit conduirait à affirmer qu'une opération précède et succède à la fois à une autre opération, ce qui est, évidemment, impossible.

pour le problème de minimisation du C_{max} , on choisit la direction des arcs disjonctifs de manière à ce que le graphe soit sans circuits, et le plus long chemin du nud 0 au nud n+1 soit le plus court possible.

Dans le présent travail, la modélisation par graphe disjonctif nous intéresse de manière particulière, non pas pour énumérer les solutions du problème, mais pour calculer les dates de disponibilités et les deadlines des tâches, connaissant leurs durées opératoires, leurs gammes, et les machines qui les exécutent. En effet, ceci est possible grâce à l'algorithme de Ford Bellman, rapporté par Esquirol et Lopez dans [EL99]. A l'origine, cet algorithme était conçu pour à trouver le plus court chemin d'un graphe pondéré à origine unique, y compris dans le cas général ou les arcs sont munis de poids négatifs. Cet algorithme est facilement adapté au problème dual de la détermination des chemins les plus longs, de telle manière que l'on puisse l'utiliser pour déterminer les valeurs de la date de disponibilité r_i, et du deadline di da la tâche i, en utilisant les formules suivantes :

$$r_i = L(0, i) \tag{2.1}$$

$$d_i = L(0, n+1) - L(i, n+1) + p_i \tag{2.2}$$

Les tâches 0 et n+1 représente les deux sommets fictifs de début et de fin du graphe conjonctif.$L(i, j)$ désigne la longueur du plus long chemin de i à j sur le graphe conjonctif du problème. On remarque que L(n+1,0) est initialisé à la valeur $(-C_{max})$, où (C_{max}) correspond à la valeur courante du Makespan.

2.2 Décomposition du problème global en plusieurs sous problèmes à une machine

Comme nous l'avons mentionné antérieurement, il est à constater que 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 d'une autonomie de décision dont il est fondamental de tenir compte. Partant de ce fait, pour résoudre le problème Job Shop à plusieurs machines avec minimisation du C_{max}, une organisation distribuée basée sur le concept de coopération entre centres de décisions associé à une méthode d'ordonnancement robuste représente un choix incontournable. Ce choix est justifié par le fait que puisque le problème $J_n||C_{max}$ est classé NP-difficile, alors il serait intéressant de le décomposer en m sous problèmes à une machine interdépendants, où chaque tâche est caractérisée par une fenêtre d'exécution $[r_i, d_i]$.

Dans [ABZ88], il est montré que la minimisation du C_{max} pour le problème global revient à minimiser le retard algébrique pour chaque sous problème à une machine (problème noté $1|r_i|L_{max}$). Cette décomposition se base essentiellement sur la méthode *Shifting Bottleneck* developpée à la fin des années 80 par Adams, Balas, et Zawack [ABZ88].

Les dates de disponibilité r_i et les dates échues d_i des travaux peuvent être déterminées ainsi que précisé dans la section 2.1.

D'autre part, la fonction ordonnancement doit être distribuée afin de parvenir à une solution globale résultant d'une négociation entre plusieurs acteurs tout en conciliant les objectifs collectifs et individuels [OBB07].

Nous allons présenté l'approche d'ordonnancement robuste à une machine utilisée, approche qui est la source de l'idée de l'organisation distribuée proposée pour la résolution du problème et qu'on présentera dans le troisième chapitre.

2.2.1 Justification de la décomposition du problème global en sous problèmes à une machine

Dans cette section nous allons démontrer la validité de la décomposition du problème global en sous-problèmes à une machine, en nous inspirant des principes de la méthode *Shifting Bottleneck*, qui se base elle même sur cette décomposition.

La méthode Shifting Bottleneck est une heuristique proposée en 1988, dont l'idée principale est d'utiliser les résultats théoriques obtenus dans l'étude du problème d'ordonnancement à une machine, pour résoudre un problème Job Shop à plusieurs machines avec minimisation du C_{max} [ABZ88].

La méthode se base essentiellement, sur la modélisation par graphe disjonctif. En effet, une fois les arcs conjonctifs et disjonctifs tracés, l'objectif est de chercher à définir un sens aux arcs disjonctifs afin de former un graphe acyclique minimisant le plus long chemin du sommet 0 au sommet n+1. Pour ce, les auteurs de la méthode propose d'y procéder graduellement en définissant machine par machine le sens des arcs disjonctifs. Il s'agit alors, de définir des ordonnancements partiels en remplaçant certains des arcs disjonctifs par des arcs conjonctifs pour les opérations entre lesquels les précédences ont été décidées.

Par conséquent, à un ordonnancement O (partiel ou complet) est associé un graphe disjonctifs $G_O = (X_O, C_O, D_O)$ ou X_O est l'ensemble des sommets, C_O est l'ensemble des arcs conjonctifs,D_O l'ensemble des arcs disjonctifs. Il faut noter que l'ensemble des arcs conjonctifs peut être décomposé en deux sous ensembles : l'ensemble des arcs conjonctifs initiaux et l'ensemble des arcs conjonctifs qui remplacent les arcs disjonctifs, conformément à l'ordonnancement partiel O. Par conséquent, si le graphe initial qui modélise le graphe est $G = (X, C, D)$, on a $C_O \supseteq C$ et $D_O \subseteq D$.

Une autre remarque importante est que l'ensembles des arcs disjonctifs D peut être décomposée en m sous ensembles distincts, chacun correspondant à une machine, c'est-à-dire :

$D = \cup D(k)$ Ou D(k) est l'ensemble des arcs disjonctifs appartenant à la machine k.

La méthode Shifting Bottleneck est basée sur le constat qu'un ordonnancement consiste à choisir, pour toute machine k, une direction pour tout arc disjonctif appartenant à D(k). Soit S(k) l'ensemble des arcs conjonctifs qui remplacent les arcs disjonctifs D(k), suite à ce choix.

Un ordonnancement complet correspond alors, à un graphe : $[V, C \cup (\cup S(k)), \phi]$.

C'est-à-dire à un graphe dans lequel l'ensemble des arcs qui définissent les successions des opérations sur toutes les machines est fixé.

Si on considère le cas ou les ordres de passage ont été décidés sur un ensemble $M \in \{1, 2, 3, ..., m\}$ de machines. Le graphe disjonctif correspondant s'écrira :

$G(M) = (X, C(M), D(M))$, ou $C(M) = \cup C(k)$ avec $k \in M$ et $D(M) = \cup D(k)$ avec $k \notin M$.

La méthode Shifting Bottleneck consiste à augmenter progressivement l'ensemble M, en introduisant à chaque fois un nouvel élément. La question pertinente qui se pose alors, est de savoir dans quel ordre les machines vont être introduites dans l'ensemble M.

Pour régler cette question cruciale, les auteurs de l'heuristique [ABZ88], ont proposé un algorithme permettant de choisir l'ordre dans lequel les machines sont introduites dans l'ensemble M.

Pour augmenter la probabilité d'obtenir une bonne solution, on réoptimise sur l'ensemble des machine déjà introduites dans M, chaque fois qu'on introduit un nouvel élément. En suivant la même logique, on insère dans l'ensemble M, les machines une par une, on s'arrête lorsque l'ensemble des arcs disjonctifs du graphe soit vide.

Cette décomosition a par la suite, été reprise par plusieurs auteurs, dont nous citons [CP96] et [Tru05].

2.3 Approche d'ordonnacement robuste pour les problèmes à une machine

Dans cette partie, nous décrivons un ordre partiel dominant pour le problème à une machine avec minimisation du retard algébrique maximal, L_{max}. Cet ordre partiel est défini par un théorème de dominance qu'est le théorème des pyramides développé dans les années quatre vingt.

Pour ce, nous présentons quelques notions relatives à ce théorème, puis nous rapportons deux algorithmes développés par Trung [Tru05]permettant de borner respectivement la qualité au mieux et au pire de l'ensemble dominant caractérisé par ce théorème. Cette qualité est déterminée en construisant, pour chaque travail j, la séquence la plus favorable et celle la plus défavorable vis-à-vis du retard algébrique, conduisant respectivement à la détermination des retards algébriques au mieux et au pire.

2.3.1 Notions relatives au théorème des pyramides

(a)Notion d'ensemble dominant de solutions

Selon Esquirol et Lopez [EL99], un sous-ensemble de solutions est dit 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. On peut donc dire qu'une solution dominante est en quelque sorte *"plus admissible"* (ou "plus optimale")

que d'autres [Fon80].

Nous présentons dans ce qui suit des sous ensembles d'ordonnancement particuliers qui présentent des propriétés de dominance vis-à-vis de tout critère régulier, rappelons que :

- Un ordonnancement est dit ***semi actif*** si aucun glissement à gauche local n'est possible sans modifier la séquence sur la ressource : on ne peut pas avancer le début d'une tâche sans modifier la séquence sur la ressource. On peut montrer que l'ensemble des ordonnancements semi actifs est dominant pour tout critère régulier.

- Un ordonnancement est dit ***actif*** si aucun glissement à gauche local et global n'est possible : on ne peut pas commencer une tâche plus tôt sans reporter le début d'une autre. Notons qu'un ordonnancement actif est semi actif et on peut également montrer que l'ensemble des ordonnancements actifs est dominant pour tout critère régulier.

La figure 2.1 illustre ces types d'ordonnancement, étant données cinq tâches 1, 2, 3, 4, 5 et les contraintes de précédence imposées $1 < 2 < 5$ et $3 < 4$.

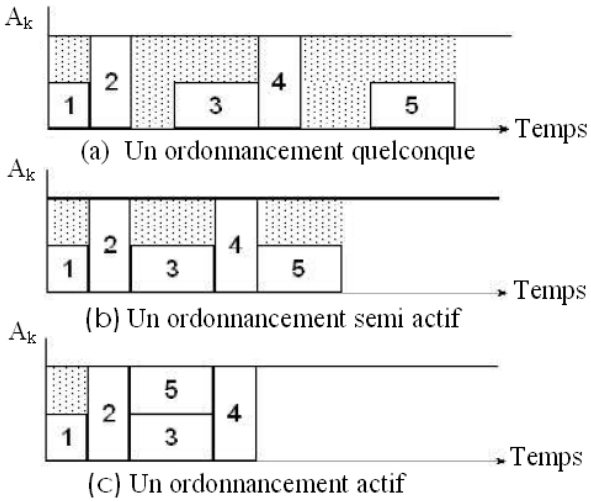


FIG. 2.1 – Différents types d'ordonnancements dominants [EL99]

Comme les ensembles d'ordonnancements semi actifs et actifs sont très vastes, exploiter leur propriété de dominance au sein de procédure d'optimisation ne permet pas de limiter très efficacement l'espace de recherche. C'est pourquoi d'autres méthodes se basant sur l'exploitation de la structure particulière d'un problème pour limiter le cardinal de l'ensemble des solutions admissibles, ont été développées.

Dans ce travail, nous nous intéressons plus particulièrement à la détermination de l'ensemble dominant des solutions à l'aide du théorème des pyramides paru dans [EFMR83]. Pour ce, voici quelques notions qui nous permettent d'aborder ce théorème.

(b)Notion de structure d'intervalles

L'utilisation de structure d'intervalles permet une représentation intéressante des caractéristiques d'un problème. En ordonnancement, une structure d'intervalles est bien adaptée à la représentation de corps d'hypothèses restreint [Tru05]. En effet, les sommets, bases et pyramides d'une structure pouvant être définis sans avoir connaissance des valeurs explicites des paramètres d'un problème.

Une structure d'intervalles est définie par un couple <I,C> où $I = i_1, , i_n$ est un ensemble d'intervalles et C est un ensemble de contraintes sur IxI. Chaque intervalle i_j est défini par un couple de points x_j et y_j tel qu'une relation d'ordre $x_j \prec_R y_j$ quelconque soit vérifiée.

Une contrainte entre deux intervalles i_j et i_k peut être exprimée, en utilisant une des treize relations de l'algèbre de Allen [All91], récapitulées sur la figure 2.2

Remarquons que les six premières relations illustrées sur la figure ont des symétriques, obtenus en inversant les intervalles A et B dans les relations.

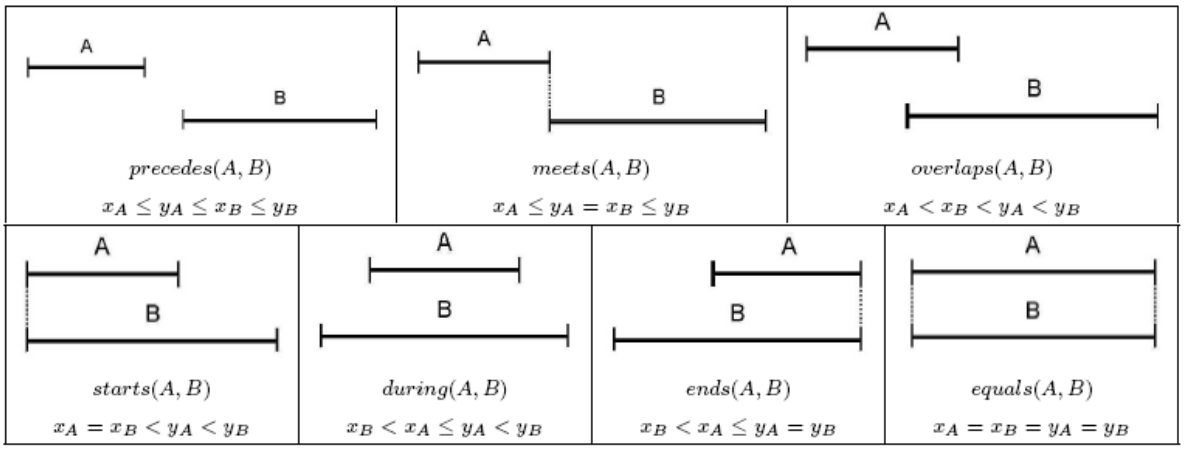


FIG. 2.2 – Algèbre de Allen [All91]

(c)Notions de sommet et de base

Par considération de la relation de Allen *during*, deux types particuliers d'intervalles peuvent être mis en évidence : l'intervalle de type sommet et celui de type base.

Définition 1

Un intervalle s est dit sommet d'une structure d'intervalle <I,C>, s'il n'existe aucun intervalle $i \in I$ tel que *during (i,s)*. C'est à dire, que l'intervalle s vérifie :
$\forall i \in I, r_i \prec r_s \wedge d_i \succ d_s$ [Tru05].

Définition 2

Un intervalle b est dit base d'une structure d'intervalles < I,C> s'il n'existe aucun intervalle $i \in I$ tel que *during(b, i)*. C'est à dire, que l'intervalle b vérifie :
$\forall i \in I, r_i \succ r_s \wedge d_i \prec d_s$.

De même, 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 [EL99].

Définition 3

Une s-pyramide P_s associée au sommet s est le sous ensemble d'intervalles $i \in I$ tel que *during (i,s)*. C'est-à-dire $P_s = \{i \in I / r_i \prec r_s \wedge d_i \succ d_s\} \cup \{s\}$ [Tru05].

Définition 4

Une b-pyramide P_b associée à la base b est le sous ensemble d'intervalles $i \in I$ tel que *during (b,i)*. C'est-à-dire :
$P_s = \{i \in I / r_{is} \wedge d_i \prec d_s\} \cup \{b\}$ [Tru05].

2.3.2 Le théorème des pyramides

Le théorème des pyramides a été formulé dans les années quatre-vingt par [EFMR83] ce, en dehors de tout contexte de recherche de robustesse, dans le but premier est de limiter la complexité algorithmique liée à la recherche d'une solution optimale au sein d'un grand ensemble de solutions. Considérant le problème d'ordonnancement à une machine avec fenêtres d'exécution, les auteurs mettent en évidence une nouvelle condition de dominance pour la recherche de solutions admissibles. Le même auteur dans [EFM85] et bien plus tard dans les travaux de [BHHL05], il est montré que le sous-ensemble de séquences caractérisé par la condition d'Erschler et al. [EFMR83] est également dominant vis-à-vis des critères réguliers d'optimisation, T_{max}, plus grand retard vrai et, L_{max}, plus grand retard algébrique et ce, par considération d'un corps d'hypothèses restreint. C'est-à-dire, un corps d'hypothèses qui ne prend en compte que l'ordre relatif des dates de début au plus tôtr_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 ne sont donc pas considérées). Voici, dans ce qui suit, l'énoncé de ce théorème.

(a)Enoncé du théorème

Un ensemble dominant de séquences peut être constitué par les séquences telles que :
- *Les sommets sont ordonnés dans l'ordre de leur indice ;*
- *avant le premier sommet, seuls sont placés les travaux appartenant à la première pyramide rangés dans l'ordre croissant de leur date de début ou, en cas d'égalité, dans un ordre arbitraire ;*
- *après le dernier sommet, seuls sont placés les travaux appartenant à la dernière pyramide rangés dans l'ordre croissant de leur date de fin ou, en cas d'égalité, dans un ordre arbitraire ;*

– *entre deux sommets* s_k *et* s_{k+1}*, sont placés en premier les travaux appartenant à la pyramide* P_k *et n'appartenant pas à* P_{k+1} *dans l'ordre croissant de leur date de fin (dans un ordre arbitraire en cas d'égalité), puis les travaux communs aux pyramides* P_k *et*P_{k+1}*dans un ordre arbitraire, et enfin les travaux appartenant à la pyramide* P_{k+1} *et n'appartenant pas à* P_k *dans l'ordre croissant de leur date de début (dans un ordre arbitraire en cas d'égalité).*

On note u l'indice de la première pyramide à laquelle une tâche j peut appartenir et v l'indice de la dernière pyramide à laquelle elle appartient.

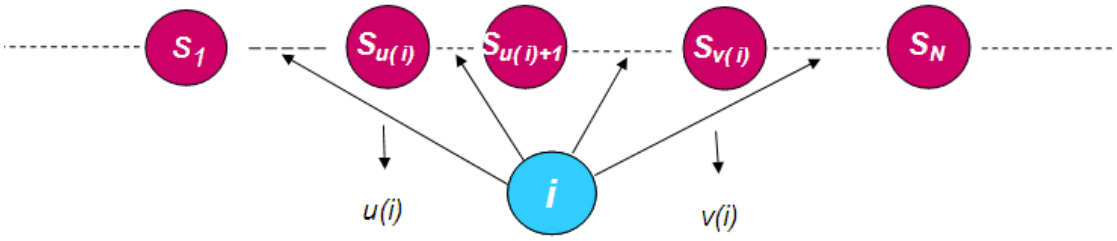


FIG. 2.3 – Séquences engendrées par le théorème des pyramides [LSB05]

Le théorème des pyramides impose des relations de précédence entre les sommets d'une part, puisque $s_k < s_{k+1}$, et entre les travaux non sommets et les sommets d'autre part, puisque $s_{u(j)-1} < j < s_{v(j)+1}$. 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 , exclu aussi certaines séquences.

Ainsi, si deux travaux non sommets i et j, tels que $r_i < r_j$, appartenant à une seule pyramide de sommet s, sont tous deux séquencés avant s alors i< j et la séquence j<i<s est interdite alors qu'elle correspond pourtant à une extension linéaire de l'ordre partiel imposé par le théorème des pyramides [Tru05]. Ce théorème introduit donc, 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.

(b)Cardinalité de l'ensemble dominant

Une propriété intéressante du théorème des pyramides est qu'il permet de mesurer la cardinalité de l'ensemble de séquences dominantes $S_{[}dom$ obtenu lors de l'application du théorème. En effet, celle-ci permet une mesure de la flexibilité séquentielle apportée [Tru05]. Cette cardinalité est calculée grâce à la formule 2.3 .

$$Card(S_{dom}) = \prod_{q=1}^{q=N} (q+1)^{n_q} \tag{2.3}$$

Où n_q désigne le nombre de travaux non sommets 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), cette formule dénombre en fait les affectations différentes des travaux selon s'ils sont placés avant ou après un sommet s_k, tel que u(j)<k<v(j).

Remarquons que cette formule ne prend pas en compte les possibilités de permutation des travaux placés entre deux sommets s_k et s_{k+1} et appartenant en même temps aux s-pyramides P_k et P_{k+1}.

Exemple

A titre illustratif, considérons un problème à une machine dont les dates de début, les dates de fin sont telles que :
$r_2 < r_5 < r_1 < d_1 < d_5 < r_3 < r_4 < d_4 < d_3 < d_2$. Les données de ce problème constituent un corps d'hypothèses restreint, puisque seul l'ordre relatif entre les dates de fin au plus tôt et de fin au plus tard soit donné.

Le diagramme des intervalles et la décomposition en pyramides, associés à cet exemple sont donnés sur la figure 2.4.

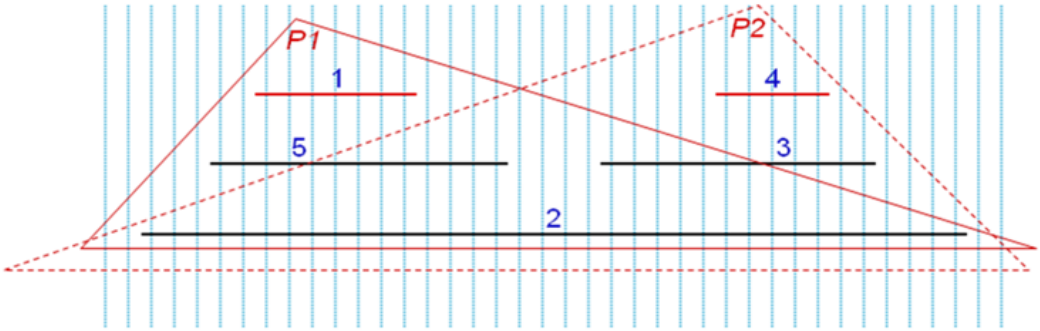


FIG. 2.4 – Diagramme des intervalles sommets et pyramides

L'ordre partiel imposé par le théorème des pyramides permet d'obtenir un ensemble de séquences dominantes schématisées par la figure2.5.

La cardinalité de l'ensemble des séquences dominantes est égale à : $Card(S_{dom}) = 12$. On remarque que le nombre de séquences possibles est de 5 !=120.

Les douze séquences dominantes sont :

2<5<1<3<4, 5<2<1<3<4, 2<5<1<4<3, 1<3<4<2<5, 1<3<4<5<2, 1<4<3<5<2, 5<1<2<4<3, 1<5<2<3<4, 1<5<3<2<4, 1<5<3<4<2, 1<5<3<4<2, 2<1<5<4<3, et 2<1<5<3<4.

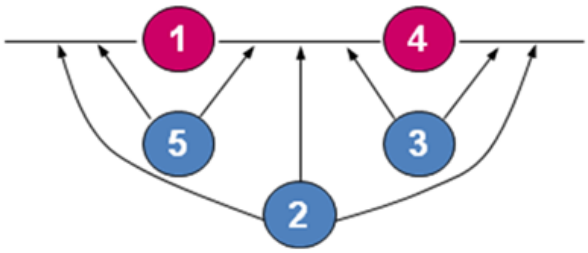


FIG. 2.5 – Séquences engendrées par le théorème des pyramides pour lexemple

2.3.3 Mesure de la qualité d'un ensemble dominant de solution

Du point de vue du retard algébrique, le théorème des pyramides nous permet s'évaluer les performances d'un ensemble dominant. Pour cela, nous nous rapportons à [Tru05] ou une méthode de calcul des valeurs L^j_{min} et L^j_{max} correspondant respectivement au pire et au meilleur retard de j parmi toutes les séquences de S_{dom} a été développée.

Le calcul de ces valeurs est basé sur l'évaluation, pour chaque travail, de la séquence la plus favorable et de la plus défavorable vis-à-vis du retard algébrique.

(a)Calcul du retard au mieux d'un travail

Intéressons nous tout d'abord au calcul deL^j_{min} . De façon évidente, le but étant de minimiser la longueur du chemin critique associé au travail j, seuls les travaux nécessairement séquencés avant j (en cohérence avec le théorème des pyramides) sont à considérer (i.e. moins le nombre de travaux placés avant j est important et plus le retard de j sera petit).

Pour cette raison, nous supposons que j est affecté à la pyramide d'indice u(j) et que seul l'ensemble $Pred^j_{min}$ des travaux k tels que v(k) $<$ u(j) est pris en compte.

Dans la pyramide d'indice u(j), il est toujours possible de séquencer j en première position. Déterminer L^j_{min} revient alors à minimiser la durée totale C_{max} du problème d'ordonnancement constitué des travaux de l'ensemble $Pred^j_{min}$.

Le retard des travaux de $Pred^j_{min}$ n'étant pas significatif pour le calcul de L^j_{min} , on fixe arbitrairement leur date de fin à la date d_j. La structure d'intervalles de $Pred^j_{min}$ ainsi obtenue définissant une structure en escalier, une séquence $Pred^j_{min}$ optimale (pour minimiser le C_{max}) est obtenue par l'application de la règle de Jackson séquencant les travaux dans l'ordre croissant de leurs dates de début. Remarquons que cette règle de construction de respecte le théorème des pyramides puisqu'elle affecte chaque travail k appartenant à $Pred^j_{min}$ à la pyramide u(k).

Soient $S_k^{Pred^{min}_j}$ les dates de début des travaux $k \in Pred^{min}_j$ dans et $C_{max}^{Pred^{min}_j}$ le Makespan correspondant, on a :

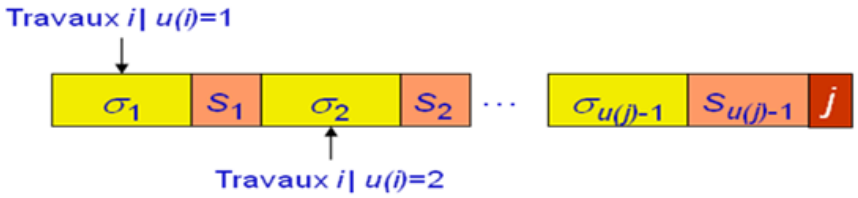


FIG. 2.6 – Construction de la meilleure séquence pour j [Tru05]

$$L_j^{min} = \max(C_{max}^{Pred_j^{min}}, r_j) + p_j - d_j \tag{2.4}$$

Tel que $\forall k \in Pred_j^{min}$:

$$C_{max}^{Pred_j^{min}} = \max(S_k^{Pred_j^{min}} + p_k) \tag{2.5}$$

A partir de cette définition de la séquence au mieux d'une tâche, Trung [Tru05], établit un algorithme permettant de calculer en un temps polynomial le retard algébrique au mieux de chaque tâche.

