UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences

Département de Mathématiques

MEMOIRE DE MAGISTER

Option : Recherche Opérationnelle

Intitulé : Modélisation Mathématiques pour l'aide à la décision

CONTRIBUTION A L'APPLICATION DE LA ROBUSTESSE ET DE LA FLEXIBILITE EN ORDONNANCEMENT

Par

Boukedroun Mohammed

Devant le jury composé de :

Mostafa, BLIDIA	Professeur, U. de Blida	Président
Moncef, ABBAS	Professeur, USTHB, Alger	Promoteur
Mourad, BOUDHAR	Professeur, USTHB, Alger	Examinateur
Nadia, OUKID	Maître de conférences A, U. de Blida	Examinatrice

Blida, Juin 2010

RESUME

Ce travail propose une version amélioré d'une approche d'ordonnancement robuste pour le problème à une machine 1| r_j d_j | L_{max}. Cette amélioration est fondée sur un théorème de dominance démontré dans les années quatre-vingt, par Couzinet-Mercé préalablement présenté dans le chapitre 4. Par considération de la structure d'intervalles définie par les dates de début et de fin de chaque travail, ce théorème permet de mettre en évidence un ensemble de séquences dominantes vis-à-vis du retard algébrique. La version améliorée d'ordonnancement proposée agit sur la structure d'intervalles du problème, de façon à restreindre l'ensemble dominant caractérisé grâce au théorème afin que seules les séquences optimales soient conservées. Les mécanismes mis en jeu lors des actualisations des dates de début et de fin des travaux sont décrits .ainsi que ce travail basée sur un algorithme dite de Carlier qui donne premièrement une solution et une séquence optimale de problème ses dernière sont utilisée dont la acine de l'arborescence comme un nœud pour l'évaluation et la séparation des nœuds caractérise grâce au théorème de pyramide.

Dans les chapitres 1 et 2 on a donnée des définitions et des notions utilisés dans le document en suite on a donnée quelque classification des approches d'ordonnancement robuste existe dans la littérature ainsi que la flexibilité. Par contre dans le chapitre 4 on a étudié le théorème de pyramide et leur extension utilisés, nous terminons le document par une version améliorée d'un problème d'une seule machine vis à vis de retard algébrique.

MOTS-CLES: Sommets, Pyramides, Dominance, Structure d'intervalles, robustesse, flexibilité, Ordonnancement Robuste.

ABSTRACT

These work deals with the application of two methods "robustness and flexibility" in the field of multicriteria scheduling. These approaches based on a theorem demonstrate in the eighteen of the last century years known theorem of dominance or theorem of pyramids and which is based on notions of vertex and pyramids. In this memory we stady a problem of scheduling a single machine in the stumbles and minimize the biggest delay algebraic $L_{max.}$ On account of the structure intervals defined by the start dates and end of each spot. The above theorem allow used to highlight a set of sequences dominant vis -à -vis the delay algebraic used to write a lower bound of the sets of sequences dominant and suggests an assessment of bounds exactly the terminal at best delay algebraic each node of the separation procedure and evaluation, finalement on writing an improved version of scheduling robust approach which is based on the structure of intervals of the problem using only the CARLIER optimal sequence. Moreover we propose an evaluation of terminals (the lower limit of delay algebraic) of each node of the separation procedure an evaluation.

KEYWORDS: vertex, pyramids, dominance, interval structure, robustness, flexibility, robust scheduling.

ملخص

يتناول هذا العمل تطبيق طريقتين هما طريقتا المتانة و المرونة في مجال الجدولة المتعددة المقاييس و اللتان ترتكزان على نظرية هامة جدا برهنت في مطلع الثمانينات تحت اسم نظرية الأهرامات أو نظرية السيطرة والتي ترتكز أساسا على بعض المفاهيم الأساسية منها القمة الأهرامات ..الخ. في هذا العمل المتواضع نتناول مشكل الجدولة على آلة واحدة من اجل التقليل من التأخر الجبري الذي يرمز له $L_{\max}$.

كما ,نأخذ بعين الاعتبار بنية المجالات المعرفة ب أوقات البداية و أوقات النهاية لكل عملية النظرية السابقة تسمح لنا بكل سهولة إعطاء مجموعة من المجدولات المسيطرة بالنسبة التأخر الجبري كما نعطي في هذه المذكرة حد من الأسفل لهذه المجموعة من المجدولات المسيطرة كما نقترح تقييم فعلي لهذا الحد من الأسفل.