Cet algorithme est donné par la figure 2.7. Nous allons voir ultérieurement comment utiliser cet algorithme pour déduire la date de début au mieux d'une tâche.

(b) Calcul du retard au pire d'un travail

Considérons à présent le calcul de L_j^{max}. De façon symétrique au cas précédent, afin de maximiser le retard du travail j, celui-ci est supposé affecté à la pyramide d'indice v(j), c'est-à-dire le plus tard possible. D'autre part, les travaux k devant nécessairement être séquencés après j (i.e. les travaux k tels que u(k) $>$ v(j)) ne sont pas considérés. Nous notons $Pred_j^{max}$ l'ensemble des travaux restants.

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.

Dans [Tru05], il a été prouvé que pour maximiser le 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 inférieure 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 S_j^1 , dans laquelle les travaux sont séquencés par ordre de d_i croissant. En effet, de cette façon, le sommet est toujours placé en tête de la sous

Proc CalculerLjMin(j, T) $(*)$
$Pred_j^{\min} = \emptyset$;

Pour chaque $i \in T$ faire
Si $v(i) < u(j)$ alors $Pred_j^{\min} \leftarrow Pred_j^{\min} \cup i$; FinSi
FinPour

Pour chaque $i \in Pred_j^{\min}$ faire
Affecter i à la pyramide d'indice $u(i)$;
FinPour

Pour chaque pyramide P_k, telle que $k < u(j)$ faire
$Seq_{P_k}^{\min} \leftarrow$ travaux séquencés par ordre croissant des r_i ;
FinPour

$Seq_j^{\min} \leftarrow Seq_{P_0}^{\min} \prec \cdots \prec Seq_{P_{u(j-1)}}^{\min} \prec j$
Calculer $L_j^{\min}$ pour la séquence $Seq_j^{\min}$;
FinProc

$(*)$ Dans cet algorithme, $Pred_j^{\min}$ est l'ensemble de travaux devant précéder j.
T est l'ensemble de travaux à ordonnancer.

FIG. 2.7 – Algorithme de calcul du retard au mieux d'un travail [Tru05]

séquence, puisqu'il possède par définition la date d'échéance di la plus petite.

Considérons à présent les travaux k tels que $u(k) \leq v(j) \leq v(k)$. Pour maximiser le retard de j, nous affectons ces travaux là à la pyramide d'indice v(j) (celle de j), car plus le nombre de travaux placés avant j est important et plus ceux-ci sont ordonnancés tard, plus le retard algébrique de j sera important. Il a été démontré dans [Tru05] que la séquence respectant le théorème des pyramides, qui place j en dernier dans la pyramide v(j), et qui possède le C_{max} le plus grand, est la séquence qui place tous les travaux k tels que $d_k > d_j$ avant le sommet (bloc A), et tous les travaux restants après le sommet $S_{V(j)}$ (bloc B).

Notons Seq_j^{max}la juxtaposition des deux sous séquences S_j^1 et S_j^2 précédemment déterminées, et $C_{max}^{Pred_j^{max}}$ le Makespan correspondant. Nous avons alors $\forall k \in Pred_j^{max}$:

$$C_{max}^{Pred_j^{max}} = \max(S_k^{Pred_j^{max}} + p_k) \tag{2.6}$$

La connaissance des retards au mieux et au pire de chaque travail permet de déduire les bornes inférieure et supérieure pour le retard algébrique L_{max}^* optimal, et ce $\forall j \in T$:

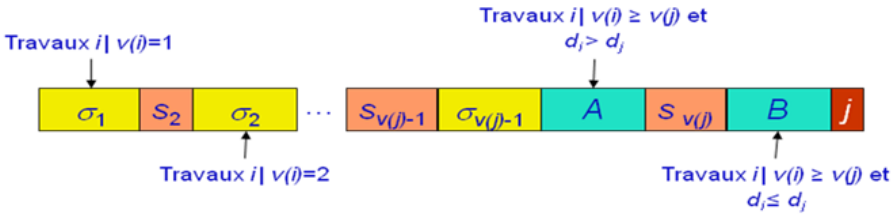


FIG. 2.8 – Construction de la séquence la plus défavorable pour j [Tru05]

$$\max(L_j^{min}) \leq L_{max}^{*} \leq \max(L_j^{max}) \tag{2.7}$$

Il est donc possible de juger si un ensemble dominant de séquences est acceptable ou non, relativement à la performance au pire. Le calcul des valeurs de L_i^{min} et L_i^{max} permettent ainsi, de déduire les valeurs des dates de début au mieux et au pire S_i^{min} et S_i^{max} de chaque tâche est donné comme suit :

$$S_i^{min} = L_i^{min} + d_i - p_i \tag{2.8}$$

$$S_i^{max} = L_i^{max} + d_i - p_i \tag{2.9}$$

2.3.4 Exemple

Pour illustrer le théorème des pyramides, nous traitons un exemple illustratif extrait de [CP96] :

Le problème considère le cas de 3 machines, et 10 opérations réparties suivant 4 gammes opératoires comme indiqué sur le tableau 2.1.

Travail	**tâche**	p_i	M_j
1	1	7	1
1	2	10	2
1	3	5	3
2	4	3	2
2	5	15	1
3	6	30	3
3	7	3	1
3	8	2	2
4	9	6	3
4	10	3	1

TAB. 2.1 – Données de l'exemple

Le graphe conjonctif de cet exemple est donné par la figure 2.10 dans laquelle nous avons représenté les arcs conjonctifs par un trait plein et les arcs disjonctifs par des pointillés. Par souci de clarté, nous nous sommes contenté de représenter sur la figure les arcs

Proc CalculerLjMax(j,T) $(*)$
$Pred_j^{S_1} = \emptyset$;
$Pred_j^{S_2} = \emptyset$;
$A = \emptyset$;
$B = \emptyset$;

Pour chaque $i \in T$ faire
Si $v(i) < v(j)$ alors $Pred_j^{S_1} \leftarrow Pred_j^{S_1} \cup i$; FinSi
Si $u(i) \leq v(j) \leq v(i)$ alors $Pred_j^{S_2} \leftarrow Pred_j^{S_2} \cup i$; FinSi
FinPour

Pour chaque $i \in Pred_j^{S_1}$ faire
Affecter i à la pyramide d'indice $v(i)$;
FinPour

Pour chaque pyramide $P_k, k < v(j)$ faire
$Seq_{P_k}^{\max} \leftarrow$ travaux séquencés par ordre croissant des d_i ;
FinPour

$S_1 \leftarrow Seq_{P_0}^{\max} \prec \cdots \prec Seq_{P_{v(j)-1}}^{\max}$

Pour chaque $i \in Pred_j^{S_2} - \{s_{v(j)}\}$ faire
Si $d_i > d_j$ alors
$A \leftarrow A + \{i\}$;
SiNon
$B \leftarrow B + \{i\}$;
FinSi
FinPour

Séquencer A par ordre croissant des r_i ;
Séquencer B par ordre croissant des d_i ;
$S_2 \leftarrow A \prec s \prec B \prec j$

$Seq_j^{\max} \leftarrow S_1 \prec S_2$;
calculer $L_j^{\max}$ sur $Seq_j^{\max}$;
FinProc
$(*)$ Dans cet algorithme, T est l'ensemble de travaux à ordonnancer. j est le travail dont le retard au pire sera calculé par l'algorithme. $Pred_j^{S_1}$ est l'ensemble de tous les travaux devant être placés avant la pyramide $P_{v(j)}$. $Pred_j^{S_2}$ est l'ensemble de travaux placés dans la pyramide $P_{v(j)}$. A (resp. B) est l'ensemble des travaux de $Pred_j^{S_2}$ placés avant (resp. après) $s_{v(j)}$.

FIG. 2.9 – Algorithme de calcul du retard au pire d'un travail [Tru05]

disjonctifs relatifs à la contrainte de capacité de la machine 2.

Pour définir la qualité de l'ensemble dominant des solutions, Le problème job shop traité peut être décomposé en 3 sous problèmes à une machine dont les dates de début au mieux et au pire $[r_i\ d_i]$ des tâches sont calculées par application de l'algorithme de Ford Bellman donné par la section 2.1, et sont précisées sur la figure2.11.

Pour chaque problème à une machine M_k, l'utilisation du théorème des pyramides permet de déterminer l'ensemble des séquences dominantes S^{Vk} et de déduire, d'après les valeurs de L_i^{min} et L_i^{max}, les dates de début au mieux S_i^{min}(resp. au pire S_i^{max}) ainsi que les dates de fin au mieux S_i^{min} (resp. au pire S_i^{max}). L'utilisation de ce théorème sur notre exemple donne les résultats illustrés par le tableau 2.2. On remarque que seule la machine M_1 possède une flexibilité séquentielle puisque la structure d'intervalles qui lui est associée permet de caractériser 4 séquences dominantes, les autres machine n'en définissent qu'une séquence unique .

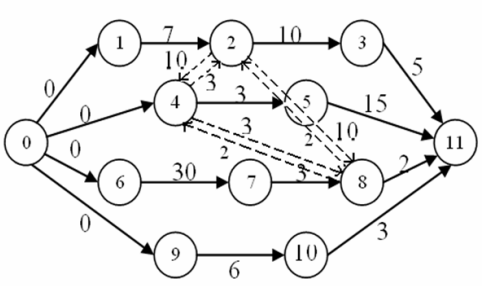


FIG. 2.10 – Graphe disjonctif correspondant à lexemple du tableau 2.1

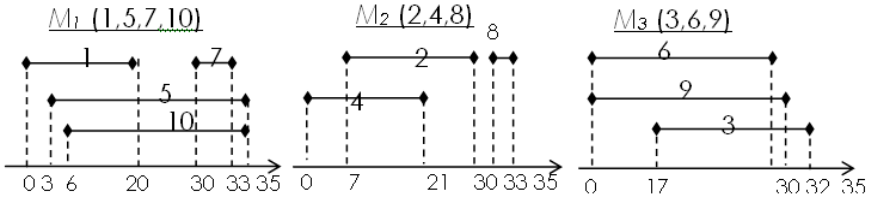


FIG. 2.11 – Structures d'intervalles de l'exemple

2.4 Insensibilité d'un ensemble dominant de solution aux perturbations

2.4.1 Vers un modèle d'incertitudes par intervalles

Rappelons qu'une propriété intéressante du théorème des pyramides est sa relative insensibilité aux variations des dates de début et de fin de travaux dans la mesure où l'ordre relatif entre les dates est conservé. En effet, étant donné un problème V , S_{dom} étant l'ensemble de séquences dominantes défini par le théorème des pyramides, l'ensemble S_{dom} reste dominant tant que l'ordre relatif des r_i et d_i est conservé, cela quelles que soient les valeurs des durées opératoires.

De plus nous venons de présenter dans la partie 2.3.3, qu'il était possible de déterminer pour chaque travail, en temps polynomial, la séquence la plus favorable et celle la plus défavorable vis-à-vis du retard algébrique. La détermination de ces séquences ne requiert également que la connaissance de l'ordre relatif des dates de début au plus tôt et de fin au plus tard des travaux.

Trung dans [Tru05] énonce La propriété suivante :

Proposition

Étant donné un problème à une machine V dont les scénarios potentiels de réalisation sont caractérisés par un modèle par intervalles de type $r_i \in [\overline{r}_i\, \underline{r}_i]$,$p_i \in [\overline{p}_i\, \underline{p}_i]$, *et* $d_i \in [\overline{d}_i\, \underline{d}_i]$ *et tel que tous les intervalles associés aux* r_i *et* d_i *sont disjoints, le théorème des pyramides caractérise un ensemble dominant de séquences* S_{dom} *pour lequel il est possible de déterminer pour chaque travail, en temps polynomial, les performances au mieux et au pire*

M_k	M_1	M_2	M_3
S^{Vk}	(1,5,10,7) (1,5,7,10) (1,7,5,10) (1,10,7,5)	(4,2,8)	(9,6,3)
$[S_i^{min}, f_i^{min}]$ $[S_i^{max}, f_i^{max}]$	i=1 :[0,7] ;[0,7]. i=5 :[7,22] ;[33,48]. i=7 :[30,33] ;[30,33]. i=10 :[7,10] ;[48,51].	i=4 :[0,3] ;[0,3]. i=2 :[7,17] ;[7,17]. i=8 :[33,35] ;[33,35].	i=9 :[0,30] ;[0,30]. i=6 :[30, 36] ;[30, 36]. i=3 :[36, 41] ;[36, 41].

TAB. 2.2 – Séquences dominantes et dates de début au mieux et au pire

vis-à-vis du retard algébrique sur l'ensemble des scénarios de réalisation. Quel que soit le scénario de réalisation considéré, compatible avec le modèle par intervalles, l'ensemble S_{dom} *contient toujours une solution optimale.*

Démonstration

Nous reprenons la démonstration donnée par Trung [Tru05]. En effet, l'auteur affirme que faire l'hypothèse que tous les intervalles associés aux r_i et d_i sont disjoints permer de garantir qu'il existe une relation d'ordre total entre ces paramètres et que le théorème des pyramides peut être appliqué.

Soit S_{dom}l'ensemble dominant obtenu. Étant dominant, il contient bien entendu une solution optimale quel que soit le scénario de réalisation considéré dans la mesure où l'ordre relatif des dates de début au plus tôt et de fin au plus tard est inchangé. De plus, il est possible de déterminer dans S_{dom}, pour chaque travail j, la séquence la plus favorable Seq_j^{min} et celle la plus défavorable Seq_j^{max} vis-à-vis du retard algébrique, ainsi que décrit dans la section 2.3.3, cela indépendamment des valeurs de r_i, p_i et d_i. Connaissant Seq_j^{min} , la valeur de L_j^{min} peut être calculé grâce à l'algorithme de la figure 2.7, en fixant pour chaque travail : $r_j = \underline{r}_j$, $p_j = \underline{p}_j$, et $d_j = \underline{d}_j$.

Similairement, la valeur de L_j^{max}, connaissant Seq_j^{max}, peut être calculé grâce à l'algorithme de la figure 2.9, en fixant pour chaque travail $r_j = \overline{r}_j$, $p_j = \overline{p}_j$, et $d_j = \overline{d}_j$.

Cette propriété est importante car elle permet de garantir que les performances d'un ensemble de séquences dominantes, caractérisé grâce au théorème des pyramides, sont robustes vis-à-vis d'un ensemble de scénarios potentiels de réalisation d'un ordonnancement.

Afin d'illustrer cette propriéte, reprenons l'exemple proposé par Trung [Tru05], Il s'agit d'un problème de 4 tâches décrit par le tableau 2.3. L'ensemble des tâches T = $\{1; 2; 3; 4\}$ caractérise la structure d'intervalles mono-pyramidale représentée sur la figure 2.12, la tâche 1 étant l'unique sommet.

0n associe à chaque tâche deux intervalles d'incertitude $[\overline{r}_i \underline{r}_i]$, $[\overline{d}_i \underline{d}_i]$, de telle manière à ce qu'ils soient disjoints. Cette propriété est indisensable à l'application du théorème des pyramides. Un ensemble dominant de séquences peut alors être déduit possédant des performances au mieux et au pire connues, calculées ainsi que définit dans la proposition 2.4.1.

tâche	r_i	d_i	p_i
1	10	15	3
2	8	19	5
3	2	17	6
4	4	23	7

TAB. 2.3 – Une instance de quatre tâches

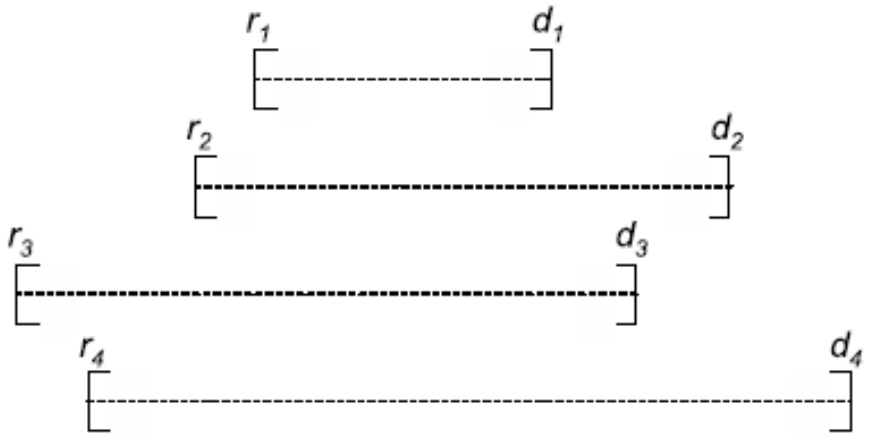


FIG. 2.12 – Structure d'intervalles du problème considéré

Remarquons de plus qu'un intervalle de durée $[\overline{p}_i\ \underline{p}_i]$ quelconque peut éventuellement être pris en compte à ce stade, sans perte de généralité. Les performances obtenues pour l'ensemble dominant sont alors *robustes* vis-à-vis de tous *les scénarios capturés par le modèle par intervalles*.

tâche	$[\overline{r}_i\ \underline{r}_i]$	$[\overline{d}_i\ \underline{d}_i]$
1	[10 12]	[13 16]
2	[7 9]	[19 21]
3	[2 3]	[17 18]
4	[4 6]	[22 23]

TAB. 2.4 – Modèle par intervalles de l'exemple traité

Ce modèle par intervalles caractérise card(ES) = 2592 scénarios de réalisation. Le théorème des pyramides caractérise quant à lui $(1+1)^3 = 8$ séquences dominantes. Le calcul des performances au mieux se fait en utilisons les valeurs $r_j = \underline{r}_j$, p_j, et $d_j = \underline{d}_j$, et Le calcul des performances au pire se fait en utilisons les valeurs $r_j = \overline{r}_j$, p_j, et $d_j = \overline{d}_j$.

On peut alors, assurer que ces performances sont robustes vis-à-vis de tout scénario de réalisation possible, respectant le modèle par intervalles.

2.4.2 Le compromis robustesse/performance

Esswein dans [Ess03], souligne que pour atteindre un objectif de robustesse il peut être intéressant de proposer statiquement des solutions qui soient flexibles, ceci afin de laisser le plus de liberté possible pour réagir lors de la phase dynamique. La robustesse de la solution dépendrait alors non seulement de la quantité de flexibilité proposée, mais en plus, elle dépend du couplage entre cette quantité et la qualité de l'algorithme dynamique chargé de " bien utiliser " cette flexibilité. Un compromis doit alors être cherché entre la flexibilité de la solution proposée statiquement et sa qualité en moyenne ou dans le pire des cas. Ceci a pour conséquence directe, l'apparition du besoin de mesurer quantitativement la flexibilité.

Pour ce, Esswein [Ess03] propose de considérer la cardinalité de l'ensemble d'ordonnancements établis statiquement comme mesure de flexibilité.

En suivant le même raisonnement, nous pouvons utiliser le cardinal des solutions admissibles que défnit le théorème des pyramides (Cf.2.3) comme une mesure de la fléxibilite d'une structure d'intervalle. Une solution non flexible ne comporterait qu'un ordonnancement possible alors qu'une solution très flexible pourrait autoriser de nombreux choix de séquencements, et/ou d'affectations définissant ainsi un ensemble d'ordonnancements comportant plusieurs éléments.

Il est vrai qu'offrir de la fléxibilité permet de mieux appréhender les perturbations, cependant il est synônyme de baisse de la performence.

La définition d'un bon compromis robustesse/performance est une tâche délicate qui repose sur une étude statistique poussée. Au même temps, elle est d'une importance capitale car conditionnant la qualité de la solution trouvée, elle permet de fournir un meilleur jugement de valeur.

Cette tâche ne fait pas objet de nôtre travail, elle faira objet d'études ultérieures.

Dans le présent travail, nous allons contourner cette tâche, et nous n'allons pas définir ce compromis. Pour juger la qualité de la solution proposée par notre heuristique nous nous contentons de nous fixer un seuil de flexibilité et un seuil de performance à ne pas dépasser.

Ces seuils seront définis dans le troisième chapitre dans la section intitulée *Objectifs et hypothèses de travail* 3.2.1).

2.5 Evaluation des marges libres présentes dans la séquence au pire d'une tâche j

Afin d'aboutir à une solution au problème de l'arrivée en temps réel d'un nouveau job composé de plusieurs tâches (qui altéreront plusieurs structures d'intervalles), considérons

tout d'abords le cas d'un job mono tâche. De cette manière, une seule structure d'intervalle change, et le problème revient à trouver la déviation maximale du L_{max} de chaque tâche de cette structure d'intervalle ainsi que la déviation que subit la fin des autres jobs.

Soit une tâche j quelconque de la structure d'intervalle concernée. L'utilisation de la séquence au pire de la tâche j 2.8 nous permet d'avoir ***un seuil supérieur de détérioration maximal*** de sa date de fin, suite à l'arrivée d'une nouvelle tâche en temps réel.

De cette manière, on se met dans le pire scénario pour la tâche j, puisqu'on la pousse à s'exécuter le plus tard possible et on fixe un intervalle d'exécution pour la nouvelle tâche tout en étant sûrs que le décalage de la date de fin de la tâche j ne dépassera jamais ce seuil.

Afin de pouvoir décider à quelle pyramide affecter la nouvelle tâche en altérant le moins possible la date de fin de la dernière tâche du job auquel appartient la tâche j, il serait intéressant de trouver dans la séquence au pire de j des espaces qui pourraient contenir la nouvelle tâche. En plus de ça, étant donné que la tâche j peut ne pas être la dernière tâche de son job, il faudra aussi prendre en considération la marge libre qui existe entre j et la dernière tâche de son job.

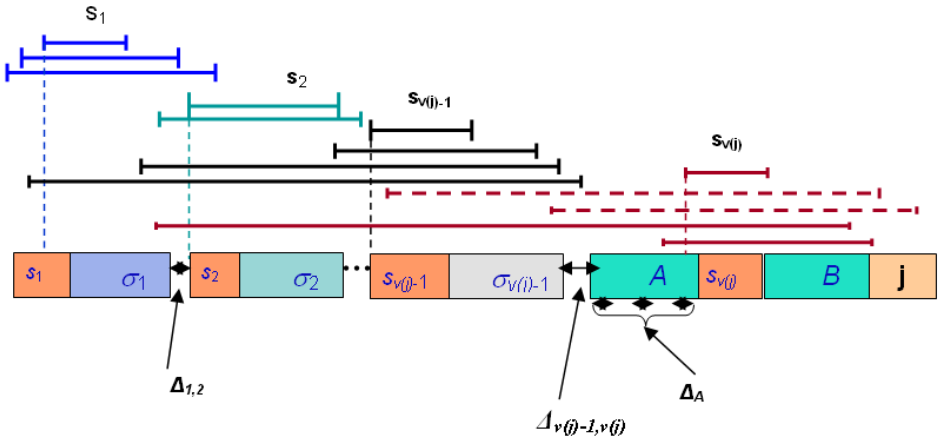


FIG. 2.13 – Ecarts disponibles dans la séquence au pire de j

Nous remarquons, tout d'abords qu'entre la dernière tâche de la séquence $\sigma_v(j)$ et le sommet $S_{v(j)+1}$, il existe un espace libre qu'on note $\Delta_{v(j),v(j)+1}$ tel que :

$$\Delta_{v(j),v(j)+1} = \max_{1(i)\leq j-2}\left(0, S_{S_{v(j)+1}} - \max_{\sigma_j} f_{\sigma_j}\right) \tag{2.10}$$

Tel(f_{σ_i}) que est la plus grande date de fin du bloc v(j), et est la date de début du sommet appartenant à la prochaine pyramide. Cette formule est valable uniquement pour les j-2 premières pyramides.

Ainsi, la marge libre présente entre le sommet de la première pyramide et la fin de la dernière tâche du bloc $\sigma_v(j) - 1$ est :

$$\Delta_{1,v(j)-1} = \sum_{v(j)-2}^{v(i)=1} \Delta_{v(j),v(j)+1} \tag{2.11}$$

Concernant l'écart existant entre la pyramide d'indice v(j)-1 et le bloc A, il serait égal à :

$$\Delta_{v(j)-1,A} = \max\left(0, S_{S_A} - \max_{\sigma_j} f_{\sigma_j}\right) \tag{2.12}$$

Tel que S_A est la date de début de la première tâche du bloc A.

De même, dans le sous-ensemble A, les tâches sont classées selon l'ordre croissant des r_i. Du coup, il serait possible qu'il existe, entre certaines tâches, des marges libres. On note Δ_A la somme de ces espaces libres telle que :

$$\Delta_A = \sum_{i \in A} (f_{i+1} - s_i) \tag{2.13}$$

En plus de ces deux marges, considérons les autres tâches du job auquel j appartient. Les contraintes de précédence entre ces tâches et l'état de disponibilité des différentes ressources font qu'il soit possible de trouver des marges libres entre les différentes tâches et la dernière tâche du job vu que sa date de fin coïncide avec la fin du job tout entier.

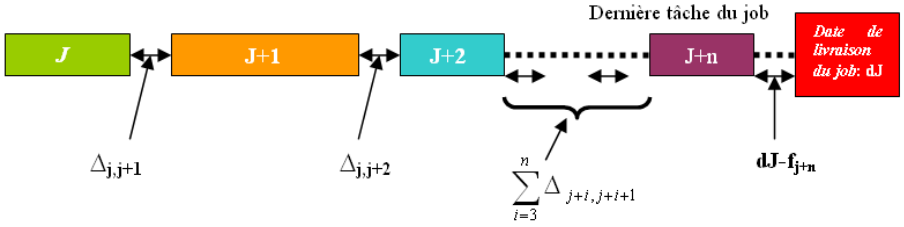


FIG. 2.14 – Ecarts disponibles entre la tâche j et la dernière tâche de son job

On exprime la marge libre existante entre j et la dernière tâche du job auquel elle appartient, par :

Tel que n est le nombre de tâches qui existent entre j et la dernière tâche du job de j et dJ est la date de livraison du job.

$$\Delta_{j,j+n} = \max_{i=1}^{n} (0, S_{j+i} - f_{j+i-1} + dJ - f_{j+n}) \tag{2.14}$$

Finalement, la marge libre dont dispose la tâche j est égale à :

$$\Delta_j = \Delta_{j,j+n} + \Delta_{v(j)-1,A} + \Delta_{1,v(j)-1} \tag{2.15}$$

2.5.1 Définition d'une fênetre d'execution [r_β, d_β] pour une tâche β

On appelle β la nouvelle tâche à ordonnancer, sa durée d'exécution est p_β, elle est disponible à partir de l'instant r_β, et on cherche à déterminer l'intervalle $[r_\beta, d_\beta]$ qui affecte le moins possible la structure d'intervalle déjà existante.

On considère d'abords le cas ou j n'est pas un sommet puis, celui ou elle en est un. En plus, dans tous les cas on considère que $r_\beta + p_\beta \leq d_\beta$.

L'idée proposée est de retrouver les séquences au pire de toutes les tâches de la structure d'intervalle concernée. Pour chaque séquence au pire d'une tâche j, on fixe la valeur de r_β à la date de disponibilité de β. Concernant la valeur de d_β, on commence par prendre une valeur telle que $d_\beta > d_j$, de cette manière, la tâche β va appartenir à la pyramide $P_{V(j)}$, on compare la durée opératoire de β à la marge libre qu'offre éventuellement cet emplacement. Si p_β est supérieure à cette marge libre il faudra chercher un autre emplacement pour la tâche β de manière à ce que son insertion affecte le moins la tâche j.

Pour ce, on fait diminuer d_β de manière à faire appartenir la tâche β à la pyramide d'indice inférieur, et on recommence la procédure (comparaison de p_β à la marge libre disponible et diminution de d_β). On s'arrête lorsqu'on trouve un emplacement qui assure une marge libre assez grande pour absorber la tâche β, ou bien lorsque r_β est trop grand pour que la tâche β soit affectée à une pyramide. Dans ce cas là, la tâche β devient elle-même un sommet et la déviation qu'elle imposera à la tâche j sera minimale.

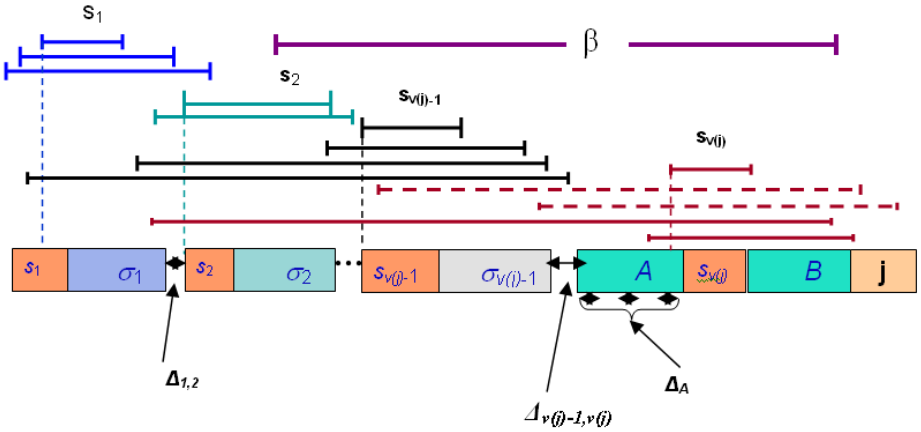


FIG. 2.15 – définition de l'intervalle de la tâche β

Cas I : *j n'est pas un sommet*

Dans ce qui suit on ne considère que les pyramides dont les sommets ont des dates de début telles que : $r_{S_{v(j)}} > r_\beta$.

- ***Cas 1*** $d_\beta > d_j$ **:**β se placera sûrement dans la même pyramide que j, dans le bloc A, dans l'ordre croissant des r_i. soit la tâche i du bloc A telle que $r_i \geq r_\beta$, on

appelle $\Delta_A^{'}$ la marge libre qui existe entre la fin de la tâche i et la sommet $S_{v(j)}$: $\Delta_A^{'} = \Delta_A - \sum_i^{l=1} (f_{l+1} - s_l)$. Voyons alors l'impacte de cet emplacement sur j :

- Si $\Delta_A^{'} + \Delta_{j,j+n} > p_\beta$: la tâche j n'est pas affectée par l'insertion de β, l'emplacement est intéressant et à retenir.

- Sinon, la déviation de la tâche j est $\sigma_j = \max\left(0, p_\beta - \Delta_A^{'} - \Delta_{j,j+n}\right)$. Et il faudra diminuer d_β encore plus pour que β puisse jouir d'une marge plus importante.

- ***Cas 2*** $d_\beta < d_j$ ***:*** β se placera sûrement dans la même pyramide que j, dans le bloc B, dans l'ordre croissant des d_i.

  - Si $\Delta_{j,j+n} > p_\beta$: la tâche j n'est pas affectée par l'insertion de β, l'emplacement est retenu.

  - Sinon, j va subir une déviation égale à : $\sigma_j = \max(0, p_\beta - \Delta_{j,j+n})$. Dans ce cas, il faudra diminuer d_β encore plus pour que β puisse jouir d'une marge plus importante.

- ***Cas 3*** $d_{S_{v(j)-1}} < d_\beta < d_{S_{v(j)}}$ ***:*** β se placera sûrement dans la pyramide d'indice v(j)-1, dans le bloc v(j)-1 dans l'ordre des d_i croissants. L'impacte de cet emplacement sur la tâche j est comme suit :

  - Si $\Delta_A + \Delta_{j,j+n} + \Delta_{v(j)-1,A} > p_\beta$: la tâche j ne subira aucune déviation. Cet emplacement serait alors retenu.

  - Sinon, la déviation de la tâche j est :
    $\sigma_j = \max\left(0, p_\beta - \Delta_A - \Delta_{j,j+n} - \Delta_{v(j)-1,A}\right)$. Et il faudra diminuer d_β encore plus, pour que β puisse jouir d'une marge plus importante.
- ***Cas 3*** $d_{S_{v(j)-2}} < d_\beta < d_{S_{v(j)-1}}$ ***:*** β se placera sûrement dans la pyramide d'indice v(j)-2, dans le bloc $\sigma_{v(j)-2}$ dans l'ordre des d_i croissants. L'impacte de cet emplacement sur la tâche j est comme suit :

  - Si $\Delta_A + \Delta_{j,j+n} + \Delta_{v(j)-1,A} + \Delta_{v(j)-2,v(j)-1} > p_\beta$:la tâche j ne subira aucune déviation suite à l'insertion de β. Cet emplacement est à retenir.

  - Sinon, la fin de la tâche j sera retardée de :
    $\sigma_j = \max\left(0, p_\beta - \Delta_A - \Delta_{j,j+n} - \Delta_{v(j)-1,A} - \Delta_{v(j)-2,v(j)-1}\right)$. Et il faudra diminuer davantage d_β .

Et ainsi de suite, on remarque que plus on diminue d_β plus l'indice de la pyramide à laquelle j va appartenir, serait petit et plus grande serait la marge libre qui pourrait absorber β. On s'arrêtera de diminuer d_β lorsque un des deux cas suivants se présente :

1. $d_\beta < p_\beta + r_\beta$: dans ce cas, on fixe $d_\beta = p_\beta + r_\beta$ et on se contente d'une déviation sur j de :
   $\sigma_j = \max\left(0, p_\beta - \Delta_A - \Delta_{j,j+n} - \Delta_{v(j)-1,A} - \Delta_{v(j)-2,v(j)-1} - ... - \Delta_{i,i+1}\right)$, telle que i

est l'indice de la dernière pyramide à laquelle la tâche β peut appartenir.

2. r_β devient supérieur à toutes les dates de début des sommets des pyramides restantes. Dans ce cas, la tâche β devient elle-même un sommet, et on se contente d'une déviation de la date de fin de la tâche j égale à :
$\sigma_j = \max\left(0, p_\beta - \Delta_A - \Delta_{j,j+n} - \Delta_{v(j)-1,A} - \Delta_{v(j)-2,v(j)-1} - ... - \Delta_{k,k+1}\right)$, tel que k est l'indice de la dernière pyramide à laquelle la tâche β peut appartenir.

Cas II : *j est un sommet*

On procèdera exactement de la même manière que pour le cas précédant, sauf que l'emplacement $d_\beta < d_j$ n'est pas significatif pour la tâche β car elle ne pourra pas être placée dans la même pyramide que j. Dans cette situation, on sera ramené au cas : $d_{S_{v(j)-1}} < d_\beta < d_{S_{v(j)}}$.

A la fin on obtient un ensemble d'intervalles possibles pour la tâche β. Chaque proposition d'intervalle engendre éventuellement, des conséquences sur les dates de fin des autres jobs, il faudrait alors trouver un compromis entre le respect de la date de livraison du nouveau job et celles des autres jobs.

2.5.2 Propriétés

A partir de tout cela, nous pouvons d'ores et déjà dégager les propriétés suivantes :

Propriété 1 :

Pour calculer la déviation maximale que subit une tâche j suite à l'insertion d'une nouvelle tâche β, il est plus intéressant de fixer r_β et de faire varier d_β plutôt que de faire l'inverse.

Démonstration :

Soit la séquence au pire d'une tâche j, on dispose d'une nouvelle tâche β à insérer dans la structure d'intervalle. Pour ce fixant la valeur de $d_\beta < d_j$ et variant la valeur de r_β.

Pour les trois positions nous avons la même valeur de d_β mais des valeurs différentes de r_β. Mais, nous remarquons que pour les trois cas, la tâche β va toujours appartenir au bloc B. Donc la seule marge libre dont elle dispose serait $\Delta_{j,j+n}$. On agissant ainsi, on va pousser la tâche β toujours dans la même pyramide que j et on ne pourra pas profiter du reste des marges libres disponibles dans la séquence au pire de la tâche j.

Donc, il est inutile de fixer d_β et de faire varier r_β. L'inverse par contre nous permet de faire appartenir la tâche β à plusieurs pyramides de la structure d'intervalle et de profiter de toutes les marges libres disponibles.

Propriété 2 :

Si pour une valeur de $d_\beta < d_j$, j n'est pas affectée par l'insertion de la tâche β, tous les autres intervalles de β obtenus en diminuant la valeur de d_β ou en l'augmentant ($d_\beta > d_j$) n'affecteront pas la tâche j.

De la même manière, si pour une valeur de $d_\beta < d_j$, j n'est pas affectée par l'insertion de la tâche β, tous les autres intervalles de la tâche β obtenus en diminuant la valeur de d_β n'affecteront pas la tâche j.

Démonstration :

Soit la séquence au pire d'une tâche j. Dans la figure 2.16 nous retrouvons trois positions possibles pour d_β :

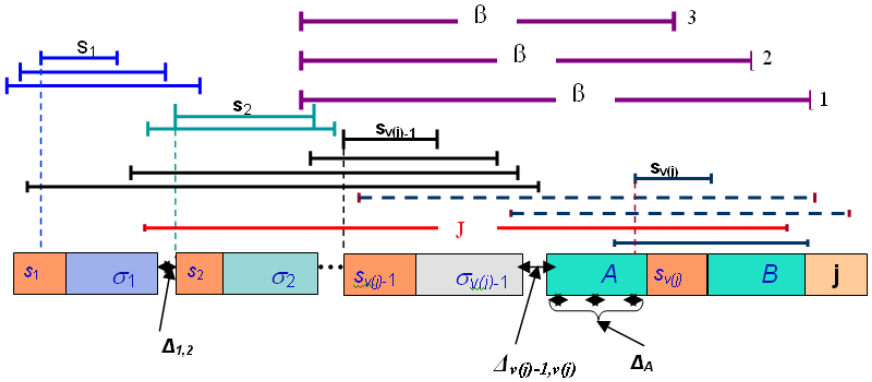


FIG. 2.16 – Positions possibles pour la tâche β

Pour la position 1 : β se placera dans le bloc A, la marge libre dont dispose β est $\Delta'_A + \Delta_{j,j+n}$.

Pour la position 2 : β se placera dans le bloc B, la marge libre dont dispose β est $\Delta_{j,j+n}$

Pour la position 3 : β se placera dans le bloc $\sigma_{v(j)-1}$, la marge libre dont dispose β est $\Delta_A + \Delta_{j,j+n} + \Delta_{v(j)-1,A}$. Si on prend d'autres positions pour d_β on trouvera que la marge libre dont dispose j va de plus en plus augmenter.

Notons toute fois que si pour la position 2 la tâche j n'est pas affectée par l'insertion de la tâche β c'est-à-dire que si $p_\beta < \Delta_{j,j+n}$, alors pour toutes les autres positions de d_β on aura toujours p_β inférieur à la marge libre. (Puisque la marge libre augmente à chaque fois).

Ainsi, en diminuant d_β on obtiendra pour la tâche β des intervalles qui n'affecteront pas la tâche j.

Propriété 3 :

j_1 et j_2 sont deux tâches de la même pyramide telles que $d_{j1} < d_{j2}$. On suppose qu'en insérant une nouvelle tâche β dans la structure d'intervalle, la tâche j_1 ne subisse aucune déviation :

- Si $d_\beta < d_{j2} < d_{j1}$, alors la tâche j_2 ne subira elle aussi aucune déviation, si et si seulement si $\Delta_{j_2,j_{2+n}} > \Delta_{j_1,j_{1+n}}$. D'autre part si j_2 et j_1 appartiennent au même job alors, j_2 ne subira aucune déviation.

- Si $d_{j2} < d_\beta < d_{j1}$, alors la tâche j_2 ne subira elle aussi aucune déviation, si et si seulement si $\Delta'_{A_2} + \Delta_{j_2,j_{2+n}} \geq \Delta_{j_1,j_{1+n}}$.

- Si $d_{j2} < d_{j1} < d_\beta$, alors la tâche j_2 ne subira elle aussi aucune déviation, si et si seulement si $\Delta'_{A_2} + \Delta_{j_2,j_{2+n}} > \Delta_{j_1,j_{1+n}}$.

- Pour toutes les autres position de d_β, la tâche j_2 ne subira aucune déviation à condition que $\Delta_{A_2} + \Delta_{j_2,j_{2+n}} > \Delta_{A_1} + \Delta_{j_1,j_{1+n}}$.

Démonstration :

Pour la cas $d_\beta < d_{j2} < d_{j1}$ nous savons que dans la séquence au pire de j_1 on va trouver les deux tâches j_2 et β dans le bloc B_1 dans l'ordre $\beta < j_2 < j_1$, du coup pour ne pas affecter j_1 il faut que $p_\beta \leq \Delta_{j_1,j_{1+n}}$.

Dans la séquence au pire de j_2 on place les deux tâches j_1 et β, respectivement dans les blocs A_2 et B_2. Tel que, dans le bloc B_2, la tâche β se place juste avant la tâche j_2.

Nous remarquons alors que les blocs B_1 et $B_2 + j_2$ sont égaux. Il est clair alors, que si j_1 n'est pas affectée par l'insertion de β, j_2 ne le sera pas non plus si et seulement si $\Delta_{j_2,j_{2+n}} \geq \Delta_{j_1,j_{1+n}}$. Car de cette manière on assure que $p_\beta \leq \Delta_{j_2,j_{2+n}}$.

Cependant, cette condition est toujours vérifiée dans le cas ou les deux tâches sont liées par la contrainte de précédence $j_2 < j_1$.

Pour toutes les autres positions de d_β, nous remarquons que les marges libres disponibles sont égales pour les deux tâches j_2, j_1. En effet à part les deux blocs A et B le reste de la séquence est la même pour les deux séquences au pire des deux tâches, du coup, pour que la tâche j_2 ne subisse aucune déviation, il suffit que :

$$\Delta_{A_2} + \Delta_{j_2,j_{2+n}} > \Delta_{A_1} + \Delta_{j_1,j_{1+n}}.$$

2.6 Conclusion

Nous avons essentiellement étudié dans ce chapitre les notions relatives au *théorème des Pyramides*, à savoir les ensembles de séquences dominantes, les structures d'inter-

valles,et les séquences au mieux et au pire.

Une attention particulière est apportée aux notions de séquence dominante au mieux et au pire, qui définissent pour chaque tâches l'ordonnancement lui permettant de commencer le plus tôt possible, et celui qui la retarde le plus. Effectivement, à l'issue de l'application du théorème des pyramides, à chaque opération i sont affectés un intervalle de début noté $[S_i^{min}\ S_i^{max}\ ]$, et un intervalle de fin $[f_i^{min}\ f_i^{max}]$.

A la fin de ce chapitre, nous avons introduits des propositions très intéressantes concernant la détection des marges libres existantes dans la séquence au pire d'une tâche. Et nous avons vu qu'il existait une manière de choisir la fênetre d'exécution d'une tâche de manière à remettre en cause le moins possible, les plans de production déja établis.

Dans le chapitre suivant, nous allons voir comment exploiter la flexibilité séquentielle apportée par le théorème des pyramides pour faire coopérer les ressource entre elles, et aboutir à des structures d'intervalles parfaitement cohérentes. Nous allons voir aussi, comment faire négocier les ressources afin d'insérer dynamiquement les jobs arrivant en temps réel.

Chapitre 3

Proposition d'une méthode d'ordonnancement distribué par coopération inter-machines

---

Dans le présent chapitre, nous vous proposons une méthode d'ordonnancement distribué par coopération inter-machines, qui se base sur le théorème des pyramides défini dans le deuxième chapitre. Pour ce, nous commenceront par présenter l'idée générale de l'approche élaborée, puis nous présenterons, le protocole de négociation, et les algorithmes qui en découlent. Pour finir, il sera question de traiter la Prise en compte des commandes aléatoires en temps réel.

3.1 Principes généraux de l'approche developée

Dans notre approche, nous supposons que chaque ressource est assimilée à un centre de décision, qu'elle gère son propre ordonnancement local, et qu'elle dispose donc de sa propre flexibilité décisionnelle. Il s'agit d'une flexibilité séquentielle exprimée par le nombre de séquences dominantes que *le théorème des pyramides* permet de caractériser sur chaque machine.

Dans [OBB07], les auteurs expliquent comment se fait l'interaction entre les différentes ressources. Comme l'illustre la figure 3.1, les centres de décision sont traversés par un flux d'en-cours. Ils s'échangent les données nécessaires à leurs prises de décision. Sur cette figure, le centre de décision *Ressource* reçoit d'un certain nombre de centres de décision amont *Fournisseurs* les produits qu'il transforme. Une fois le produit transformé, le centre de décision Ressource le transfère à son tour vers les centres de décisions aval *Clients* chargés des prochaines transformations de la gamme. On note que les liens physiques (flux d'en-cours) entre les ressources dépendent des gammes opératoires associées aux produits manufacturés ; ces gammes étant supposées initialement connues.

Les échanges d'information entre centres de décision ont pour but de définir les intervalles de livraisons d'un produit d'un centre vers un autre et ce, selon les contraintes de

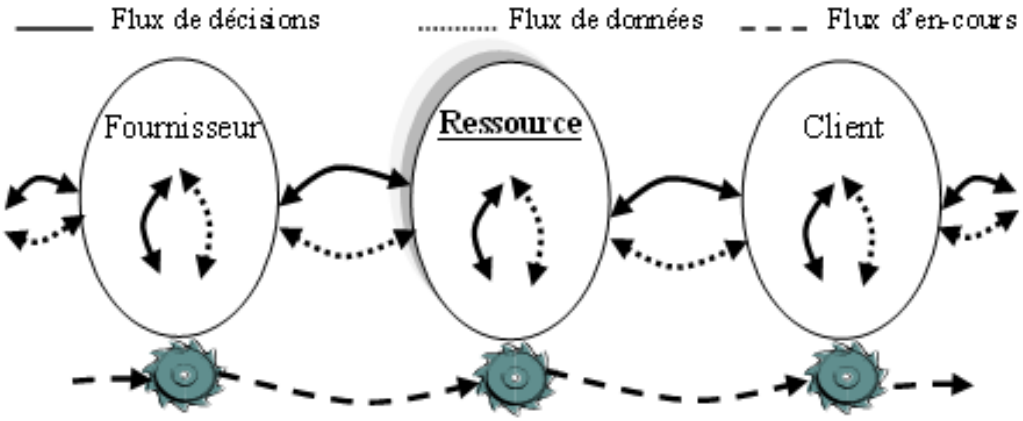


FIG. 3.1 – Relations entre les centres de décisions [OBB07]

précédence dictée par la gamme opératoire. En effet, une négociation entre une paire de centres de décision est amorcée lorsque l'une des ressources réalise une tâche j, précédée au sens de la gamme par une tâche j-1, réalisée elle même par la deuxième ressource. A ce niveau, la flexibilité séquentielle dont dispose chacune des ressources est d'une importance capitale puisqu'elle permet de négocier des intervalles et non des valeurs unique ce qui donne plus de chance aux tâches d'être cohérentes.

Durant la négociation il est possible qu'une machine change une ou plusieurs fois sa structure d'intervalle afin de répondre aux contraintes de précédences.

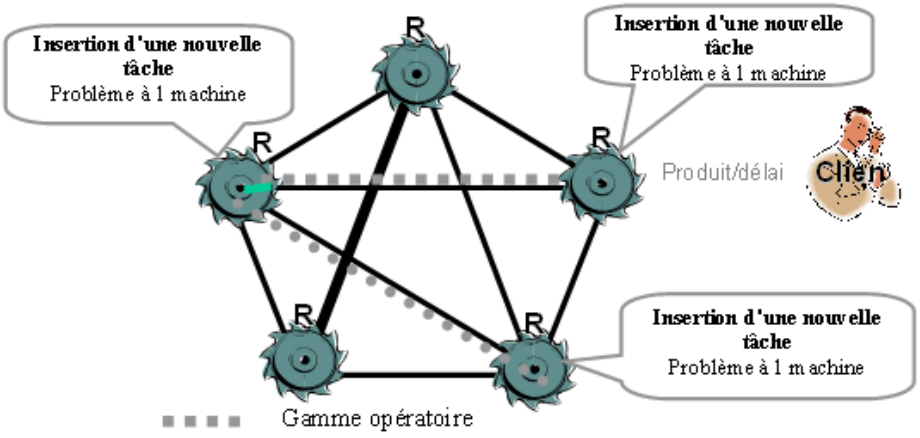


FIG. 3.2 – Prise en charge dune nouvelle commande [OBB07]

Dans la phase dynamique de l'ordonnancement, l'arrivée dans le système d'une nouvelle commande passée par un client, induit une série de conversations (propositions et contre-propositions) entre les ressources concernées par la réalisation du produit. La figure3.2 , illustre comment se propagent les conversations de ressource en ressource en

remontant la gamme opératoire. Lorsque les conversations sont achevées, toutes les tâches nécessaires à la réalisation du produit doivent être insérées dans les ordonnancements locaux concernés.

Concrètement, ceci se traduit d'abords par la géneration d'une structure d'intervalle initiale sur chaque ressource, puis par la remise en cause de ces structures d'intervalles de manière à établir un scénario de cohérence entre les tâches exécutée sur les différentes ressources mais se succédant selon la gamme opératoire. Ces structures d'intervalles *corrigées* permettent de génerer un ensemble dominant de solutions contenenant l'optimum local (le minimum des L_{max}), et permettant de calculer un intervalle pour la solution globale c'est à dire un intervalle pour le C_{max}.

3.2 Description de l'approche proposée

3.2.1 Objectifs et hypothèses de travail

Comme nous venons dans le chapitre précédant (section 2.2), notre approche se base sur la décomposition du problème $J_n||C_{max}$, en n sous problèmes $1|r_i|L_{max}$. Cette décomposition est très commune dans la littératures et plusieurs travaux dont [Tru05] s'en sont inspirés. Un ordonnancement local est géré sur chaque ressource disjonctive (ordonnancement à une machine).

Les dates de disponibilité et les dates échues des opérations sont calculées grâce à l'algorithme de Ford Bellman (cf. section 2.1), qui se base sur le graphe conjonctif.

Pour chaque problème à une machine, l'application du théorème des pyramides permet de déterminer un ensemble de séquences dominantes vis-à-vis du retard algébrique. La dominance est alors, relative au problème à une machine et non au problème job shop (puisque les problèmes à une machine sont interdépendants). Pour chaque tâche i, nous pouvons déduire les dates de début au mieux et au pire, s_i^{min} et s_i^{max}, grâce aux équations 2.8 et 2.9.

Notre objectif est de parvenir à une solution d'ordonnancement globale en respectant l'autonomie décisionnelle de chaque ressource. Pour ce, celles-ci sont assimilées à des centres de décision, entre lesquels un protocole de coopération permet d'aboutir à une solution globale respectant certaines contraintes de cohérence dictées par les gammes opératoires. Le but principal est alors, d'établir un protocole de coopération qui permet à la fois de préserver l'autonomie décisionnelle de chaque ressource afin qu'elle puisse réagir au perturbations, sans imposer une perpétuelle remise en cause des décisions déjà prises, tout en assurant une performance globale satisfaisante.

Les algorithmes que nous proposons visent à trouver sur chaque machine une structure d'intervalles caractérisant un ensemble de séquences dominantes respectant les conditions de cohérence et de fléxibilité que l'on définira dans la section suivantes.

L'évaluation des performances au mieux et au pire de l'ensemble de solutions du problème peut être réalisé de la façon suivante :

Notons ED_i l'ensemble de séquences dominantes obtenu sur la machine M_i et $C_{ED_i}^{min}$ et $C_{ED_i}^{max}$ respectivement la durée au mieux et au pire de l'ordonnancement sur chaque machine.