في أخر هذه المذكرة نقترح كذالك تحسين للمجدولة المتينة و التي ترتكز أساسا على بنية المجالات وذاتك باستعمال الحل المثالي أو الأمثل ل CARLIER.

REMERCIEMENTS

Je remercie Dieu tout puissant clément et miséricordieux de m'avoir soigné et aidé.

Je tiens, avant tout, à exprimer ma profonde gratitude a monsieur **Moncef Abbas**, professeur à (USTHB) Alger qui a assumé la direction de ce travail. Qu'il veuille bien trouver ici l'expression de ma reconnaissance pour son dévouement, sa patience, sa disponibilité, ses conseils et son aide constant qu'il m'a apporté tout au long de ce travail.

Je remercie les membres de jury qui ont accepté de juger ce travail et d'y apporter leur caution :

- Monsieur **M.Blidia,** professeur à l'université Saad Dahleb Blida (USTB), qui me fait le grand honneur d'accepter la présidence du jury.
- Monsieur **M.Boudhar** professeur à (USTHB) pour l'honneur qu'il me fait en acceptant également de participer à ce jury.
- Madame **N.Oukid** maitre de conférences à l'université Saad Dahleb Blida (USTB), pour l'honneur qu'elle me fait en acceptant également de participer à ce jury.

J'adresse mes vifs remerciements à touts les enseignant, leur enseignement, leur encouragement et leur aide, ont contribués à ma formation durant toutes mes études à l'université de Blida.

Un grand exprimé à mes parents pour leurs aides morales et matérielles durant tout mon cursus universitaire.

A tous et à tout un grand merci.

DEDICACE

Je m'incline devant Dieu le tout puissant qui m'a ouvert la porte du savoir et m'a aidé à la franchir.

Je dédie ce travail à mes parents, ma mère pour ses encouragements et ses prières tout au long de mes études, mon père pour tous ce qu'il avait fait pour avoir de résultat.

Je le dédie à mes frères et sœurs et je les remercie pour les encouragements et leur s aide ainsi que ma grande famille.

- A tous mes amis sans citer les noms.
- A mes collègues de la promotion 006/007 de poste graduation.
- A tous ceux que je porte dans mon cœur.

M. Boukedroun

Table des matières

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

L'optimisation multi objectif consiste à optimiser plusieurs composantes d'un vecteur de fonction coût, contrairement à l'optimisation uni objectif, la solution d'un problème multiobjectif (PMO) n'est pas une solution unique, mais une solution de compromis qui sera choisie parmi un ensemble de solutions efficace. Un PMO peut être défini de la manière suivante (PMO) : min $F(x) = (f_1(x) , f_2(x) , \ldots, f_n (x))$, sous les contraintes x appartient à C ou $n \geq 2$ est le nombre de fonctions objectifs , $x = (x_1 , x_2 , \ldots ,x_k)$ est le vecteur représentant les variables de décision , C représente l'ensemble des solutions réalisables associés à des contraintes d'égalité , d'inégalité et des bornes explicites (espace de décision) , et $F(x) = (f_1(x) , f_2 (x) ,\ldots, f_n(x))$ est le vecteur des critères à optimiser .

Dans le cadre de l'optimisation multiobjectif, le plus souvent est que le décideur raisonne plut tôt en terme d'évolution d'une solution sur chaque critère et se place naturellement dans l'espace des critères. L'ensemble $Y = F(C)$ représente les points réalisables dans l'espace de critères (espace d'objectifs) et $Y = (y_1 , y_2 , \ldots, y_n)$ ou $Y_i = f(x_i)$ représente un point de l'espace des critères. Une relation d'ordre partiel sur cet ensemble de points appelée relation de dominance est posée. Pour la résolution des problèmes à deux critères (problèmes à petites tailles), les méthodes exactes telles que le "branch and bound" (Sen et al. 1988), l'algorithme A* (Stewart and White, 1991), et la programmation dynamique (White, 1982) sont utilisés pour la résolution, puisque elles sont efficaces. Par contre, pour les problèmes à plus de deux critères ou de grandes tailles, il n'existe pas des procédures exactes et efficaces pour les résoudre. Étant données les difficultés simultanées de la complexité NP-difficile, et le cadre multicritère des problèmes, les approches métaheuristiques sont utilisables pour la résolution des (POM), et spécialement de problèmes réels dans plusieurs domaines (télécommunications, transports, environnement,…etc.).