On a :

$$C_{ED_i}^{min} = \max(s_i^{min} + p_i), \forall i \in ED_i \tag{3.1}$$

$$C_{ED_i}^{max} = \max(s_i^{max} + p_i), \forall i \in ED_i \tag{3.2}$$

Notons alors $\underline{C}_{max}$ et $\overline{C}_{max}$ respectivement les makespan au mieux et au pire du job shop. On a alors :

$$\underline{C}_{max} = \max(C_{ED_i}^{min}), \forall ED_i \tag{3.3}$$

$$\overline{C}_{max} = \max(C_{ED_i}^{max}), \forall ED_i \tag{3.4}$$

Le makespan C_{max} du problème est donc tel que :

$$\underline{C}_{max} \leq C_{max} \leq \overline{C}_{max}$$

Nous considérons acceptables les solutions C_{max} remplissant la condition suivante :

$$C_{max} \leq \frac{5C_{max}^*}{4} \tag{3.5}$$

Tel que C_{max}^* est le C_{max} optimal calculé par d'autres méthodes approuvées.

On impose aussi des valeurs limites pour le taux de flexibilité et le taux de cohérence [1], à savoir :

$$\tau_{flex}^{max} = 0.5 \tag{3.6}$$

et,

$$\tau_{coh}^{min} = 0.25 \tag{3.7}$$

Comme hypothèses principales de travail, nous supposons qu'une décision négociée n'est jamais remise en cause, et que chaque centre de décision possède une base de connaissance contenant les informations suivantes :

– Les dates de disponibilité et les dates échues de toutes les tâches qui y sont exécutées.

[1] Ces deux paramètres vont être définis dans les deux sections suivantes

- Pour chaque tâche i, les ressources R(i-1) et R(i+1), exécutant respectivement la tâche qui la précède et la tâche qui la succède.
- Les valeurs des taux de cohérence et de flexibilité.
- le délai de livraison du job, négocié avec le client.
- L'historique des négociations.

3.2.2 Contraintes liées au modèle d'ordonnancement coopératif

(a)Contraintes de cohérence

L'approche d'ordonnancement coopératif que nous proposons vise à trouver, pour chaque machine, une structure d'intervalles caractérisant un ensemble de séquences dominantes respectant les conditions suivantes :

$$s_{i+1}^{min} \geq s_i^{min} + p_i, i = 1, ..., n \tag{3.8}$$

$$s_{i+1}^{max} \geq s_i^{max} + p_i, i = 1, ..., n \tag{3.9}$$

Où (i,i + 1) est un couple de deux tâches tel que i doit précéder i + 1, selon la gamme opératoire.

Tout le long de ce manuscrit, nous allons appelé la contrainte 3.8, contrainte au mieux, et la contrainte 3.9, contrainte au pire.

On représente sur la figure 3.3 deux tâches i et i+1 pour lesquelles les deux conditions de cohérence 3.8 et 3.9 sont respectées.

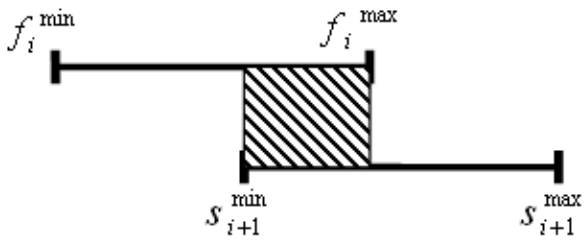


FIG. 3.3 – Intervalles respectant les contraintes de cohérence

Sur la figure 3.3, nous remarquons que si les intervalles $[f_{min}^i \ f_{max}^i]$ et $[s_{i+1}^{min} \ s_{i+1}^{max}]$ étaient disjoints (c'est-à-dire que la zone hachurée n'existe pas) alors, la date fin au pire de l'opération i est compatible avec la date de début au mieux de l'opération i+1 et le risque d'incohérence est nul, mais, la réalisation de i+1 ne pouvant se faire immédiatement après la fin de i, il sera nécessaire de stocker le produit ayant subi la tâche i

avant de pouvoir procéder à la tâche i+1 (ce qui induit une perte de performance globale).

Les intervalles peuvent éventuellement aussi se recouvrir (figure 3.3). Dans ce cas, le temps d'attente entre les ressources sera réduit et on aura une performance globale meilleure, en contre partie autoriser le chevauchement pourrait faire en sorte que la date effective de fin de i ne soit pas compatible avec la date de début effective de i+1.

Afin de favoriser la performance globale tout en assurant un risque d'incohérence faible, Ourari dans [OBB07] propose d'introduire la contrainte suivante, où τ_{coh}^{min} désigne un seuils d'incohérence minimal que l'on impose.

$$\tau_{coh}^{min} \leq \tau_{coh} = \frac{f_{max}^{i} - s_{min}^{i+1}}{s_{max}^{i+1} - f_{min}^{i}} \tag{3.10}$$

Il est clair que pour assurer le respect des conditions 3.8 et 3.9, il faut agir sur les dates de début ou de fin au pire et au mieux de chaque couple de tâches incohérentes. Pour ce, il suffit d'agir sur les structures d'intervalles des ressources qui réalisent i et i+1, car ce sont elles qui sont à l'origine de la définition des dates de début au mieux et au pire .

Si on notait $\widetilde{r_i}$ et $\widetilde{d_i}$ les valeurs fixes des releases dates et des due dates connus à la base du graphe conjonctif du problème, la nouvelles structure d'intervalles qu'une ressource définira ou négociera avec la ressource en amont ou en aval, doivent donc être cohérentes avec les valeurs initiales et vérifier les contraintes suivantes :

$$r_i \geq \widetilde{r_i} \tag{3.11}$$

$$d_i \geq r_i + p_i \tag{3.12}$$

(a)Contraintes de fléxibilité

Au niveau de chaque ressource k, est définie une structure d'intervalle Δ qui permet de caractériser un ensemble dominant de séquences par application du théorème de pyramide que l'on note $S^{\Delta}_{dom(k)}$. La cardinalité de cet ensemble de séquences dominantes définit la flexibilité décisionnelle de la ressource. Dans le but d'assurer globalement un même niveau de flexibilité pour chaque ressource, il est intéressant de contraindre la perte de flexibilité, lorsqu'on passe d'une structure d'intervalle Δ à une autre structure $\widetilde{\Delta}$, de sorte que cette perte soit toujours inférieure ou égale à une valeur supérieure τ_{flex}^{max}.

$$\tau_{flex} = \frac{\left\| S^{\Delta}_{dom(k)} \right\| - \left\| S^{\widetilde{\Delta}}_{dom(k)} \right\|}{\left\| S^{\Delta}_{dom(k)} \right\|} \tag{3.13}$$

3.2.3 Mécanisme d'aide à la décision pour assurer la cohérence entre les tâches

Afin d'assurer le respect des contraintes 3.8 et 3.9, chaque centre de décision devrait agir compte tenu de son autonomie. Cela signifie que la remise en cause des structures d'intervalles calculées initialement, devrait se faire de manière totalement locale. Dans cette partie nous proposons un mécanisme d'aide à la décision décrivant le comportement de chaque centre de décision face à une incohérence.

Sur la figure 3.4, la zone hachurée (1), représente le non respect de la condition de cohérence au mieux, tandis que la zone hachurée (2), représente le non respect de la condition de cohérence au pire. Afin d'assurer la cohérence des tâches, i et j représentées sur cette figure, nous proposons d'agir en deux temps :

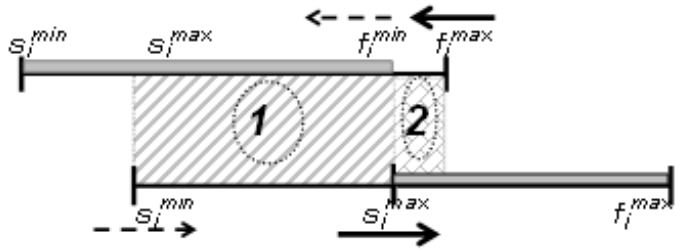


FIG. 3.4 – Réduction et augmentation des dates de début et de fin

- On impose d'abords, le respect de la contrainte de cohérence au mieux, en agissant sur les séquences au mieux des tâches. Car ceci peut faire en sorte que le couple de tâches devienne cohérent vis-à-vis de la contrainte au pire sans avoir à effectuer d'autres changements,

- ensuite si la condition de cohérence au pire n'est toujours pas vérifiée, on agit sur les séquences au pire des tâches.

Dans le paragraphe suivant, nous allons établir quelques règles qui permettent de diminuer ou d'augmenter une date de début ou de fin, et ce en se referant aux définitions de la séquence la plus favorable et de la séquence la plus défavorable d'une tâche.

1. Respect de la contrainte au mieux

Afin d'assurer le respect de la contrainte de cohérence au mieux pour un couple de tâche (i,i+1), on peut soit diminuer la valeur de f_i^{min}, soit augmenter la valeur de s_{i+1}^{min}. En se basant sur la définition de la séquence au mieux d'une tâche (Cf. section 2.3.3).

Pour réduire la date de fin au mieux, il est fortuit de pense à diminuer le nombre de tâches devant nécessairement être séquencées avant i. Pour ceci, on peut :

Règle 1 : Réduire le deadline d_i de la tâche i, tout en réspectant la condition 3.12.

Règle 2 : Augmenter les valeurs des releases dates des tâches k telles que u(k)<u(i) [2], jusqu'à ce que u(k) devienne supérieur à u(i). Il se peut que cette augmentation rende les taches k incohérentes vis-à-vis de la contrainte 3.12, dans ce cas, il faut augmenter aussi leurs deadlines.

Pour augmenter la date de début au mieux de la tâche (i+1), on augmente le nombre de tâches devant nécessairement être séquencées avant i+1. Pour cela on peut :

Règle 1 : Augmenter la valeur de r_{i+1}, date de disponibilité de la tâche i+1. Si jamais la contrainte 3.12 n'est plus vérifiée augmenter d_{i+1} aussi.

Règle 2 : Diminuer les deadlines des tâches k telles que u(k)>u(i+1), jusqu'à ce que u(k) devienne inférieur à u(i+1), cette réduction est limitée par la contrainte 3.12.

2.Respect de la contrainte au pire

De la même manière, pour assurer le respect de la contrainte de cohérence au pire pour un couple de tâche (i,i+1), on peut soit diminuer la valeur de f_i^{max}, soit augmenter la valeur de s_{i+1}^{max}. On se base sur la définition de la séquence au pire d'une tâche , pour proposer quelques règles permettant d'augmenter ou de diminuer une date de début au pire.

En effet, pour diminuer la valeur de la date de début au pire s_i^{max} d'une tâche i, on peut :

Règle 1 : Réduire la taille de la zone B (Cf. figure 2.8), pour ce on peut soit augmenter les deadlines des tâches k du bloc B jusqu'à ce qu'ils deviennent supérieurs à d_i, soit diminuer d_i jusqu'à ce que $d_i < d_k$.

Règle 2 : Réduire v(i) ; l'indice de la dernière pyramide à laquelle appartient la tâche i, en diminuant la valeur de d_i. Cette réduction est limitée par la contrainte 3.12.

Règle 3 : Augmenter les indices u(k) des premières pyramides auxquelles appartiennent les tâches k de la séquence au pire de la tâche i, jusqu'à avoir u(k)>v(i). Pour ce on peut soit Augmenter les release dates des tâches k, soit augmenter leurs deadlines, soit augmenter les deux.

Pour augmenter la valeur de la date de début au pire s_{i+1}^{max} d'une tâche (i+1), on peut :

[2] u(i) est l'indice de la première pyramide à laquelle apartient la tâche i (Cf.section 2.3.2)

Règle 1 : Augmenter la taille de la zone B (Cf. figure 2.8). Pour ce on peut soit augmenter le deadline de la tâche (i+1), soit diminuer les deadlines des tâches qui appartiennent au bloc A (Cf. figure 2.8). Ceci doit se faire dans le respect de la contrainte 3.12.

Règle 2 : Augmenter la date de disponibilité de la tâche (i+1) de manière à ce que (i+1) appartienne à des pyramides d'indice u(i+1) plus grand. Pour respecter la contrainte 3.12, on peut aussi augmenter son deadline.

3.3 Protocole de coopération inter Ressources

3.3.1 Les primitives de Négociation retenues

Dans le paragraphe suivant, nous vous proposons les différentes primitives de négociation retenues dans l'élaboration de notre protocole de négociation. Pour rappel, le centre de décision initiateur est le centre de décision qui entame la conversation avec un autre centre de décision qui sera appelé particiant.

(a) Primitives de l'initiateur

L'initiateur possède trois primitives de négociation :

- ***Propose (contrat)*** : c'est la première primitive que l'initiateur envoie aux participants pour leur proposer des contrats.

- ***confirme (contrat)*** : ce message indique aux participants que le contrat en cours est confirmé et que la négociation a été un succès.

(b)Primitives du participant

Les messages envoyés par le participant sont uniquement destinés à l'initiateur.

- ***Accepte (contrat) :*** ce message répond à la proposition de contrat faite par l'initiateur. Le participant indique par ce message à l'initiateur qu'il accepte le contrat tel qu'il est. Il vient en réponse aux messages Propose émis par l'initiateur.

- ***Contre propose (contrat)*** : C'est la réponse du participant dans le cas où il propose une contre-proposition.

(c) Primitives communes à l'initiateur et au participant

- ***Test faisabilité (contrat)*** : ce message est envoyé par le participant ou l'initiateur aux autres centres de décision afin de les consulter ou autrement dit de *" prendre leurs avis "* concernant la prise d'une décision. La réponse à cette primitive se fait soit avec le message *faisable* qui indique à l'expéditeur du message que la décision qu'il est sur le point de prendre ne lèse en rien un autre centre décision, soit avec

Non faisable dans le cas contraire.

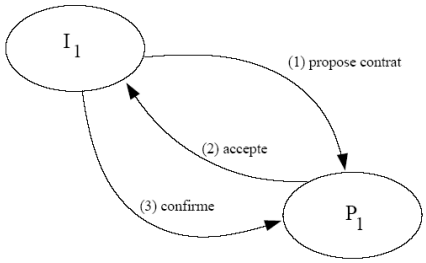


FIG. 3.5 – Exemple simple de négociation : I_1 propose un contrat à P_1 qui l'accepte

3.3.2 Philosophie générale du protocole de négociation

L'approche developpée dans ce manuscrit étant une approche *proactive-réactive*, le protocole de négociation que nous avons adopté sera lui aussi scindé en une phase statique et une phase dynamique. Dans la phase statique, on suppose que les jobs à traiter sont connus et qu'aucune perturbation n'est prise en compte. Il sera alors question de négocier une fois pour élire le job à prendre en charge, puis négocier pour assurer la cohérence des tâches du job choisi. Dans la phase dynamique, il sera question de définir des fenêtres d'exécutions pour les tâches des nouveaux jobs. Ceci se fait dans l'ordre d'apparition des nouvelles commandes (job par job). Pour finir, la liste des jobs est mise à jour et le rocessus de négociation est relancé.

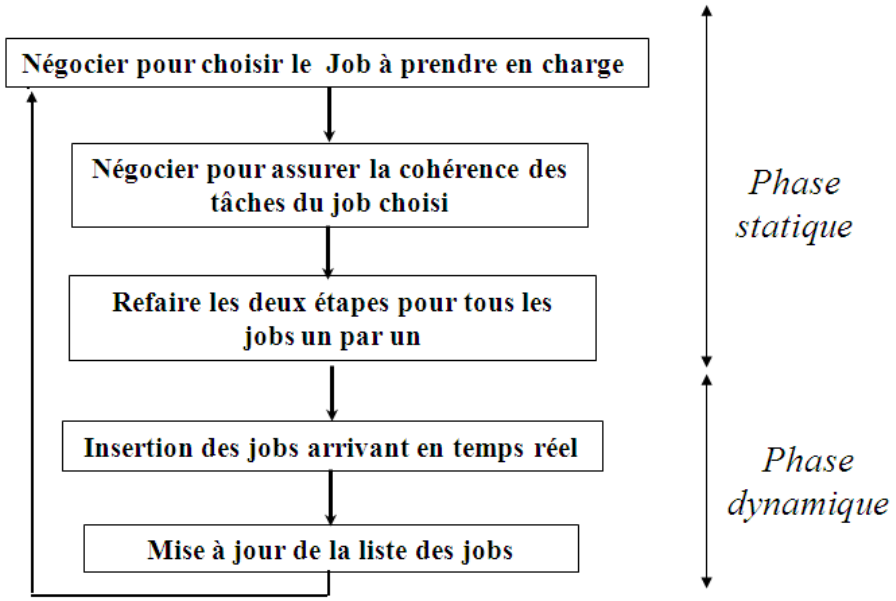


FIG. 3.6 – Etapes éminentes du protocole de négociation

3.3.3 Etapes éminentes du protocole de négociation adopté

Première étape : *Choix du Job à prendre en charge*

Les ressources négocient une première fois pour élire le job à prendre en charge, leur choix devrait porter sur le job ayant la plus petite marge entre la fin de sa dernière tâche et le délai négocié avec le client.

Ce choix est justifié par le fait que le premier job à être pris en charge bénéficie de plus de chance d'exécuter les plans locaux et donc, de réduire les valeurs de fin localement, plutôt que d'émettre des contre-propositions qui sont synonymes de l'augmentation des valeurs de début et de dégradation des performances globales.

Deuxième étape : *Négocier pour assurer la cohérence entre les tâches du job choisi*

Comme nous l'avons déja expliqué, le théorème des pyramides permet de générer sur chacune des ressources une structure d'intervalle, c'est à dire une fenêtre d'exécution $[r_i$ $d_i]$ pour chaque tâche i (Cf.section 2.3.2).

Dès lors que ces données sont disponibles au sein de chaque centre décisionnel, un processus d'envoi de propositions d'intervalles est amorcé. En effet, les propositions sont formulées en remontant la gamme opératoire, un centre de décision exécutant la tâche (i+1) va se voir émettre une proposition au centre de décision exécutant la tâche i. Cette proposition va concerner l'intervalle de début au mieux et au pire de la tâche i+1 $[s_{i+1}^{min}$ $s_{i+1}^{max}]$.

Lorsqu'un centre de décision reçoit une proposition d'intervalle, il effectue aussitôt un test de cohérence lui permettant de détecter si les deux tâches (i,i+1) exécutées par les deux centres de décisions initiateur et participant sont cohérentes ou pas. ce test, qu'on va nommer *Test de Cohérence*, consiste à vérifier si les deux contraintes de cohérence 3.8 et 3.9, sont vérifiées, et si le taux de cohérence minimal τ_{coh}^{min} est respesté, ceci se traduit par le vérification de la contrainte 3.10.

Une fois que le centre de décision participant a déctecté l'état d'incohérence du couple de tâche (i,i+1), un processus décisionnel est amorçé dans le but de décider de l'action à suivre selon l'état d'incohérence détecté. Trois actions sont alors possibles :

- Si le couple de tâche est *cohérent* alors le centre de décision participant envoit le message *Accept* vers le centre de décision initiateur ;
- Si le couple de tâche est *incohérent* alors le centre de décision participant suit un plan local our modifier la structure d'intervalle ;
- Si le couple de tâche est *cohérent* et aucune action locale n'est possible alors le centre de décision participant envoit le message *Contre proposition* vers le centre de décision initiateur ;

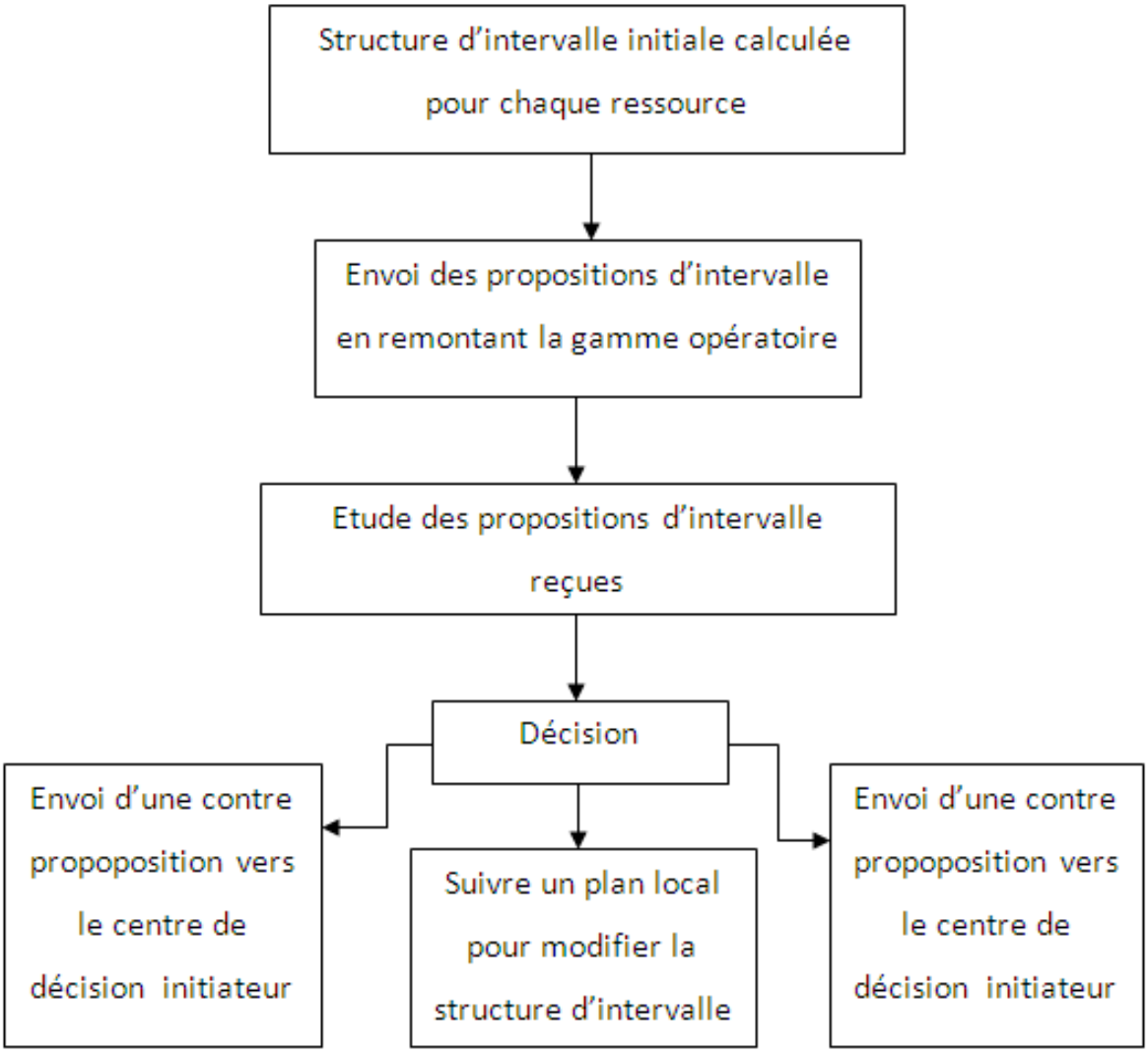


FIG. 3.7 – organisation générale du protocole de négociation

La figure 3.8 résume la philosophie générale de ce protocole de négociation.

De manière plus concrète voici ce qui se passe au sein d'un centre de décision :

Pour commencer, le centre de décision ne peut répondre à une proposition d'intervalle que si le centre de décision se situant en amont a répondu à la proposition de celui-ci.

Le test de cohérence vise à vérifier les conditions 3.8 et 3.9. Le test de cohérence est une étape importante, car c'est lui qui conditionne le choix de l'action à suivre.

Tel qu'il a été mentionné dans la partie 3.2.3, nous nous commencerons d'abords par imposer le respect de la contrainte de cohérence au mieux puis de la contrainte de cohérence au pire.

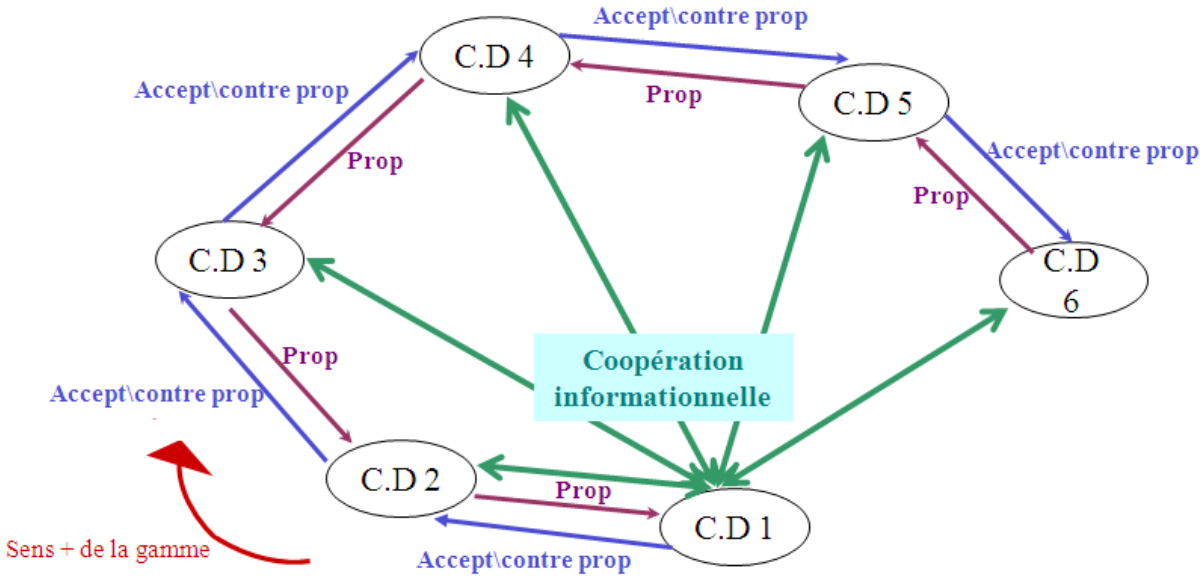


FIG. 3.8 – Philosophie générale du protocole de négociation adopté

Pour ce, les plans locaux A et B visent respectivement à diminuer la dates de fin au mieux et la date de fin au pire de la tâche i. Ces deux plans locaux sont représentés sur les figures 3.9, 3.10.

Rappelons que l'envoi du message ***test faisabilité***, permet au centre de décision de vérifier à chaque fois si la décision qu'il entreprend crée une autre incohérence ailleurs ou bien si elle remet en cause une décision déja négociée. En recevant ce message, les autres centres de décisions (les centres de décisions accointance) sont sensé y répondre en priorité, effectuer un test de cohérence et renvoyer le résultat au centre de décision qui les a sollicité. Si le test de cohérence révèle l'existance d'une incohérence alors leur réponse serait ***infaisable*** sinon elle serait ***faisable***. Et ce n'est que dans ce dernier cas que la décision sera validée.

Lorsqu'une contre proposition est émise, c'est à dire une demande d'augmentation des valeurs de début au mieux et au pire, les plans d'action locaux C et D sont sollicités.

Ceux-ci sont construits tel qu'il est représenté sur les figures 3.11, et 3.12.

Pour le plan d'action A et C, le paramètre Δ est calculé de la manière suivante :

$$\Delta = f_i^{min} - s_i^{min} \tag{3.14}$$

Pour le plan d'action B et D, le paramètre Δ est calculé de la manière suivante :

$$\Delta = f_i^{max} - s_i^{max} \tag{3.15}$$

Troisième étape : *Prise en compte des commandes aléatoires en temps réel*

Etant donné une date de livraison du nouveau job, et les durées opératoires des tâches qui le composent, noua avons défini dans le précédant chapitre (Cf. 2.5) pour chaque tâche du noveau job une date de début et de fin qui permet de respecter le délai de livraison imposé tout en essayant d'altérer le moins possibles les dates de livraisons des autres jobs. Dans le cas ou ceci est impossible, nous avons donné la déviation subite par les dates de fin des tâches affectées suite à l'insertion du nouveau job.

Une fois que l'on aurai définit pour chacune des tâches du nouveau job une fenêtre d'éxecution, il sera question de vérifier si l'ensemble reste cohérent vis à vis des deux contraintes 3.8 et 3.9. Ceci est réalisé en amorçant une seconde fois le processus coopératif après mise à jour de la liste des jobs.

Notons toute fois que les nouveaux jobs seront insérés un par un, selon leur ordre d'apparition, ou bien selon un ordre de priorité négocié avec le client.

3.4 Exemple

Pour illustrer ce protocole de négociation, considérons à nouveau l'exemple donné dans la section 2.3.4. Chaque ressource gère une structure d'intervalles et dispose localement et de ce fait d'un ensemble d'informations sur les dates de début et de fin au pire et au mieux. Ces informations sont communiquées aux ressources qui lui sont liées par une relation de gamme opératoire.

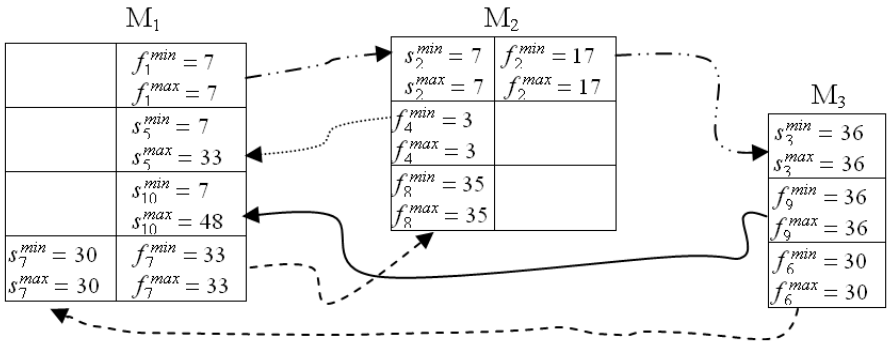


FIG. 3.13 – Schématisation des cohérences entre ressources

Il existe une incohérence entre les tâches 9 et 10. Des propositions de négociation sont amorcées par la ressource 1 qui demande à la ressource 3 de diminuer la date de fin de la tâche 9. Comme il n'est pas possible de diminuer la date de début de 9, c'est la ressource 1 qui doit augmenter la date de début du travail 10 en éliminant les séquences dominantes faisant commencer 10 trop tôt. On obtient ainsi les dates au pire et au mieux indiquées sur les figures 3.14, 3.15 .

A ce stade, toutes les dates étant cohérentes entre elles, la conversation entre les ressources s'achève, et la solution finale est un ordonnancement dans lequel les ordres

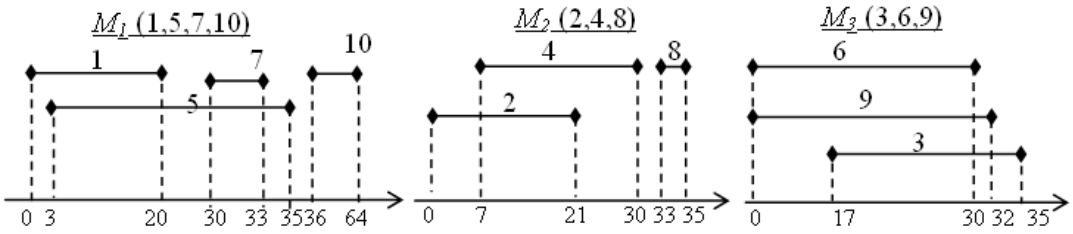


FIG. 3.14 – Nouvelles structures d'intervalles après négociation

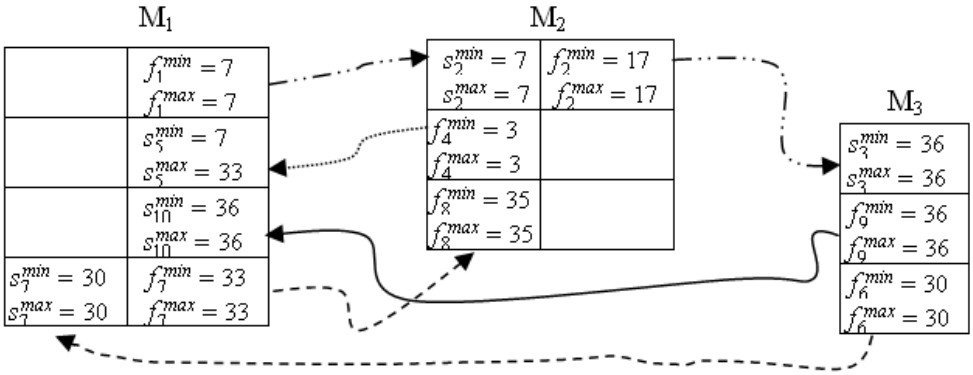


FIG. 3.15 – Schématisation des cohérences entre ressources après négociation

de passage sur les ressources 1, 2 et 3 sont respectivement (1,5,7,10), (4,2,8) et (6,9,3), ou bien (1,7,5,10), (4,2,8) et (6,9,3). En ne considérant que les ordonnancement actifs, la ressource 1 peut uniquement retenir la séquence (1,5,7,10) qui conduit à une solution optimale ayant un Makespan de 41. Remarquons que la solution optimale donnée dans [ABZ88] correspond à l'ordonnancement (1,5,10,7), (4,2,8) et (9,6,3).

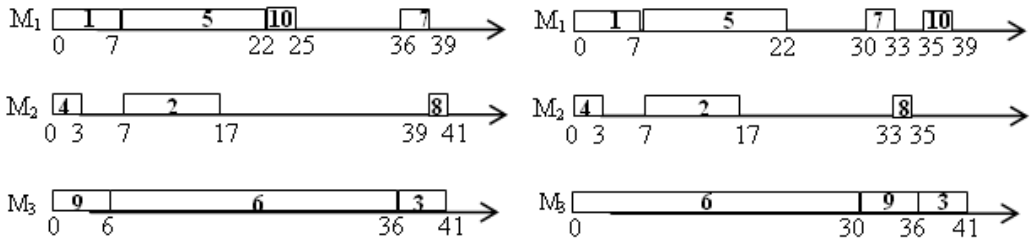


FIG. 3.16 – solutions optimales

3.5 Conclusion

Dans le présent chapitre, nous avons établit un protocole de négociation permettant de résoudre le problème posé d'une manière complètement distribuée, et sans faire intervenir un agent humain. Nous avons aussi décrit une méthode de réordonnancement permettant d'amortir l'impact que pouvait avoir l'insertion de tâches arrivant en temps réel .

1) Envoi d'une proposition d'intervalle, le contrat (l'intervalle $[s_i^{min}\ s_i^{max}]$) est envoyé vers le centre de décision R(i-1) s'il existe.

2) Réception de la proposition d'intervalle émise par le centre de décision R(i+1).

2.a) **Si** la tâche i est la première de son job **alors** passer à l'étape 3).

2.b) **Sinon** attendre la réponse du centre de décision R(i-1).

2.b.1)**Si** la réponse est ***Contre prop*** **alors** :

2.b.1.1)**Si** c'est la contrainte de cohérence au mieux (equation 3.8) qui n'est pas vérifiée **alors** appliquer ***le plan local*** C, puis passer à l'étape 3).

2.b.1.2)**Si** ce sont les deux contraintes de cohérence (equations 3.8, et 3.8) qui ne sont pas vérifiées **alors** appliquer ***le plan local*** C, puis passer à l'étape 3).

2.b.1.3)**Si** c'est la contrainte de cohérence au pire (equation 3.9)qui n'est pas vérifiée **alors** appliquer ***le plan local*** D, puis passer à l'étape 3).

2.b.2) **Si** la réponse est ***Accept*** alors passer à l'étape 3).

3) Effectuer le test de cohérence.

3.a) **Si**les intervalles sont cohérents alors envoyer message **Accept** vers le centre de décision initiateur.

3.b) **Si** les intervalles sont incohérents par rapport à la contrainte de cohérence au mieux et/ou au pire **alors** envoyer message ***test faisabilité*** vers les centres de décisions accointance [3].

3.b.1) **Si** toutes les réponses sont positives **alors** appliquer ***le plan local*** A.

3.b.2) **Si** au moins une réponse est négative **alors*envoyer message Contre prop*** vers le centre de décision initiateur.

3.c) **Si** les intervalles sont incohérents au pire **alors** ***envoyer message test faisabilité*** vers les centres de décisions accointance.

3.b.1) **Si** toutes les réponses sont positives **alors** appliquer ***le plan local*** B.

3.b.2) **Si** au moins une réponse est négative **alors** ***envoyer message Contre prop*** vers le centre de décision .

TAB. 3.1 – Etapes du processus coopératif

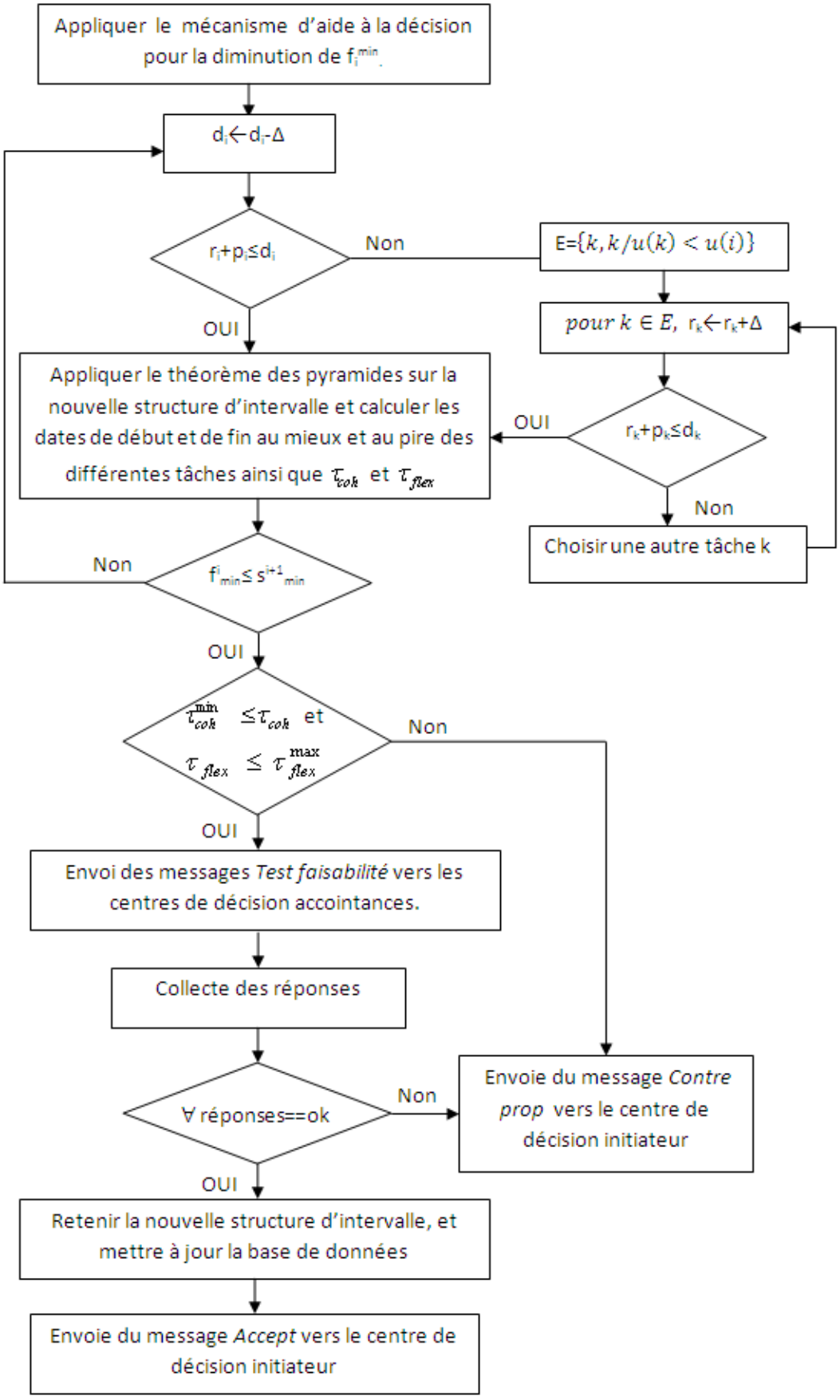


FIG. 3.9 – Plan local A

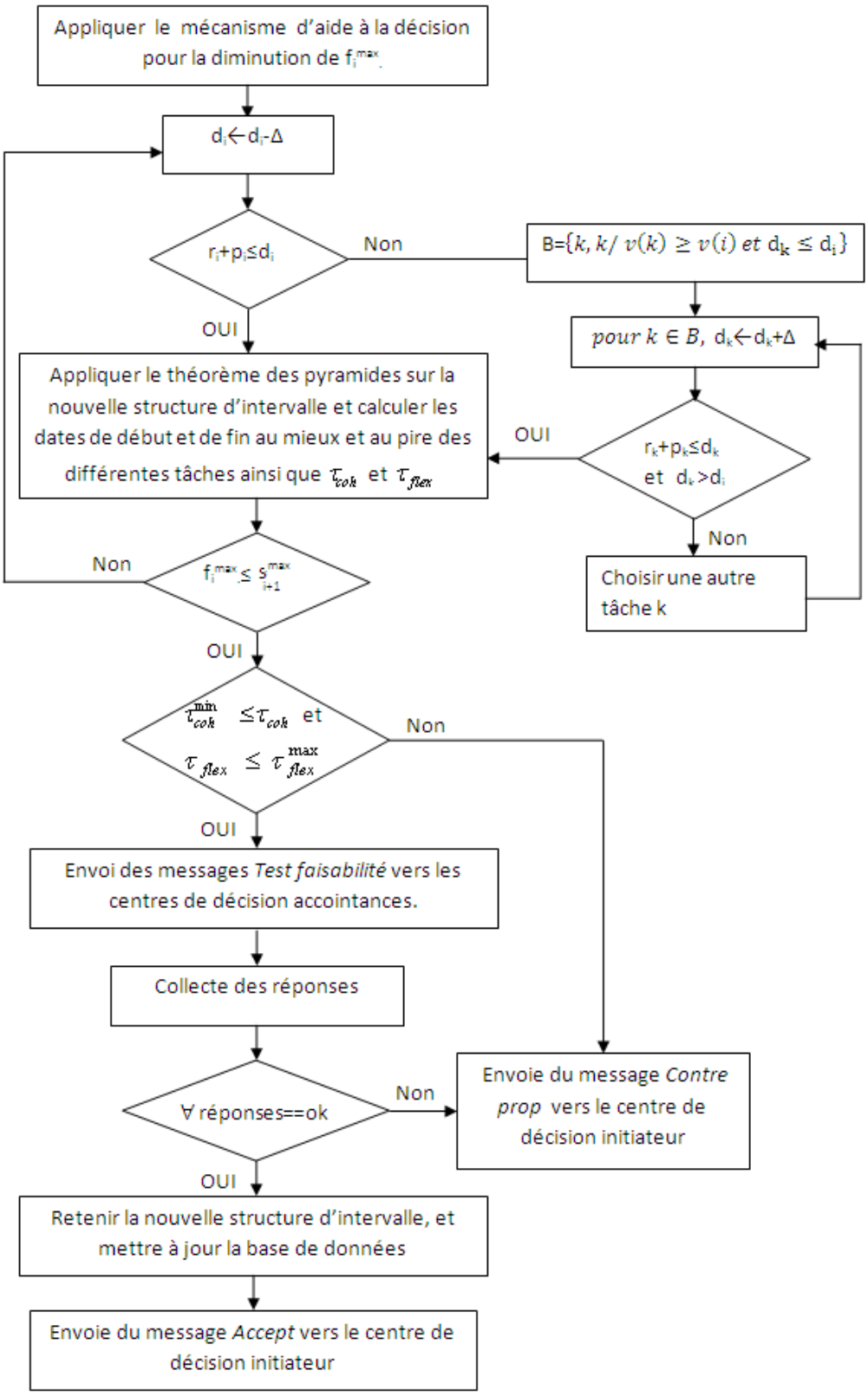


FIG. 3.10 – Plan local B

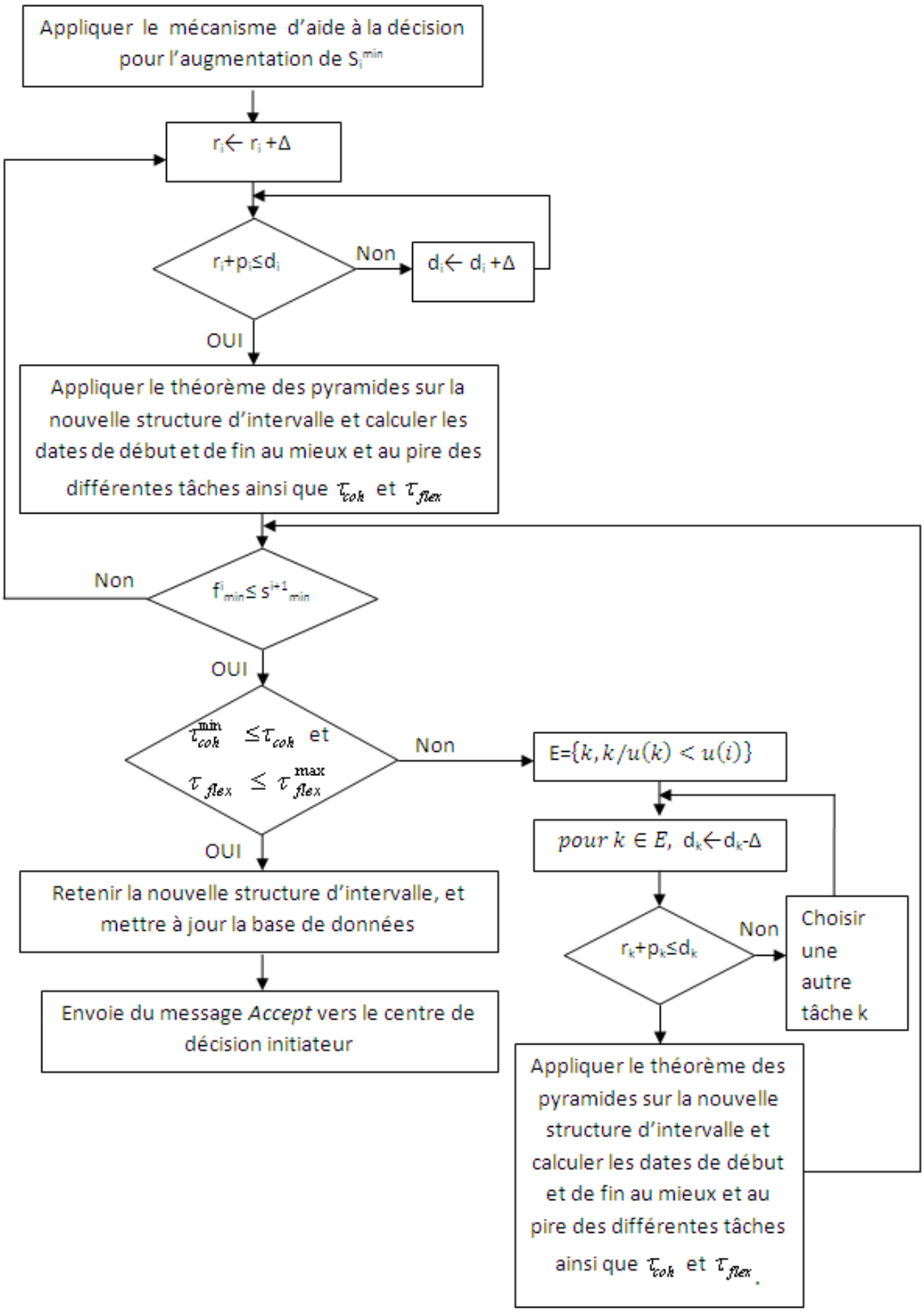


FIG. 3.11 – Plan local C

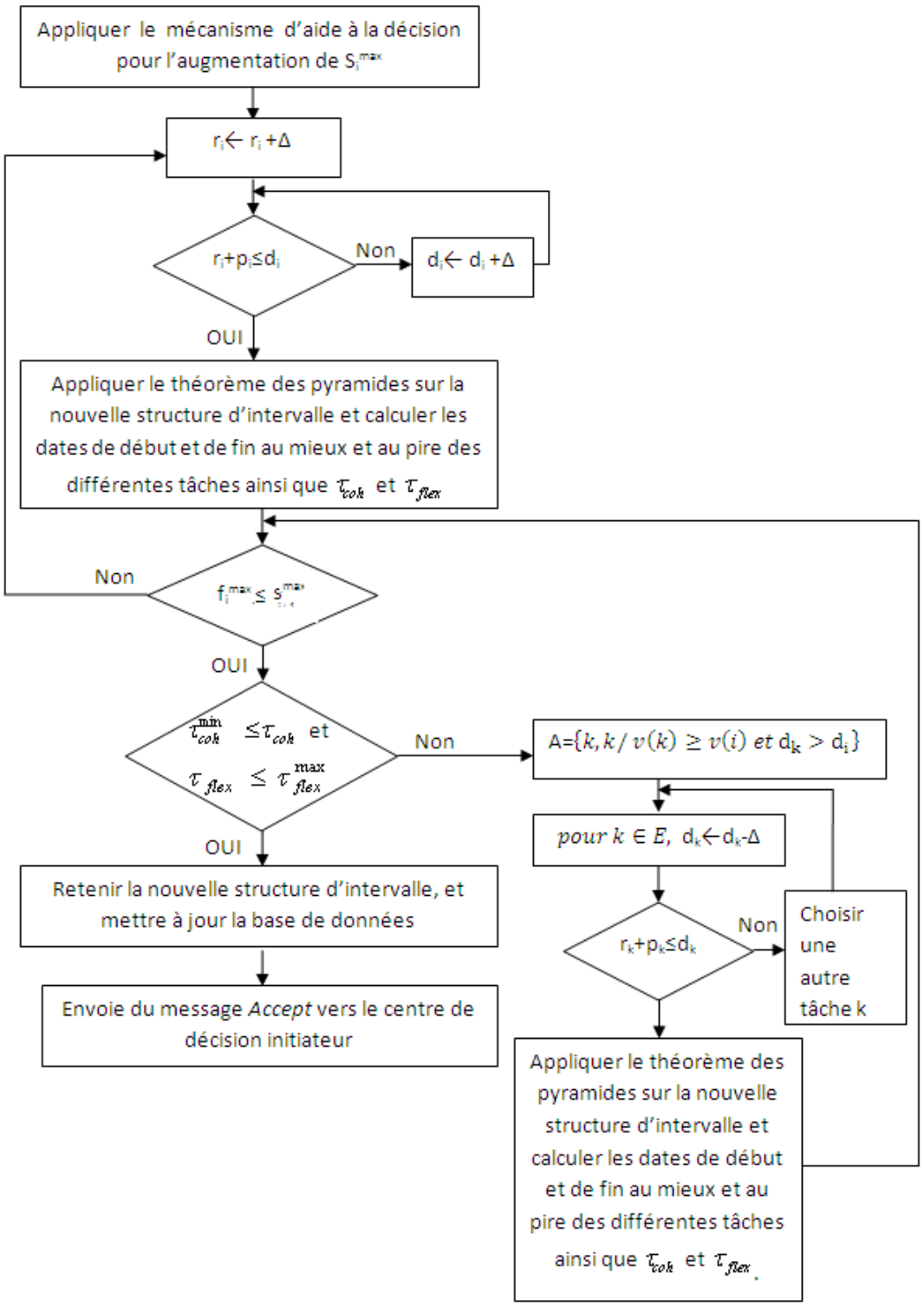


FIG. 3.12 – Plan local D

Chapitre 4

Implémentation et mise en oeuvre : Cas pratique d'un atelier Job Shop à quatre machines

---

Le présent chapitre illustre la mise en oeuvre de la méthode d'ordonnancement coopératif et distribué en simulant un cas pratique traitant du problème $J_4||C_{max}$,et en utilisant l'heuristique développée tout au long de ce travail. Nous décrirons tout d'abord le modèle du centre décisionnel retenu et son architecture en détaillant les différents modules qui le composent et l'intéraction qui existe entre eux, Nous préciserons ensuite l'architecture informatique du logiciel developpé en language C++, et les résultats de sa mise en peuvre sur le cas pratique. Pour finir, il sera question de définir les limites de la plateforme developpée dans l'optique de la rendre plus générale et plus apte à gérer des problèmes de taille plus élevée.