En effet, des méthodes heuristiques sont nécessaires pour résoudre des problèmes multiobjectifs de grandes tailles et des problèmes avec un nombre de critères supérieur à deux. Elles ne garantissent pas de trouver de manière exacte l'ensemble des solutions efficaces mais plutôt une approximation de cet ensemble. La plupart des approches heuristiques proposées sont basées sur la transformation d'un problème multiobjectif en un problème monoobjectif. Parmi ces méthodes se trouvent les méthodes d'agrégation, les méthodes $\varepsilon-$contraintes et les méthodes de programmation par but (goal programming).

Parmi les problèmes d'optimisation multiobjectif, on a le problème de l'ordonnancement, qu'on peut décomposer en trois types : l'ordonnancement de grand projets, dans les ateliers et l'ordonnancement en temps réel où la réponse au problème doit être instantanée. Le traitement de l'ordonnancement dans la littérature s'est tout d'abord orienté vers une optimisation monocritère. L'environnement manufacturier évoluant rapidement et la concurrence devenant de plus en plus acharnée, les objectifs des entreprises se sont diversifiés et le processus d'ordonnancement est devenu de plus en plus multicritère. Les critères que doivent satisfaire un ordonnancement sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement [Esquirol et Lopez 99] :

- *les objectifs liés au temps* : on trouve par exemple la minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales de réglage ou des retards par rapport aux dates de livraison.
- *les objectifs liés aux ressources* : maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches sont des objectifs de ce type.
- *les objectifs liés au coût* : ces objectifs consistent généralement a minimiser les coûts de lancement, de production, de stockage, de transport,
- les objectifs liés à une énergie ou un débit.

La préoccupation de la notion de robustesse apparaît de plus en plus fréquemment en Recherche Opérationnelle et Aide à la décision (RO-AD) ou elle donne lieu à des démarches, des concepts et des résultats de plus en plus riches et diversifiés. Le qualitatif (robuste) est appliqué à des objets

aussi variés que : une solution, un paquet de solutions, une méthode, des conclusions, ..., la signification de qualificatif (robuste) en RO-AD à savoir : une aptitude à résister à (des à peu prés) et (des zones d'ignorance). Parmi les questions qui se pose d'après B.ROY en 2004, pourquoi cette préoccupation ? Ou bien pourquoi la robustesse ? ROY montre, sur divers exemples, que c'est précisément pour se protéger de tels impacts que l'on se préoccupe de la robustesse en RO-AD. En second lieu, il propose des définitions de procédures, méthodes, conclusions, définitions qu'il croit pertinentes pour cette préoccupation de la robustesse. Enfin, il montre aussi que celles-ci conduisent à définir un ensemble V de versions et un ensemble P de procédures. Mais La recherche de la robustesse doit-elle porter sur une solution, un paquet de solutions, une méthode, des conclusions, ... ?

La robustesse doit-elle être prise en compte de façon binaire (robuste ou non robuste) ou de façon graduelle (degré de satisfaction) ou encore à l'aide de qualificatifs tels que parfaitement robuste, approximativement robuste, pseudo robuste, ... ?. Est-il souhaitable, voir nécessaire, de chercher à comparer la robustesse de solution, de paquet de solutions, de méthode, de conclusion ? Si oui, faut-il asseoir cette relation de comparaison sur un ou plusieurs critères ?

Les problèmes d'ordonnancement sont très variés par nature. On les rencontre dans de nombreux domaines comme les systèmes de production, de gestion, la logistique, l'informatique, etc. Dans ce mémoire on va faire une intégration de la flexibilité et de la robustesse dans l'étude des problèmes d'ordonnancement comme une application. Certaines informations sur la nature du problème à résoudre et sur les données à traiter sont connues, mais cette connaissance est imparfaite et plus ou moins fiable. La robustesse caractérise la performance d'un algorithme en présence d'incertitudes sur les données. On peut parler de la robustesse d'une solution, mais aussi de la robustesse d'une procédure ou de la robustesse d'une assertion. C'est un qualificatif qui se rapporte plus généralement à une aptitude à résister à l'à peu près ou à l'ignorance.

Plan du mémoire

Ce manuscrit est composé de cinq chapitres, Dans le chapitre 1, on introduit dans la section 1 le domaine de l'optimisation multicritère. Nous présentons les principes et les concepts mathématiques relatifs à l'optimisation multicritère, dans la section 2, on décrit les définitions de base de l'ordonnancement ainsi que des classifications de problèmes d'ordonnancement.