4.1 Modèle de centre décisionnel ordonnanceur retenu

Le développement de tout système coopératif amène à s'interroger sur la distinction entre les comportements d'un centre indépendants des actions des autres centres de décision du système, de ceux résultant d'interactions entre les centres de décision [WJK99]. Ces réflexions ont abouti à certains choix de représentation des comportements en distinguant d'une part les comportements autonomes, ne nécessitant dans leur accomplissement aucun contact avec les autres centres de décisions, et d'autre part les comportements sociaux spécifiant à un moment ou à un autre, une communication avec un centre de décision.

Cette distinction se traduit par la décomposition du modèle d'un centre décisionnel en un modèle individuel et un modèle social. Nous explicitons ci-après ces deux modèles en nous cliquant sur description générale de la figure 4.1.

Le modèle individuel contient les caractéristiques d'un centre de décision indépendantes de tout contexte social (i.e. les autres agents). Il établit les comportements autonomes de l'agent ainsi que ses compétences, ses critères de décision et ses objectifs individuels. Le composant "décision" du modèle individuel fournit un ensemble d'outils

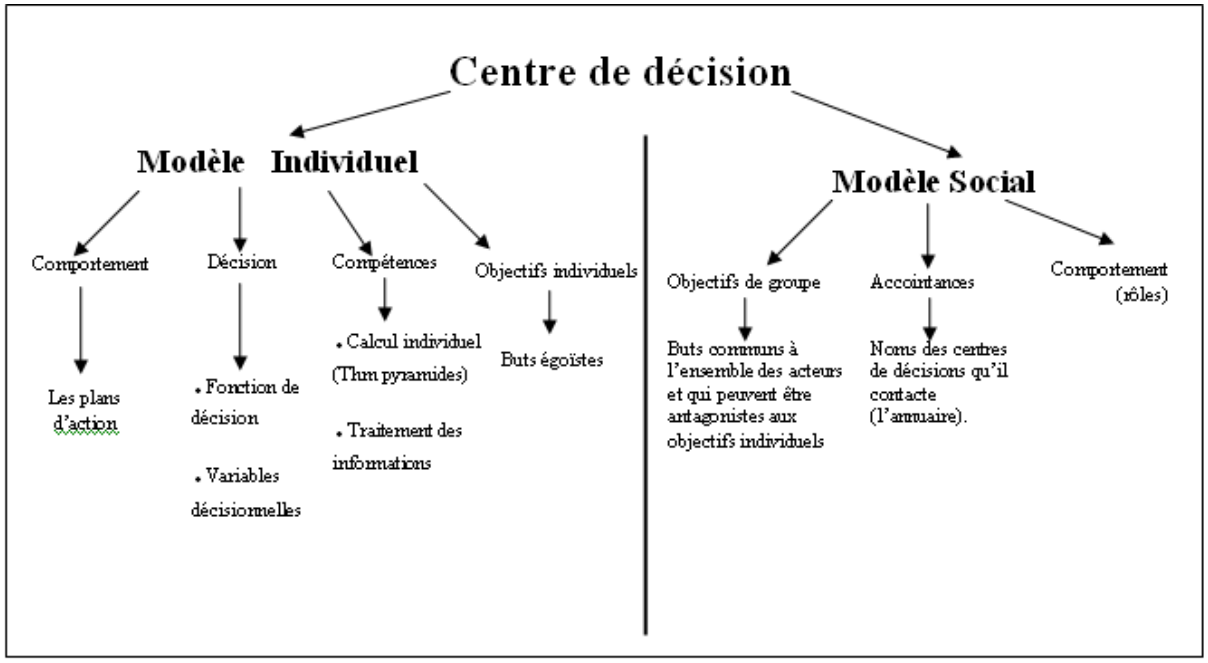


FIG. 4.1 – modèle du centre de décision ordonnanceur retenu

et de critères décisionnels supportant la prise de décision réfléchies d'un centre de décision. Les compétences d'un acteur décrivent ce qu'il est capable de faire d'un point de vue technique (compétences physiques) et d'un point de vue traitement de l'information (compétences cognitives).

Les objectifs individuels sont liés aux objectifs locaux au niveau décisionnel auquel il est associé. Du point de vue relations sociales, Nous avons retenu trois caractères principaux : les objectifs de groupe, les accointances et les comportements sociaux.

Les objectifs de groupe définissent des critères globaux d'optimisation à savoir dans notre cas la minimisation du C_{max}. Ils évitent ainsi que les centres décision totalement égoïste choisissent des solutions locales préjudiciables à l'ensemble de l'atelier de production.

La notion classique d'accointances, correspond simplement à l'annuaire des centres de décision contactés par un acteur. Les comportements sociaux sont décrits comme dans le modèle individuel, par des Plans Comportementaux ou plans d'action. Les comportements sociaux se traduisent par des interactions entre au moins deux acteurs adoptant chacun au moins deux rôles duaux. Ces rôles définissent la tâche de chacun des agents durant leurs interactions au travers d'échanges de message formant ainsi une conversation.

4.2 Architecture du centre de décision ordonnanceur retenu

L'architecture d'un agent ordonnanceur se décompose en quatre modules : *Communication*, *Connaissances*, *Décision* et *Expertise*. La figure 4.2illustre comment ces quatre modules interagissent pour dynamiser le comortement de ce centre de décision.

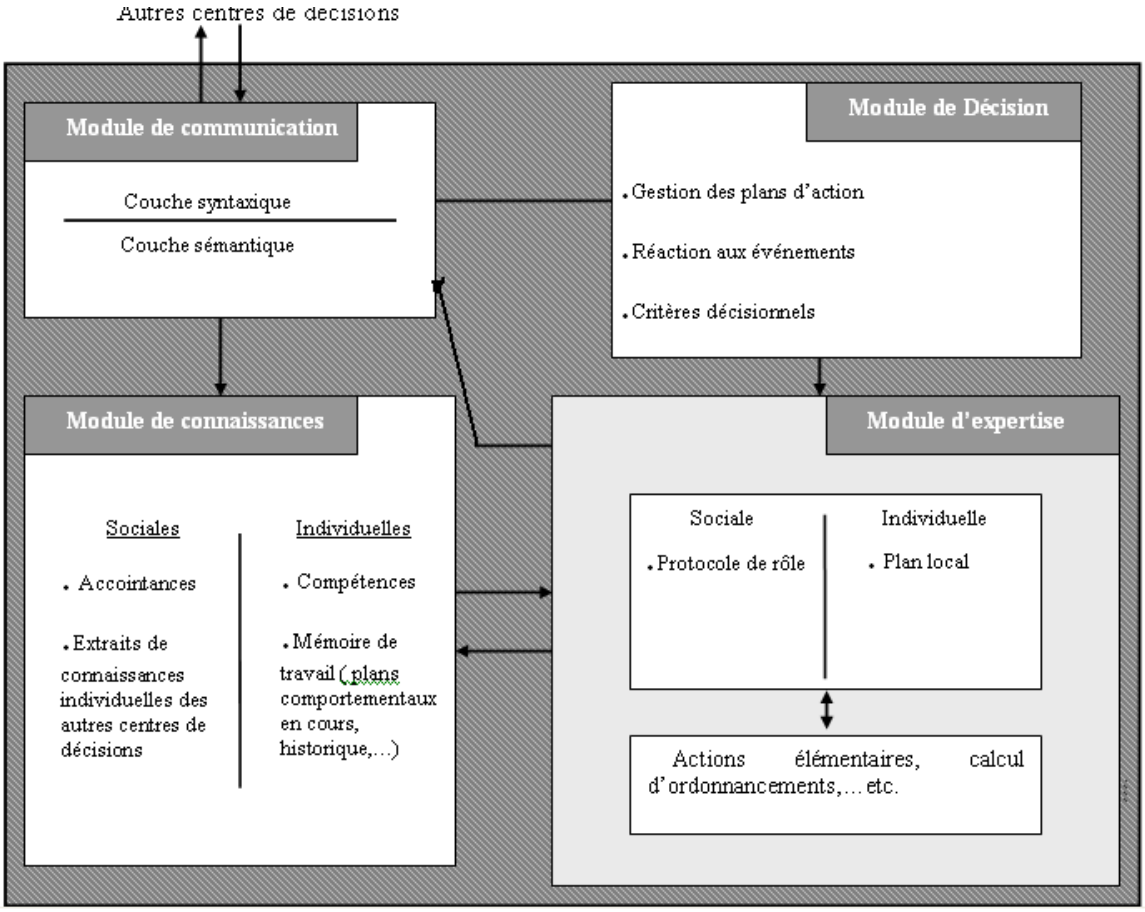


FIG. 4.2 – Architecture d'un centre de décision ordonnanceur

4.2.1 Module Communication

Le module "*Communication*" gère l'envoi et la réception des messages à destination ou en provenance des autres acteurs. Cette gestion concerne le médium linguistique de la transmission des messages, c'est à dire l'expression et l'interprétation d'un message. Pour qu'un message soit compréhensible par un centre de décision, il faut que ce message soit exprimé correctement dans un langage connu du centre de décision et qu'il porte sur des informations interprétables par cet acteur. En d'autres termes, le message doit être syntaxiquement et sémantiquement correct. Pour s'assurer que ces deux contraintes sont respectées, le module communication est organisé en deux couches : l'une dite *syntaxique*

et l'autre *sémantique*.

La couche syntaxique vérifie simplement que le message respecte les règles grammaticales du langage utilisé. La couche sémantique s'assure de la bonne compréhension du message, en vérifiant que le contexte du message est connu. Le contexte est formé du nom du protocole, de l'identifiant de la conversation et éventuellement de l'enveloppe. Lorsqu'un message est identifié comme étant hors contexte, un message d'erreur est renvoyé indiquant si le protocole ou la conversation sont inconnus, ou encore si la conversation spécifiée est déjà terminée.

On distingue deux modèles de communication :

1. Communication par partage d'information
   - mode adopté dans les systèmes à tableau noir
   - communication via le tableau noir.
2. Communication par envoi de messages
   - agents en liaison directe
   - Envoi directement et explicitement au destinataire

4.2.2 Module Connaissance

Le module "*Connaissances*" contient l'ensemble des informations détenues par un centre de décision sur lui-même (connaissances individuelles), ou sur les autres (connaissances sociales). Les connaissances individuelles d'un centre de décision ordonnanceur, forgent son identité, c'est-à-dire qu'elles identifient ce qu'il sait faire (compétences physiques et cognitives) et ce qu'il fait ou va faire (croyances métier et mémoire de travail). Les connaissances sociales caractérisent son implication dans le groupe. Elles portent tout d'abord sur les accointances, mais eut aussi comporter des extraits des connaissances individuelles de certains des autres centres de décision ordonnanceurs.

Ainsi si un centre de décision recherche une coopération avec d'autres centres de décisions il se réfère à ses connaissances sociales pour ne s'adresser qu'à un ensemble restreint et pertinent d'acteurs. Ce mécanisme présente l'avantage de minimiser les flux de communication en évitant un recours systématique au broadcast [1].

Dans notre cas les connaissances individuelles sont par exemple, les données relatives aux dates de début au mieux et au pire des tâches d'un même centre de décision, et les connaissances sociales sont les accointances grâce auxquelles le centre de décision sait qui exécute quelle tâche.

4.2.3 Module Expertise

Alors que les compétences dans le module "*Connaissance*" décrivent ce que sait faire le centre de décision, le module "*Expertise*" détaille le comment de ces compétences.

[1] dite aussi méthode de diffusion des messages sur tableau noir, cette méthode se caractérise par le fait que tout les messages échangés sont visibles par tous les centres décisionnels.

Le module Expertise contient en effet l'ensemble des Plans Comportementaux (PC) à savoir les plans locaux (plans A 3.9, B 3.10,C 3.11 ,et D 3.12) et des actions élémentaires exécutables par un centre de décision. Retenons qu'un plan spécifie une séquence d'actions élémentaires tels que l'envoi de message, des calculs divers (dont d'ordonnancement), la mise à jour des connaissances,...etc.

4.2.4 Module Décision

Le module "*Décision*" contrôle l'exécution des actions entreprises au niveau du module "Expertise". C'est le module le plus important car c'est sur lui que repose la réactivité du centre de décision ordonnanceur et sa capacité de réagir aux perturbations.

4.3 Implémentation et mise en oeuvre de la méthode

4.3.1 description de la plateforme de simulation

La plateforme de test est développée en utilisant le ***language C++***. En effet, des classes de ressources son construites dynamiquement et contiennent chacune, les différentes procédures qui permettent de calculer des données telles que les dates de disponibilité et les deadlines des tâches, et leurs dates de début et de fin au mieux et au pire.

Soulignat tout de même, qu'il existe de multitudes de plateformes préconçues pour les systèmes multi-agents qui auraient pu acceuillir notre protocole de négociation. A titre d'exemple nous citons le language LALO basé lui aussi sur le language C++ [Tra01], les languages CLIPS et JESS basés sur le language JAVA [Tra01].Notre choix a été dicté par le fait que l'adoption d'une plateforme déja préconçue pour les systèmes multi-agent allait nous obliger à nous alligner sur certains formalismes que les SMA imposent. Pour éviter d'avoir une contrainte supplémentaire, nous avons opter pour construire notre propre plateforme qui est sans aucun doute plus attentive au contexte de l'étude, quit à tout redéfinir.

Le programme se scinde en deux grands volets, à savoir un volet contenant les procédures qui sont executées sur tous les agents ordonnanceurs, et dont le but serait d'effectuer les calculs préliminaires (précédants les négociations), et les calculs intermédiaires (survenant pendant et après négociation), et un volet consacré aux procédures intervenant lors de la négociation, telles que les routines qui permettent la prise en charge des messages, le remplissage des journaux de bords de chaque agent ordonnanceur (historique des messages envoyés et reçus) et les plans d'actions.

Initialement le programme nous donne la main pour choisir la configuration de l'atelier. Le nombre de tâches et de gammes opératoires peut aller de 1 jusqu'à 99.

Dans le but de permettre une certaine interactivité avec l'utilisateur, et afin de préserver l'abstarction nécessaire à tous système informatisé, l'interface qu'offre le logiciel permet de rentrer manuellement les données du problème à savoir le nombre de tâches, de machines, les longueurs des gammes opératoires et les durées opératoires.

Le programme nous permet aussi de générer de manière aléatoire toutes ces données, et ce grâce à la fonction *Random*. Cette dernière option est particulièrement séduisante dans phase de test et d'évalutation du logiciel.

Ci après un exemple d'une configuration à 10 tâches, 4 machines, et 4 jobs. les durées opératoires et la distribution aléatoire des machines et des gammes opératoires ont été obtenus en cliquant sur les onglets random opérations, random machines, et random gammes , qui se situent dans le menu fichier.

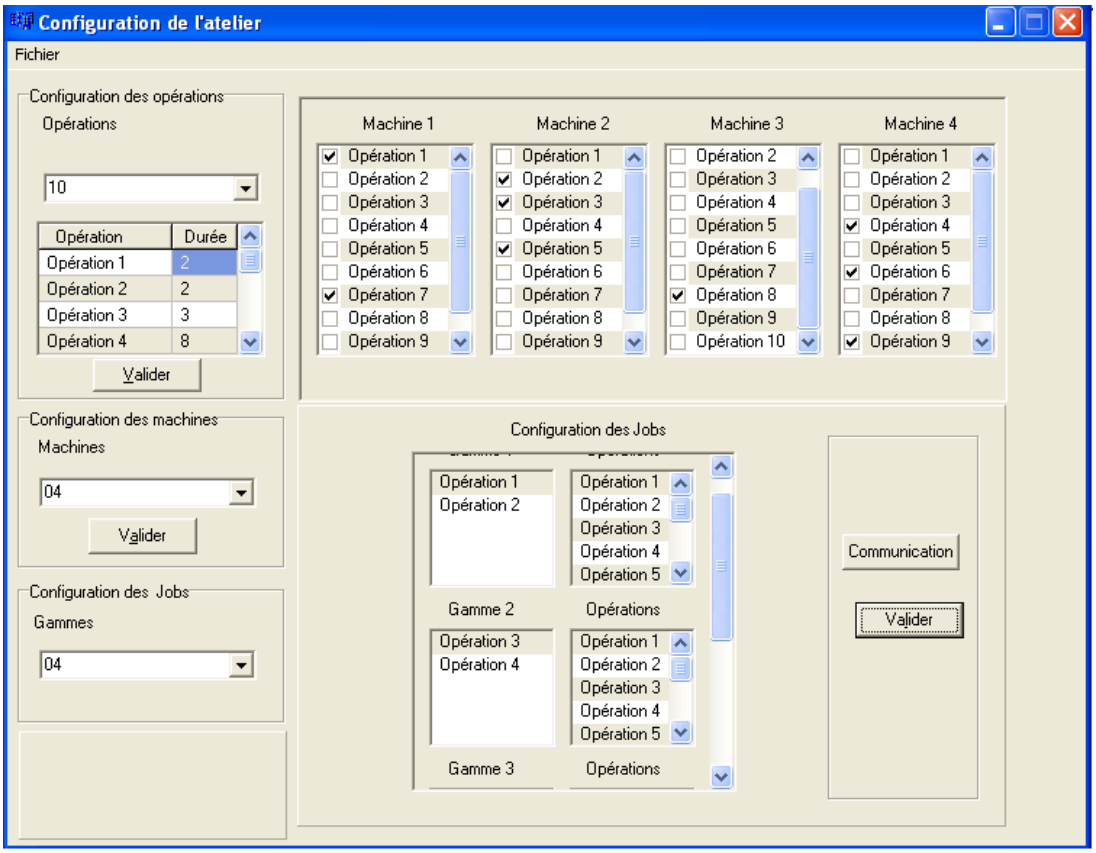


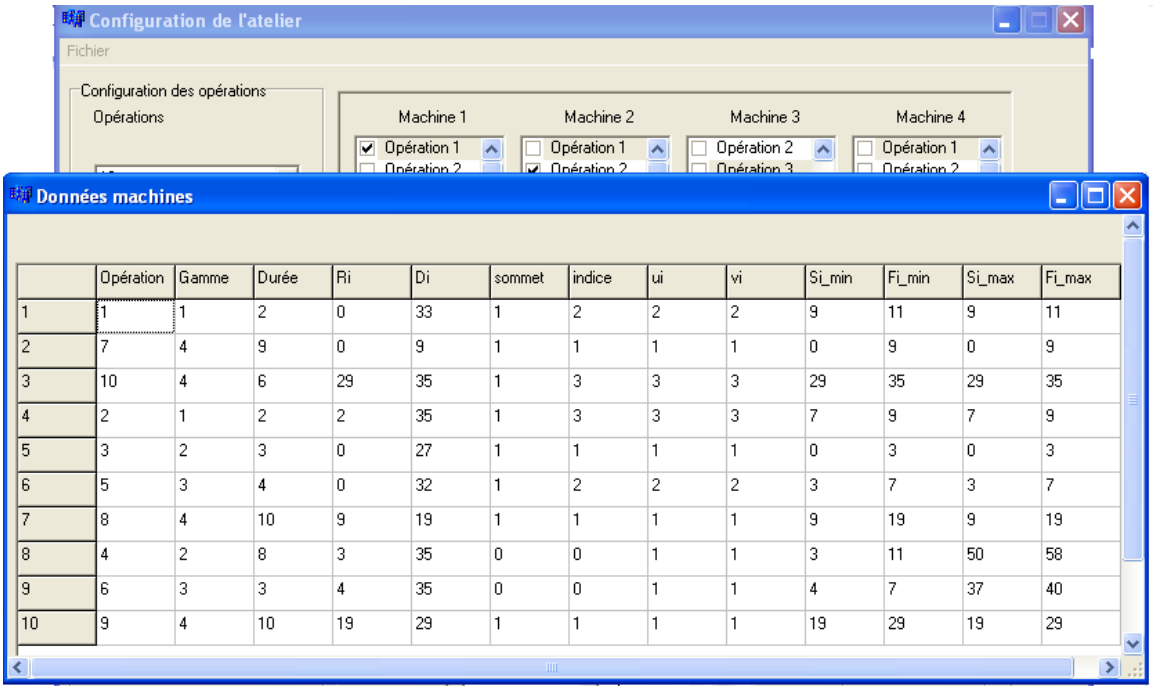
FIG. 4.3 – Exemple d'une configuration de l'atelier

En cliquant sur le bouton *valider*, on appelle les routines qui font le calcul dérivant du théorème des pyramides, à l'issue de ce calcul un tableau contenant toutes les données de bases de l'exemple, est affiché.

Pour des raisons de clareté nous avons organisé l'affichage des données relatives à chaque opération, de manière à ce que les opérations soient groupées selon la ressource qui les exécutent, et selon l'ordre croissant de leurs indices.

La table de données est composée de 13 colonnes, et contient diverses informations importantes telles que les indices de la première et de la dernière pyramide auxquelles appartient chaque opération, et les dates de début au mieux et au pire.

Ces informations sont calculées de manière complètement distribuée puisque chaque agent ordonnanceur possède sa propre classe et ses propres routines, ce qui simule un certain parallèlisme dans les calculs, augmantant ainsi les performances de la plateforme.



	Opération	Gamme	Durée	Ri	Di	sommet	indice	ui	vi	Si_min	Fi_min	Si_max	Fi_max
1	1	1	2	0	33	1	2	2	2	9	11	9	11
2	7	4	9	0	9	1	1	1	1	0	9	0	9
3	10	4	6	29	35	1	3	3	3	29	35	29	35
4	2	1	2	2	35	1	3	3	3	7	9	7	9
5	3	2	3	0	27	1	1	1	1	0	3	0	3
6	5	3	4	0	32	1	2	2	2	3	7	3	7
7	8	4	10	9	19	1	1	1	1	9	19	9	19
8	4	2	8	3	35	0	0	1	1	3	11	50	58
9	6	3	3	4	35	0	0	1	1	4	7	37	40
10	9	4	10	19	29	1	1	1	1	19	29	19	29

FIG. 4.4 – résultat des calculs du théorème des pyramides

Ce sont ces mêmes informations qui vont être exploitées dans le procéssus de négociation qui sera amorcé dès qu'on appuit sur le bouton *communication*. En effet, dès lors qu'on a entamé la phase de négociation, il y a élection du premier job à prendre en charge pendant la négociation.

Les ressources négocient une première fois pour élire le job à prendre en charge, leur choix devrait porter sur le job ayant la plus petite marge entre la fin de sa dernière tâche et le délai négocié avec le client. Ce choix est justifié par le fait que le premier job à être pris en charge bénéficie de plus de chance d'exécuter les plans locaux et donc, de réduire les valeurs de fin localement, plutôt que d'émettre des contre-propositions qui sont synonymes de l'augmentation des valeurs de début et de dégradation des performances globales 3.3.3.

Pour assurer le processus communicatif, nous avons utilisé ***les sockets TCP*** predéfinies dans le ***Builder C++***. En effet, à chaque ressource seront réservées deux sockets :

– ***Une socket client*** servant à envoyer les messages,

– ***Une socket serveur*** servant à recevoir les messages.

Les sockets serveurs présentent la particularité d'être à l'écoute des ports qui les carctérisent, dans l'attente d'un quelconque message.

Afin d'émettre des messages, les sockets clients quant à elles, se mettent sur le port qui caractérise la socket serveur pour laquelle le message est destiné.

Les messages doivent obligatoirement être formulés suivant le format suivant :

FIG. 4.5 – format du message

Tel que :

- Exp désigne l'indice de la ressource qui expédie du message.
- Dest désigne l'indice de la ressource déstinataire.
- Objet désigne l'objet du message, l'objet 1 corresond à une *proposition* de contrat, l'objet 2 à une *contre proposition*, l'objet 3 à *accept*, l'objet 4 correspond au *test faisabilité*, l'objet 5 est la réponse faisable,l'objet 6 est la réponse non faisable.
- Tâche désigne la tâche concernée par le message.
- Borne_inf et Borne_sup désignent respéctivement les dates de début ou de fin relatives à la tâche concernée.

Afin de suivre l'évolution du processus de négociation nous avons inséré dans le programme des interruptions qui permettent d'afficher les messages envoyés et reçcus par les différents agents ordonnanceurs , et des différentes étapes du protocole des négociation comme par exemple l'état d'incohérence du couple de tâches concerné ainsi que le passage au différents plans d'actions.

La suite du déroulement du programme dépend des données insérées et des possibilités qu'offre le protocole de négociation à savoir les différents plans comportementaux, et les cycles de négociation sont différents d'un exemple à un autre.

4.3.2 Cas pratique : atelier job Shop à quatres machines

Nous considérons un atelier de production composé de 3 machines, et 10 opérations réparties suivant 4 gammes opératoires comme indiqué sur le tableau 4.1. Ce cas d'étude nous a permis d'identifier les formes de coopération possibles entre les machines de production de cet atelier. Nous étudions aussi dans cette partie, comment la méthode de réordonnancement coopératif peut être mise en oeuvre.

Travail	**tâche**	p_i	M_j
1	1	7	1
1	2	10	2
1	3	5	3
2	4	3	2
2	5	15	1
3	6	30	3
3	7	3	1
3	8	2	2
4	9	6	3
4	10	3	1

TAB. 4.1 – Données du cas pratique

4.3.3 Mise en oeuvre de la méthode d'ordonnancement coopératif et distribué sur le cas pratique

Dans cette partie nous allons adopter un traitement coopératif de l'atelier de production présenté. En effet, l'outil informatique développée dans ce travail, nous permet de simuler les étapes du protocole de négociation, et de réaliser un ordonnancement parfaitement distribué.

En effet, pour commencer visualisons comment est-ce-que les données du problèmes sont introduites dans le logiciel.