Le chapitre 2 est consacré à l'étude des notions sur la robustesse et la flexibilité et leurs applications dans le domaine de l'ordonnancement ainsi que des classifications des approches d'ordonnancement robustes.

Au chapitre trois, des notions sur les sommets, bases, structure d'intervalles, les pyramides et des relations sur l'algèbre d'ALLEN est présenté, ainsi que des notions sur les corps d'hypothèses des notions sur les ordres partiels dominants.

Pour le chapitre quatre, nous présentons l'application de la robustesse et de la flexibilité en ordonnancement. Une description de deux procédures l'une est appelée de CARLIER et l'autre de SCHRAGUE. Un théorème de dominance est présenté ainsi que leur extension. Une borne inférieure est présentée et des notions sur la séquence maître pyramide.

Pour le chapitre cinq, nous présentons une procédure de « TCOR » et une version améliorée d'un algorithme de TCOR pour les problèmes d'ordonnancement d'une seule machine dont le but est de déterminer une solution robuste et flexible vis-à-vis du plus grand retard algébrique $L_{\max}$.

CHAPITRE 1
OPTIMISATION MULTICRITERE ET PROBLEMES D'ORDONNANCEMENT

Introduction

L'optimisation combinatoire regroupe une large classe de problèmes ayant des applications dans de nombreux domaines applicatifs.

Dans la première section de ce chapitre, nous présentons tout d'abord un ensemble de définitions liées aux problèmes d'optimisation multicritère, les principes et les concepts mathématiques relatifs à l'optimisation multicritère, ensuite nous abordons les différentes approches de leur résolution. Dans la deuxième section, nous donnons des notions sur l'ordonnancement multicritère.

Optimisation multicritère:

Un problème multi objectif ou multicritère peut être défini comme un problème dont on recherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonction objectif. Généralement à plusieurs solutions car la définition d'un optimum ne peut pas être établie dans le problème multi objectifs.

1. 1 Problème d'optimisation multicritère (multi objectif):

Un problème d'optimisation combinatoire est défini par un ensemble fini de solutions discrètes D et une fonction objectif f associant à chaque solution une valeur (la plupart du temps, une valeur réelle). Ainsi, un problème d'optimisation combinatoire consiste en l'optimisation (minimisation ou maximisation) d'un certain critère sous différentes contraintes permettant de délimiter l'ensemble des solutions réalisables (ou solutions admissibles). Le PMO peut être défini comme suit:

$$(PMO)\begin{cases} optimiserF(x) = (f_1(x), f_2(x),..., f_n(x)) \\ x \in D \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

n est le nombre d'objectifs, $x = (x_1, x_2,..., x_k)$ est le vecteur représentant les variables de décision. D représente l'ensemble des solutions réalisables et chacune des fonctions f_i est à optimiser, c'est-à-dire à minimiser ou à maximiser.

A la différence de l'optimisation mono objectif, la solution d'un problème multi objectif n'est pas unique, mais c'est un ensemble de solutions non dominées.

Un problème d'optimisation recherche l'action x* telle que les contraintes soient satisfaites et qui optimise la fonction $(f_1(x^*), f_2(x^*),..., f_n(x^*))$, et l'union des domaines de définition de chaque variable et les contraintes forment l'ensemble d'actions réalisables, notons F l'ensemble des objectifs réalisables.

1.1.1 Notions de Dominance:

Soient x_1 et x_2 deux variables de D, et $f(x_1)$, $f(x_2)$ leurs performances respectives.

Définition 1.1

On dit qu'une solution $y = (y_1, y_2,..., y_k)$ domine une solution $z = (z_1, z_2,..., z_k)$ dans le cas d'une minimisation d'objectifs si et seulement si $\forall i \in [1,2,3,...,n]$ $f_i(y) \leq f_i(z)$ et $\exists i \in [1,2,3,...,n]$ telle que $f_i(y) \prec f_i(z)$.

Définition 1.2

Un sous ensemble de solution est dit dominant pour l'optimisation, s'il contient au moins un optimum pour ce critère. De manière analogue, on définit la dominance par rapport à un ensemble de contraintes lorsque le sous ensemble dominant contient au moins une solution admissible (qui satisfait

toutes les contraintes) si elle en existe. La recherche d'une solution optimale ou admissible peut ainsi être limitée à un sous ensemble dominant.