Lorsqu'on appuit sur le bouton *valider* le tableau 4.7 contenant les résultats du théorème des pyramlides apparait.

4.3.4 résultats et discutions

Lorsqu'on appuit sur le bouton *communication*, le processus de négociation s'amorce et nous obtenons une suite d'évènements. Ainsi, des propositions sont envoyées, et la recherche d'état de cohérence des couples de tâches est enclenchée, à titre indicatif les tâches 6 et 7 étant cohérentes 4.8, nous détectons le couple de tâches incohérentes (9,10), 4.9.

Dès que l'incohérence est détéctée, le programme la prend tout de suite en charge en faisant l'appel au Plan comportemental A afin de diminuer la date de fin au pire de la tâche 9 exécutée sur la machine 3 4.10.

Comme il est impossible de diminuer la date de fin au pire de la tâche 9, il y a passage au Plan comportemental C pour augmenter la date de début du travail 10 en éliminant les séquences dominantes faisant commencer 10 trop tôt.

Ainsi, on arrive à rendre cohérent ce couple de tâches critique, on diffuse la nouvelle structure d'intervalle et on réussit à rendre le système parfaitement cohérent 4.11.

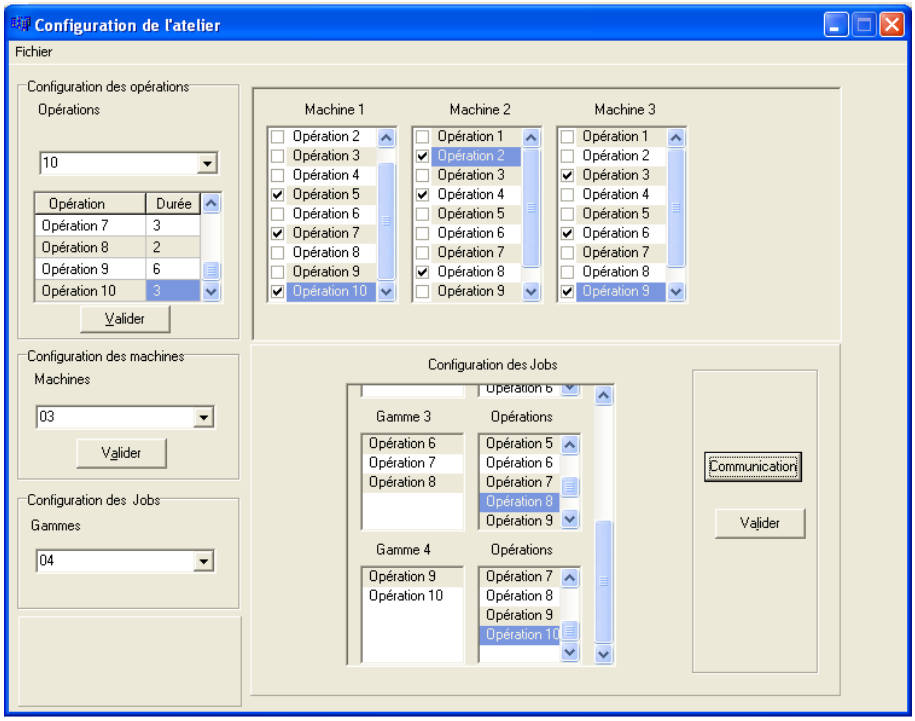


FIG. 4.6 – données du problèmes

Nous avons donc obtenu un système parfaitement cohérent 4.12, ou autrement dit une structure d'intervalle cohérente. En effet, soulignant le fait que notre logiciel manipule exclusivement des structures d'intervalles, permettant de définir un ensemble de solutions et non pas une solution solution d'ou l'apport en robustesse.

Cependant, il est vrai que dans notre approche le contrainte de performance nous obligent à approcher le plus possible la solution optimale calculée par d'autres méthodes qui ne se basent pas forcément sur l'aspect coopératif.

4.4 Limites de l'approche développée

Une des critiques fondamentales de la plateforme développée est le surbookage des sockets lorsque le nombre de machines ou des tâches augmente. En effet, le système que nous essayons de reconstituer sur notre plateforme est un système parfaitement parallèle (chaque centre de décision est un système informatique à part), alors que notre plateforme est construite sur le même système d'exploitation qui ne peut exécuter plusieurs tâches en parallèle.

L'idéal serait d'implémenter notre plateforme sur plusieurs PC relié en réseau et de les faire communiquer par les moyens classiques de communication au sein d'un réseau. C'est à dire assigner à chaque PC une adresse ip de manière à ce que toutes les adresses appartiennent au même sous réseau, et envoyer des requêtes TCP à travers le réseau. Cette

solution demandant beaucoup de moyens, nous nous somme contenté de tout simuler sur le même pc, mais ceci ne remet nullement en cause la validité des résultats obtenus pour des systèmes dont le nombre de machines est peu élevé.

Afin de contourner cette limitation, et pour les systèmes dont le nombre de machines est assez important, nous proposons de regrouper plusieurs machines afin d'obtenir des pools. En un premier temps, on faira négocier les machines d'un même pool entre elles de manière à les rendre cohérentes, puis on faira négocier les pools entre eux. La figure 4.14) résume ce principe.

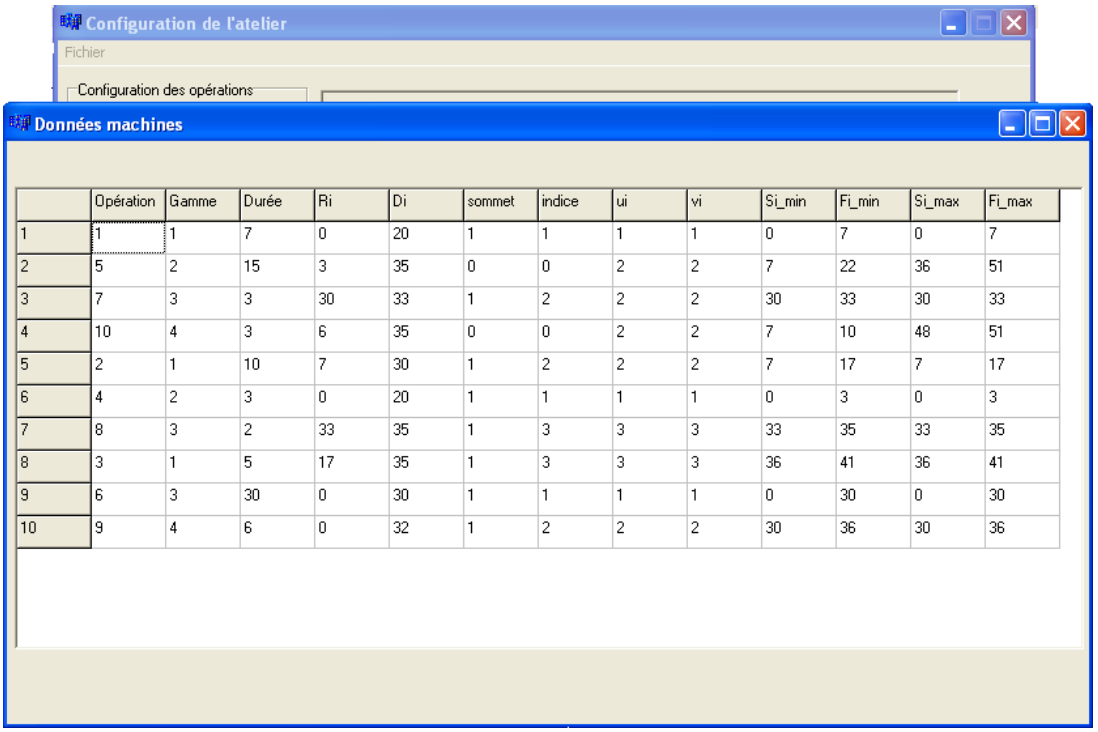
En effet, dans ce schéma, nous considérons que le nombre de machines dans chaque pool est n_1 et que le nombre total des pools est n, le nombre initial des machines étant de nxn_1,nous réduisons ainsi considérablement les risques de bloquage du programme du à un trafics de sockets trop important. Le tout serait alors de trouver le critère sur lequel on se base afin de créer les pools.

Pour finir, notre logiciel ne prend pas encore en charge l'insertion des tâches arrivant en temps réel. Pour ce, il serait intéressant d'intégrer les règles définies dans le chapitre 2.

4.5 Conclusion

Ce chapitre, la mise en oeuvre de la méthode de coopération décrite dans le chapitre 3, est présentée. Le travail de mise en oeuvre aboutit à la proposition d'un prototype informatique. Enfin, ce prototype a été utilisé pour simuler le comportement des centres décisionnels dans un atelier de production Job Shop de Quatre machines.

Nous avons aussi fait une démonstartion de notre logiciel sur un exemple déja traité tout au long de ce travail et dont on connait la solution optimale, et nous avons déduit que notre prototype informatique fournissait la même solution optimale avec un temps d'execution de l'ordre de quelques milli-secondes.



	Opération	Gamme	Durée	Ri	Di	sommet	indice	ui	vi	Si_min	Fi_min	Si_max	Fi_max
1	1	1	7	0	20	1	1	1	1	0	7	0	7
2	5	2	15	3	35	0	0	2	2	7	22	36	51
3	7	3	3	30	33	1	2	2	2	30	33	30	33
4	10	4	3	6	35	0	0	2	2	7	10	48	51
5	2	1	10	7	30	1	2	2	2	7	17	7	17
6	4	2	3	0	20	1	1	1	1	0	3	0	3
7	8	3	2	33	35	1	3	3	3	33	35	33	35
8	3	1	5	17	35	1	3	3	3	36	41	36	41
9	6	3	30	0	30	1	1	1	1	0	30	0	30
10	9	4	6	0	32	1	2	2	2	30	36	30	36

FIG. 4.7 – Résultats du théorème des pyramides

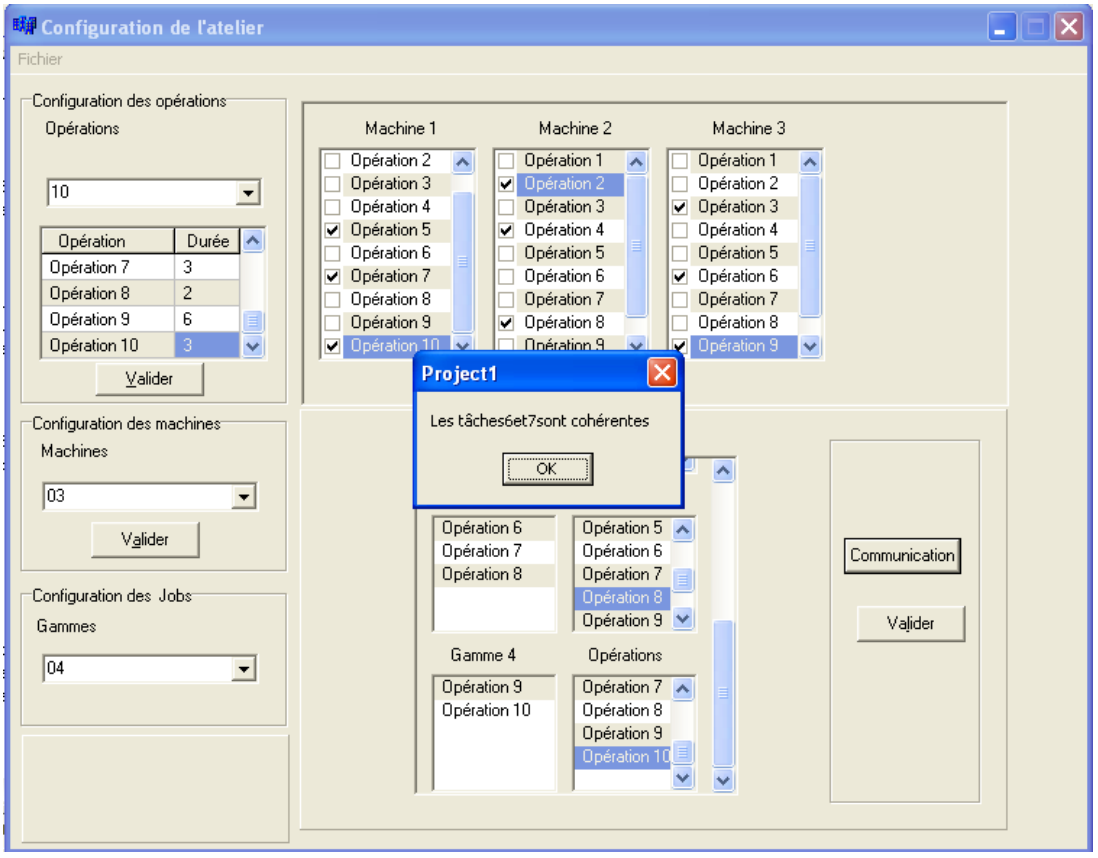


FIG. 4.8 – Exemple d'état de cohérence du couple de tâches(6,7)

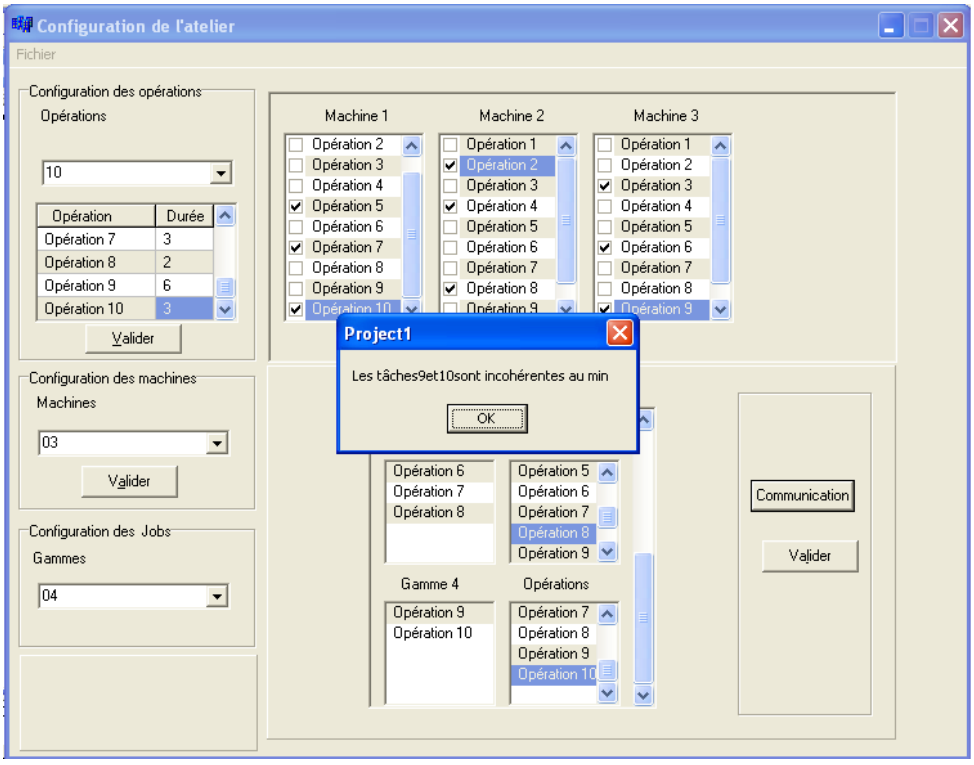


FIG. 4.9 – Détection de l'incohérence du couple de tâches(9,10)

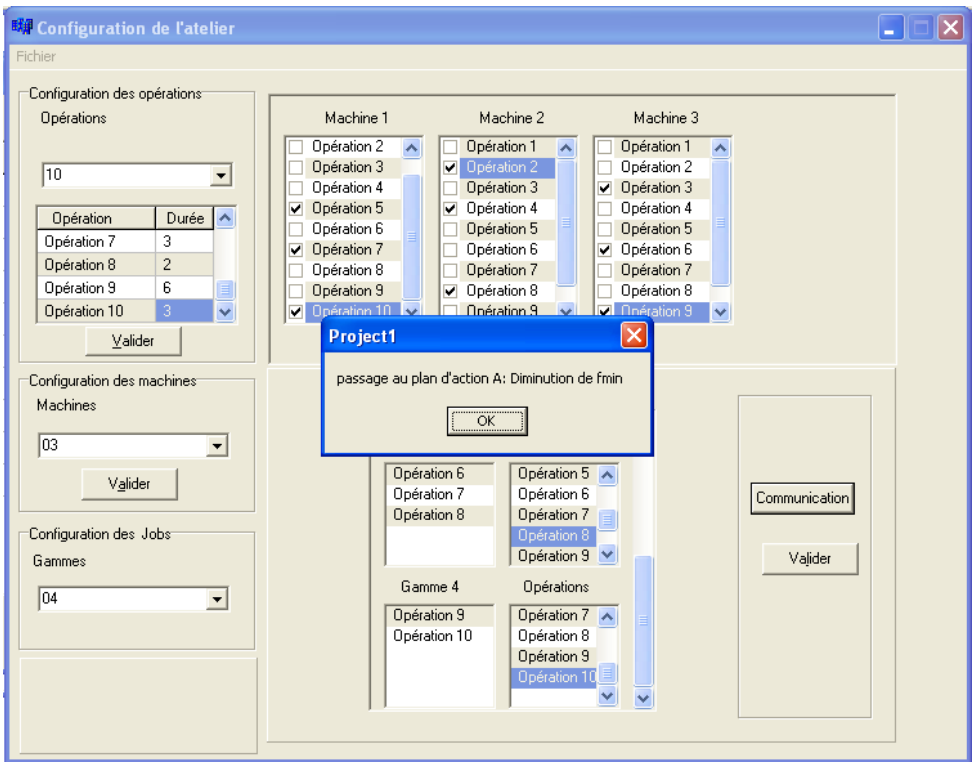


FIG. 4.10 – Passage au plan local A pour tenter de rendre le couple de tâches (9,10) cohérent

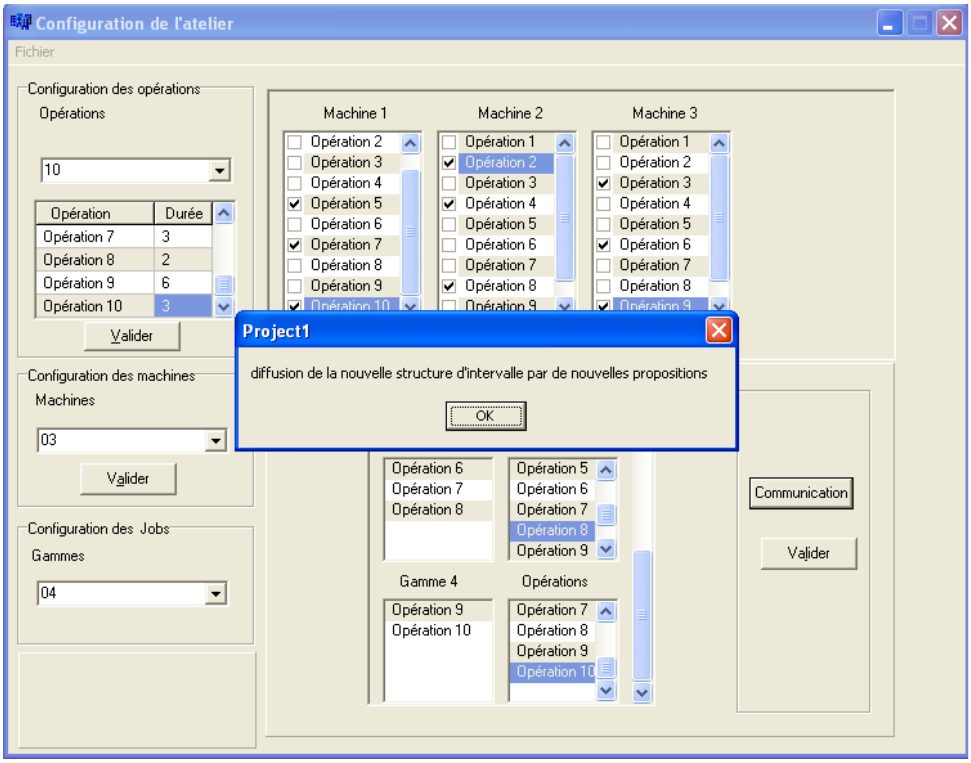


FIG. 4.11 – Diffusion de la nouvelle structure d'intervalle obtenu après exécution du PlanA

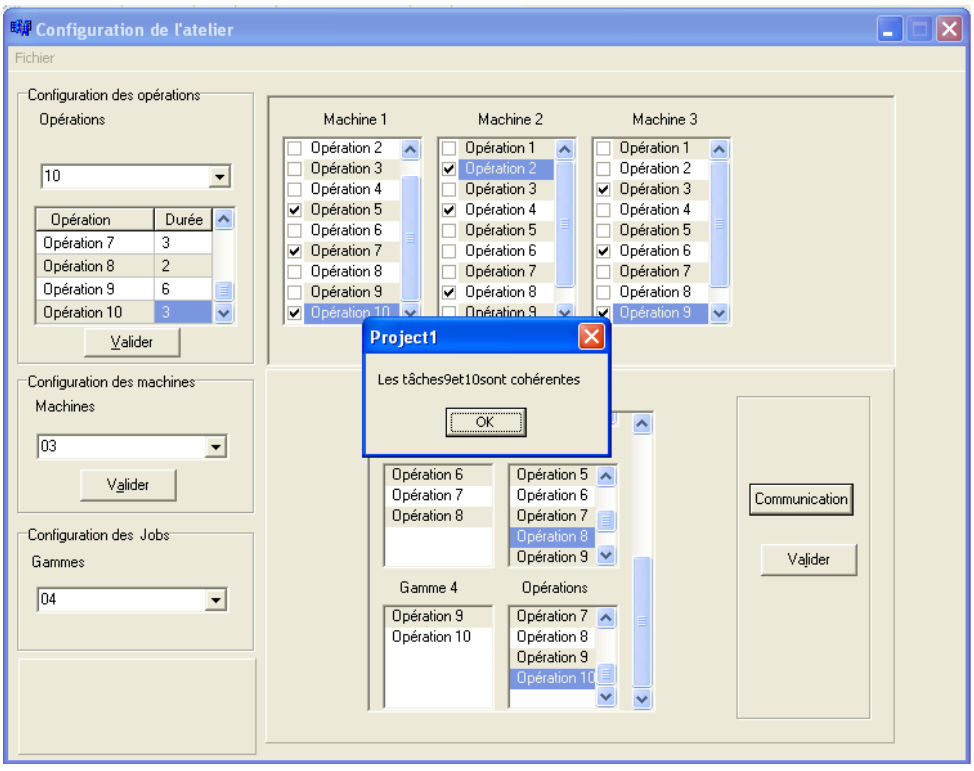


FIG. 4.12 – Les tâches 9 et 10 redeviennent cohérentes

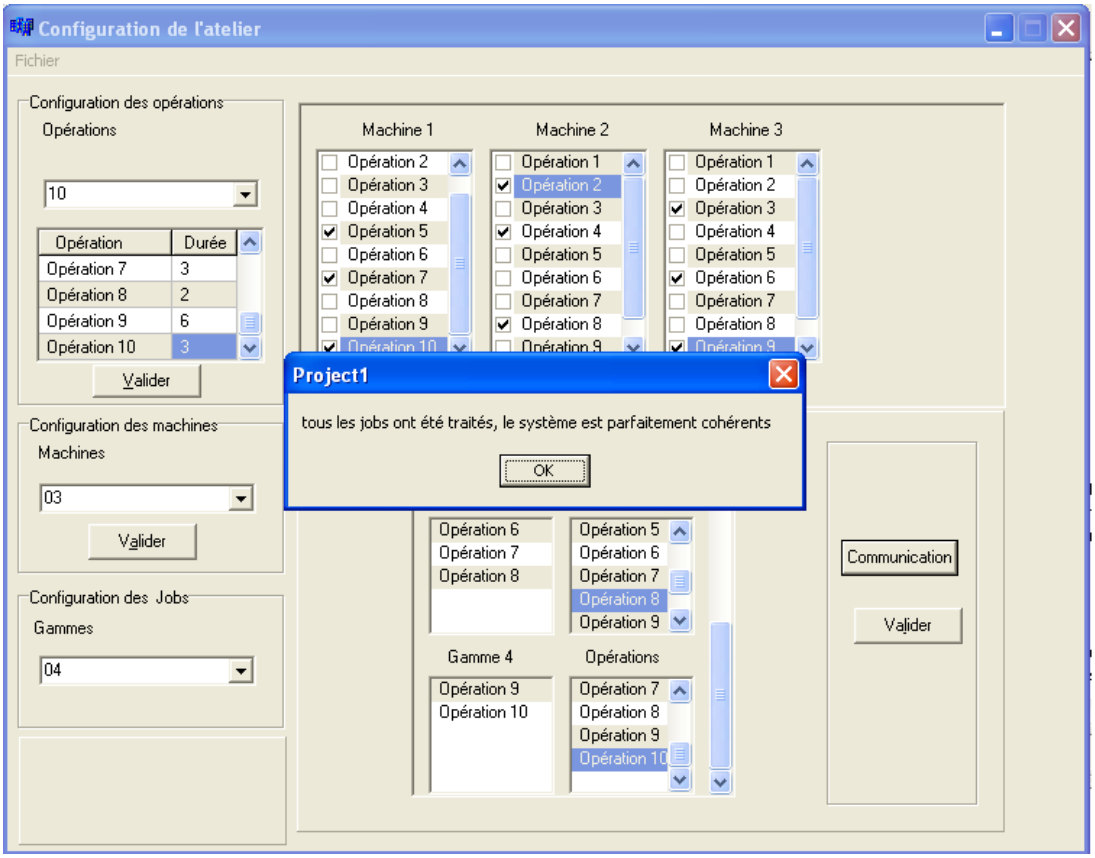


FIG. 4.13 – Fin du processus coopératif après que tous les cas d'incohérence ne soient traités

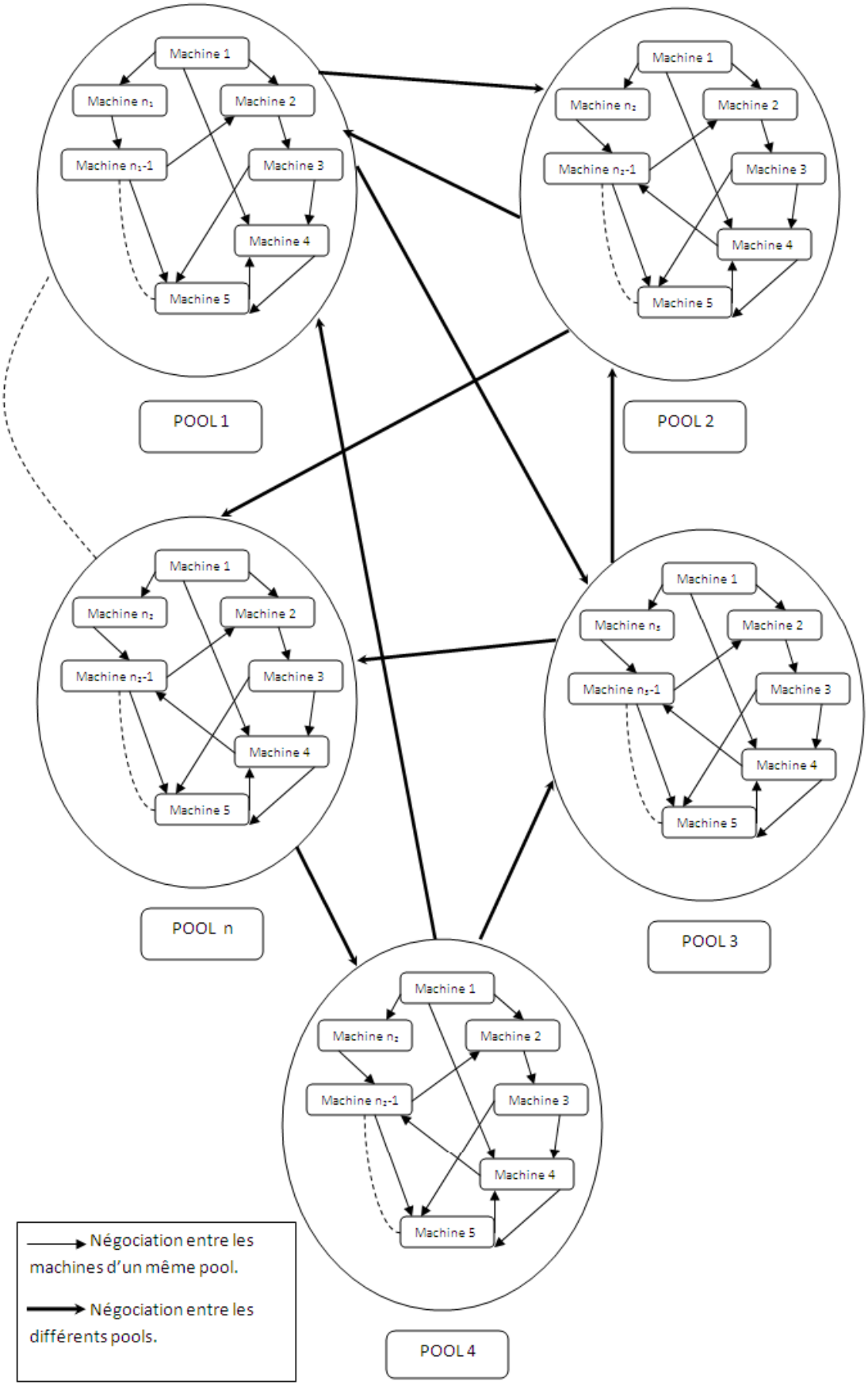


FIG. 4.14 – négociation entre pools

Conclusion générale et perspectives

Plusieurs études prospectives consacrées à l'ordonnancement ont montré l'écart important entre la recherche théorique et les besoins pratiques, en pointant notamment du doigt le fait que, les méthodes d'ordonnancement étant pour la plupart déterministes, ne prennent pas suffisamment en compte le caractère incertain de leur environnement de mise en oeuvre. De ce fait, une solution déterministe, bonne à un instant donné, peut s'avérer rapidement obsolète, suite à une perturbation même mineure. Pour pallier ce problème, des méthodes d'ordonnancement, dites robustes, ont été développées, ayant la capacité de *résister* aux perturbations de l'environnement tout en assurant une stabilité de la performance. Le travail présenté dans ce mémoire s'inscrit dans ce champ de recherche.

Certaines méthodes d'ordonnancement proactives-réactives proposent d'ajouter une flexibilité soit temporelle, soit séquentielle, à une solution d'ordonnancement afin de faire face aux incertitudes. Dans ce cadre, la nécessité d'un compromis flexibilité / performance a été mis en évidence, augmenter la flexibilité d'un ensemble de solutions se traduisant généralement par une perte de performance, et vice-versa. Notre travail s'est principalement focalisé sur les méthodes produisant de la flexibilité séquentielle, dans une optique de recherche de compromis, ce type de flexibilité induisant en effet de la flexibilité temporelle et semblant avantageux pour traiter un panel très large de perturbations.

Plusieurs méthodes d'ordonnancement produisent de la flexibilité séquentielle en exploitant le concept de groupes de tâches permutables. Ce concept est certes, intéressant à plusieurs titres, toutefois, il présente quelques limites. En effet, seules des tâches placées de façon contiguë sur la même ressource peuvent être permutées, ce qui limite quelque peu la flexibilité séquentielle pouvant potentiellement être générée. D'autre part, la constitution de groupe de tâches ne tire pas partie d'une modélisation a priori des incertitudes, du coup il serait impossible de quantifier la robustesse apportée.

De plus, la pluart des méthodes abordent le problème de manière tout à fait centralisée, en négligeant les capacités individuelles de chacune des ressources assimilées à des centres décisionnels.

Ce sont ces trois constats qui ont motivés nos recherches. En effet, ce travail s'inscrit dans le cadre de la contribution à la recherche d'heuristiques performantes pour la résolution du problème Job Shop à plusieurs machines avec minimisation du C_{max}, et l'approche abordée est une approche par coopération dans le cadre d'un ordonnancement robuste distribué.

Le problème global est décomposé en plusieurs sous-problèmes à une machine, chacune gérant son propre ordonnancement robuste. Cet ordonnancement robuste est calculé grâce au théorème des pyramides qui définit un ordre partiel dominant pour le problème $1|r_i|L_{max}$. Ceci a pour conséquence directe d'apporter de la fléxibilité séquentielle indispensable pour la construction d'un ordonnancement global cohérent et capable de faire face à l'arrivée de nouvelles tâches.

La solution globale résulte de la superposition des ordonnancements locaux calculés au sein de chaque centre de décision individuellement, et corrigés pour être cohérents, grâce à un processus coopératif.

Dans un premier temps, nous nous sommes focalisés sur l'étude du problème à une machine $1|r_j|L_{max}$. Pour ce problème, nous avons montré l'intérêt d'un théorème de dominance démontré dans les années quatre-vingt. L'avantage principal de ce théorème réside dans sa capacité à caractériser un ensemble dominant de séquences, sans les énumérer, grâce à un ordre partiel construit sur un corps d'hypothèses restreint, ne considérant exclusivement que les relations d'ordre entre les dates de début au plus tôt et de fin au plus tard des travaux. Un intérêt de cet ordre partiel est qu'il présente potentiellement davantage de flexibilité séquentielle que l'ordre partiel relatif au concept de groupe de tâches permutables. Ensuite, nous avons présenté des algorithmes qui calculent en un temps polynomial les performances au mieux et au pire de l'ensemble caractérisé par le théorème des pyramides. Enfin, nous avons montré qu'il était possible de calculer les marges libres existantes dans la séquence au pire d'une tâche, et ce, dans la perspective d'utiliser ces marges libres pour insérer les jobs arrivant en temps réel.

Dans un deuxième temps, ce travail structure la relation de coopération liant les différentes ressources assimilées à des centres de décision dans le cadre d'un ordonnancement distribué. La formalisation de la coopération proposée s'appuie principalement sur un modèle structurel du centre décisionnel, à savoir la décomposition en quatre modules : *Communication*, *Connaissances*, *Décision* et *Expertise*. La coopération correspond à trois fonctions distinctes : négocier, renégocier et coordonner. La négociation et la renégociation sont réalisées à l'aide de conversation asynchrone, fondées sur l'échange de propositions et de contre-propositions, visant à déterminer un cadre de décision accepté par les deux partenaires. La coordination consiste en l'envoi de messages informatifs, à des moments clés où dès qu'un aléa se produit. Ces messages concernent les caractéristiques liées aux dates de début au mieux et au pire des différentes tâches.

Pour guider les centres de décisions lors de la prise de décision concernant la modification des structures d'intervalle, une aide à la décision a été élaborée. Elle est basée sur des mécanismes de diminution ou d'augmentation des dates de début ou de fin des tâches présentant un état d'incohérence vis à vis de contraintes que nous avons définis.

Ce travail propose un prototype logiciel permettant d'instrumenter la relation de coopération inter-machines. Cet outil permet de simuler le processus coopératif et le protocole de négociation développé. Cette plateforme simulant ce processus coopératif a été définie en language C++, et elle permet de tester le protocole établit sur diverses confi-

gurations de l'atelier Job Shop.

Le cas pratique d'un atelier $J_4||C_{max}$ a été traité, et les résultats obtenus par l'application de l'heuristique developpée dans ce manuscrit sont complètement satisfaisants.

Les algorithmes développés dans ce travail sont encore au stade d'évaluation et plusieurs améliorations peuvent leurs être apportées. notamment en ce qui concerne les plans d'actions et les mécanismes d'aide à la décision pour la mocification des structures d'intervalle.

De plus, notons que pour évaluer les performances de l'heuristique, il serait intéressant d'implanter la plateforme développée sur un vrai réseau informatique distribué, dans un premier temps, puis dans un second temps sur un réseau d'automates, et d'y effectuer des bilans de tests benchmarks. Car ce travail se base plus sur la faisabilité de l'approche proposée, que la quantification de ses performances.

Une des perspectives intéressantes de ce travail serait la recherche d'un meilleur compromis robustesse/performance, en proposant d'autres critères permettant de choisir l'ordre de traitement des jobs, et l'écart avec lequel les dates de début ou de fin des tâches sont augmentées ou diminuées lors de l'execution des plans locaux. Une autre perspective serait d'enrichir l'outil développé en y implémentant la méthode de réordonnancement définie dans le deuxième chapitre.

Pour finir, notons qu'il serait intéressant aussi, d'étendre l'algorithme proposé à d'autres problèmes que le Job Shop, ou à d'autre critère que la minimisation du C_{max}

Bibliographie

[ABPR02] C. Artigues, C. Briand, M.C. Portmann, and F. Roubellat. *Pilotage d'atelier basé sur un ordonnancement flexible. in. Méthodes du pilotage des systèmes de production sous la direction de P. Pujo et J.P. Kiefler*, pages 61–97. Hermes Science, 2002.

[ABZ88] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3) :391–401, 1988.

[All91] J.F. Allen. Time and time again : The many ways to represent time. *International Journal of Intelligent Systems*, 6(4) :341–355, 1991.

[Art97] C. Artigues. *Ordonnancement en temps réel d'ateliers avec temps de préparation des ressources.* Thèse de troisième cycle, Université Paul Sabatier, Toulouse, France, 1997.

[Bal69] E. Balas. Machine sequencing via disjunctive graphs : An implicit enumeration algorithm. *Operations Research*, 17(6), 1969.

[BCJ02] J.F. Boujut, J.B. Cavaillé, and A. Jeantet. *Instrumentation de la coopération. In. Coopération et connaissance dans les systèmes industriels sous la direction de R. Soënen et J. Perrin*, pages 91–109. Lavoisier, Hermes Science, 2002.

[Bel57] R. Bellman. *Dynamic Programming.* Princeton University Press. Princeton, 1957.

[BHHL05] C. Briand, M-J. Huguet, H.T.La, and P. Lopez. Approche par contraintes pour l'ordonnancement robuste. dans j-c.billaut, a.moukrim e.sanlaville, editeurs, flexibilité et robustesse en ordonnancement. *Traité IC2, ISBN 2-7462-1028-2*, pages 191–215, 2005.

[Bil93] 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.

[Bil99] JC. Billaut. *Recherche Opérationnelle et aide à la décision pour les problèmes d'ordonnancement.* Habilitation à diriger des travaux de recherche, Laboratoire d'Informatique, Université François Rabelais, Tours, France, 1999.

[BLE03] C. Briand, H.T. La, and J. Erschler. *Ordonnancement de problèmes à une machine : une aide à la décision pour un compromis exibilité vs performance*, pages 146–147. Dans 5ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'03), 2003.

[BM92] S. Bussmann and J. Muller. *A Negotiation Framework for Co-operating Agents*, page 117. In M, D. S., editor, Proc. CKBS-SIG, Dake Centre, University of Keele, 1992.

[BMS04] J-C. Billaut, A. Moukrim, and E. Sanlaville. *Flexibilité et robustesse en ordonnancement.* Hèrmes, 2004.

[Bén98] J. Bénassy. *La gestion de production.* Hermès, 1998.

[BS04] D. Bertsimas and M. Sim. *The price of robustness*, volume 52,no. 1, pages 35–53. Operations Research, 2004.

[Cam00] J.P. Camalot. *Aide à la décision et à la coopération en gestion du temps et des ressources.* Thèse de Doctorat, Institut National des Sciences Appliquées de Toulouse, Toulouse, France, 2000.

[CC88] J. Carlier and P. Chrétienne. *Problèmes d'ordonnancement : Modélisation, Complexité, Algorithmes.* Paris, Masson, 1988.

[CCC96] Chu-Carrol and Carberry. *Conflict Detection and Resolution in Collaborative Planning ?. In Agents Theories, Architectures, and Languages*, pages 111–126. Springer Verlarg Lectures Notes, 1996.

[CdM96] B. Chaib-draa and B. Moulin. *An Overview of Distributed Artificial Intelligence. In Foundations of Distributed Artificial Intelligence*, chapter 1, pages 3–55. G.M.P. O'Hare et N.R. Jennings eds., J. Wiley and sons, 1996.

[CP96] C. Chu and J.M. Proth. *L'Ordonnancement et ses applications.* Masson, 1996.

[DB00] A.J. Davenport and J.C. Beck. *A survey of techniques for scheduling with uncertainty.* 2000.

[Dem77] 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.

[Des05] E. Despontin. *Aide à la décision our une coopération inter-entreprise dans le cadre de la production à la commande.* Thèse de doctorat,Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, Université de Paul Sabatier, Toulouse, France, 2005.

[Dou84] G. Doumeingts. *Méthode GRAI : Méthode de conception en productique.* Thèse de doctorat, Université de Bordeaux 1, France, 1984.

[DS88] R. Davis and R.G. Smith. *Negociation as a Metaphor for Distributed Problem Solving. in Readings in Distributed Artificial Intelligence*, pages 333–356. Bond and Gasser (eds), Morgan Kauffman, 1988.

[Dup97] L. Dupont. *Gestion Industrielle : concepts et outils.* E.N.S.G.I, 1997.

[EFM85] J. Erschler, G. Fontan, and C. Merce. Un nouveau concept de dominance pour l'ordonnancement de travaux sur une machine. *RAIRO Recherche Opérationnelle/Operations Research*, 19(1), 1985.

[EFMR83] J. Erschler, G. Fontan, C. Merce, and F. Roubellat. A new dominance concept in scheduling n jobs on a single machine with ready times and due dates. *Operations Research*, 31(1) :114–127, 1983.

[EL99] P. Esquirol and P. Lopez. *L'ordonnancement.* Paris, Economica, 1999.

[Ers96] J. Erschler. *Approche par contraintes pour l'aide à la décision et à la coopération : une nouvelle logique d'utilisation des modèles formels. In. Coopération et Conception sous la direction de G. de Terssac et E. Friedberg*, pages 137–147. Octares Editions, 1996.

[Ess03] C. Esswein. *Un apport de exibilité séquentielle pour l'ordonnancement robuste*. Thèse de Doctorat, Université François Rabelais,Tours, France, 2003.

[ET88] J. Erschler and G. De Terssac. *Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production*. Rapport laas no 88137, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse, France, 1988.

[Fer95] J. Ferber. *Les systèmes multi agents : Vers une intelligence collective*. Inter-Edition, 1995.

[Fon80] G. Fontan. *Notion de dominance et son application à l'étude de certains problèmes d'ordonnancement*. Thèse de Doctorat, Université Paul Sabatier, Toulouse, France, 1980.

[Gal89] A. Le Gall. *Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'atelier*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 1989.

[Gaz03] S. H. Gazoby. *Planification des tâches dans un environnement d'ateliers à flux tirés*. Mémoire de Magister, EMP, Algérie, 2003.

[Gia88] V. Giard. *Gestion de production*. Economica, 1988.

[GLLK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. *Optimization and approximation in deterministic sequencing and scheduling : a survey*, volume 5, pages 287–326. Annals of Discrete Mathematics, 1979.

[GOT93] GOTHA. *Les problèmes d'ordonnancement*, volume 27, n°1, chapter 4, pages 77–150. R.A.I.R.O. Recherche opérationnelle/Operational Research, 1993.

[GOT02] GOThA. *Flexibilité et Robustesse en Ordonnancement*, volume 8, pages 10–12. Le bulletin de la ROADEF, 2002.

[HETL96] M.J. Huguet, J. Erschler, G. De Terssac, and N. Lompre. Negotiation based on constraints in cooperation, 1996.

[Jav97] G. Javel. *Organisation et Gestion de la production*. Masson, 1997.

[JEP+96] J.Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling in computer and manufacturing processes*. Springer Verlag, 1996.

[KPB85] G. Kallel, X. Pellet, and Z. Binder. Conduite décentralisée coordonnée d'atelier, 1985.

[Let01] A. Letouzey. *Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs*. Ecole Nationale d'Ingénieurs de Tarbes, 2001.

[LK78] J. Lenstra and A. Rinnoy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, (26) :22–35, 1978.

[LR01] P. Lopez and F. Roubellat. *Ordonnancement de la production*. Paris, Hermès, 2001.

[LSB05] H.T. La, J.L. Santamaria, and C. Briand. Une aide à la décision pour l'ordonnancement robuste en contexte mono-ressource : un compromis flexibilité/performance. *6ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'05), Tours (France)*, pages 101–116, 2005.

[LWS94] V.J. Leon, S.D. Wu, and R.H. Storer. Robustness measures and robust scheduling for job shops, 1994.

[LZ02] M.K. Lim and Z. Zhang. *Iterative multi-agent bidding and coordination based on genetic algorithm*, pages 682–689. Erfurt, Germany, 2002.

[ML93] B.L. MacCathy and J. Liu. *Addressing the gap in scheduling research : a review of optimization and heuristic methods in production scheduling*, volume vol. 31, no. 1, pages 59–79. International Journal of Production Research, 1993.

[Mon01] 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.

[OBB07] S. Ourari, C. Briand, and B.Bouzouia. Vers une approche distribuée pour l'ordonnancement réactif sous incertitudes, 2007.

[Our01] S. Ourari. *Ordonnancement temps-réel en présence d'évènements aléatoires*. Mémoire de Magister, EMP, Algérie, 2001.

[Par96] H. V. D. Parunak. *Applications of Distributed Artificial Intelligence in Industry, in. Foundations of Distributed Artificial Intelligence*, chapter 4, pages 139–164. G.M.P. O'Hare et N.R. Jennings eds., J. Wiley and sons Inc., 1996.

[PGQ98] P. Priore, D.D. Garcia, and I.F. Quesada. *Manufacturing systems scheduling through machine learning*, pages 914–917. Dans Neural Computation, NC'98,Viena, Austria, 1998.

[RS64] B. Roy and B. Sussman. *Les Problèmes d'ordonnancement avec contraintes disjonctives*. SEMA, Paris, France, 1964.

[SR95] K. Slimani and A. Ramudhin. Applications des systèmes d'aide à la décision, des systèmes à base de connaissance et de l'intelligence artificielle aux problèmes d'ordonnancement : une revue de la littérature. *Congrés International de Génie Industriel de Montréal : La productivité dans un monde sans frontière, Montréal, Canada*, 2 :405–414, 1995.

[Tho80] V. Thomas. *Aide à la décision pour l'ordonnancement d'atelier en temps réel*. Thèse de troisième cycle, Université Paul Sabatier, Toulouse, France, 1980.

[Tra01] E. Tranvouez. *IAD et ordonnancement : une approche coopérative du réordonnancement par systèmes multi-agents*. Thèse de doctorat,Université de Droit, dEconomie et des Sciences dAix-Marseille III, Faculté des Sciences et Techniques de Saint-Jérôme,France, 2001.

[Tru05] L. Trung. *Utilisation d'ordre partiel pour la caractérisation de solutions robustes en ordonnancement*. Thèse de doctorat, Laboratoire d'analyse et d'architecture des systèmes (LAAS),CNRS,Toulouse,France, 2005.

[Ver04] M. Verrons. *GeNCA : un modèle général de négociation de contrats entre agents*. Thèse de doctorat de l'Université des Sciences et Technologies de Lille, France, 2004.

[WJK99] M. Wooldridge, N. R. Jennings, and D. Kinny. A methodology for agent-oriented analysis and design. *Proc. 3rd Int. Conf. On Autonomous Agents (Agents 99), Seatle, WA*, 1999.

Annexe A

Voici ce qu'expriment les champs $\alpha/\beta/\gamma$ de la classification de Graham et al. [GLLK79] :

Valeur	Description
$\emptyset$	machine unique
P	machine parallèles identiques
Q	machine parallèles proportionnelles
R	machine parallèles non reliées
F	flow shop
J	job shop
O	open shop
FH	flow shop hybride
JG	job shop généralisé
OG	open shop généralisé
XAG avec $X \in \{P, Q, R, F, J, O\}$	problème X avec affectation générale

FIG. A.1 – Quelques valeurs du champ α_1

Valeur	Description
$prec$	il existe des contraintes de précédence générale entre les opérations
r_i	une date de début au plus tôt r_i est associée à chaque travail i
d_i	une date d'échéance préférentielle d_i est associée à chaque travail i
$\tilde{d}_i$	une date d'échéance stricte $\tilde{d}_i$ est associée à chaque travail i
$p_i = 1$	les durées opératoires sont unitaires
$pmtn$	la préemption des opérations est autorisée
$no - wait$	les opérations de chaque travail doivent se succéder sans attente
$Snsd(Rsnd)$	les ressources doivent être préparées avant et/ou après chaque exécution indépendamment de la séquence des travaux

FIG. A.2 – Quelques valeurs du champ β

Valeur	Fonction objectif à minimiser
$C_{\max}$	la durée totale (makespan)
$L_{\max}$	le plus grand retard algébrique
$T_{\max}$	le plus grand retard vrai
$\sum U_i$	le nombre de travaux en retard
$\sum [w_i]C_i$	la durée moyenne ou pondérée des travaux
$\sum [w_i]U_i$	le nombre moyen ou pondéré de travaux en retard
$\sum [w_i]T_i$	le retard moyen ou pondéré

FIG. A.3 – Quelques valeurs du champ γ

Résumé

Ce travail s'intéresse au problème job shop à plusieurs machines avec minimsation du C_{max}, noté $J_n||C_{max}$. Contrairement aux approches classiques, nous proposons dans ce travail une approche de résolution distribuée. En effet, Chaque ressource gère son propre ordonnancement local (ordonnancement à une machine), l'ordonnancement global résulte alors d'une coopération entre les diverses ressources. On suppose que les ordonnancements locaux sont des ordonnancements incorporant de la flexibilité séquentielle, cette flexibilité permettant d'une part à chaque ressource de négocier avec les autres et, d'autre part, de faire face à l'arrivée aléatoire en temps réel d'autres jobs. Pour cela, un modèle de contraintes est défini, des mécanismes de coopération entre ressources, permettant la négociation de décision d'ordonnancement, et le protocole de négociation adoptés sont décrits.

Mots clefs

Ordonnancement distribué, problème $J_n||C_{max}$, fléxibilité séquentielle, Théorème des pyramides, coopération, protocole de négociation.

ملخص

يهتم هذا العمل بمشكل الجدولة في الورشة JnIICmax بعدة ما كانات. على عكس الطرق التقليدية، الطريقة المقترحة في هذا العمل تعتمد على التوزيع. بالفعل, كل وسيلة تسير جدولتها المحلية الخاصة (جدولة بماكنة واحدة) .
نستنتج الجدولة الشاملة من تعاون بين مختلف الماكنات . نفترض أن الجدولة المحلية تتمتع بمرونة في المقاطع، هذه المرونة ستسمح من جهة لكل وسيلة للتفاوض مع الوسيلة الأخرى، و من جهة أخرى بمواجهة وصول أعمال جديدة في الزمن الواقعي.لهذا الهدف الشامل هو توافق قرارات كل وسيلة مع باقي الوسائل و احترام درجة معينة من المرونة.
يقترح هذا العمل طريقة التفاوض و اخذ القرارات بين الوسائل بالاعتماد على التوزيع.

كلمات مفتاح

جدولة بالتوزيع ، مشكل الورشة JnIICmax ، المرونة في المقاطع ، فرضية الأهرام ، التعاون ، طريقة التفاوض.

Abstract

This work deals with the Job Shop problem noted $J_n||C_{max}$. In the opposition with the classic methods, we suggest the using of a distributed sheduling. In fact, this permit to each machine to manage its own local scheduling. The global scheduling results from a process of cooperation between the different machines. We suppose that the local schedulings include a sequential flexibility. This last will permit to each machine to start a process of negiciation, and to shedule jobs that acquires in real time. For that, a model of constraints is defined and the mechanisms of coperation between resources, allowing the negotiation of decision of scheduling, are proposed.

Key words

Distributed scheduling, $J_n||C_{max}$ problem, sequential flexibility, Pyramides Theorem, cooperation, protocole of negociation.