1.1.2 Notions d'efficacité [17]

Définition 1.3

Une solution $x^* \in D$ est une solution *efficace* si et seulement si il n'existe pas de $x \in D$ tel que $f^k(x) \leq f^k(x^*),\ k = 1,...,n$, avec au moins une inégalité stricte.

A partir d'un point efficace, il est impossible d'augmenter la valeur d'un des critères sans diminuer la valeur d'au moins un autre critère.
Autre définition de l'efficacité:

Définition 1.4

Une solution $x^* \in D$ est une solution *efficace* si et seulement si son vecteur de performances $f(x^*)$ est non-dominé.

Définition 1.5

Une solution est *faiblement efficace* si son vecteur de performances n'est pas strictement dominé.

1.1.3 Résolution d'un problème d'optimisation combinatoire [14]

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points particuliers suivants:

– La définition de l'ensemble des solutions réalisables,
– L'expression de l'objectif à optimiser,
– Le choix de la méthode d'optimisation à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution. Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application.

Le choix de l'objectif à optimiser ainsi que la définition de la fonction objectif mérite toute l'attention de l'analyste car rien ne sert de développer de bonnes méthodes d'optimisation si l'objectif à optimiser n'est pas bien déterminé.

Le choix de la méthode de résolution dépendra souvent de la complexité du problème, car suivant sa complexité, le problème pourra ou non être résolu de façon optimale. Si le problème est de petite taille alors une méthode de séparation et évaluation (Branch & Bound) permettant de trouver la solution optimale.

D'après l'étude bibliographique l'optimisation multicritère est très riche par les approches de résolution on trouve par exemple les approches basées sur la transformation des problèmes multicritère vers des problèmes monocritère et d'autres méthodes basées sur la notion de dominance. Parmi ces méthodes on trouve la méthode d'agrégation, la méthode lexicographique ou cette dernière classe les critères en fonction d'un ordre d'importance proposé par le décideur. Ensuite, les fonctions critères sont traitées dans cet ordre pour obtenir l'optimum. L'optimisation séquentielle des différents critères aboutit à la découverte d'une seule solution optimale. L'inconvénient majeur de cette méthode est qu'elle tend à générer des solutions qui sont largement optimisées pour certains critères et très peu pour les autres critères. Les solutions de compromis sont négligeables.

1.2 Notions préliminaires d'ordonnancement :

Cette section est consacrée à des définitions et des résultats de problème d'ordonnancement ainsi que des classes d'ordonnancement.

1.2.1 Présentation du problème d'ordonnancement: [06]

En dépit de la complexité des problèmes d'ordonnancement il existe un modèle standard sur lequel est basée la définition du problème. Dans ce qui suit, nous nous intéressons à décrire ce modèle théorique du problème d'ordonnancement.

Définition 1.6

Nombreuses sont les définitions proposées au problème d'ordonnancement, dans ces définitions on retrouve l'aspect commun de l'affectation de ressources aux tâches, en recherchant une certaine optimisation.

Parmi ces définitions on a la définition de (VAC, 2000),

Étant donnée un ensemble de tâches à accomplir, le problème d'ordonnancement consiste à déterminer quelles opérations doivent être exécutées, et à attribues des dates et des ressources à ces opérations de façon à ce que les tâches soient, dans la mesure du possible, accomplies en temps utile, au moindre coût et dans les meilleures conditions. (VAC, 2000).

Résoudre un problème d'ordonnancement consiste à ordonnancer c.à.d. programmer ou planifier dans le temps, l'exécution des taches en leur attribuant les ressources nécessaires ,matérielles ou humaines de manière à satisfaire un ou plusieurs critères préalablement définies, tout en respectant les contraintes de réalisation . (LOP et Esq. 1999). D'une manière générale il y a problèmes d'ordonnancement quand il existe (JEF et al ,2006).

- Un ensemble de travaux ou de tâches ou encore jobs ou lots à réaliser ;
- Un ensemble de ressources à utiliser par ces travaux ;

- Un programme à déterminer pour allouer convenablement les ressources aux tâches.

Le problème d'ordonnancement consiste alors, à programmer dans le temps l'exécution de certaines tâches, en leur attribuons les ressources, et en respectant bien les contraintes imposées afin d'atteindre certains objectifs, pour la notion d'ordonnancement, on peut envisager sa définition en deux types :
Comme une fonction de la gestion de production ou comme une solution réalisable aux problèmes d'ordonnancement définie précédent.

1.2.2 Les éléments du problème d'ordonnancement : [06]

D'après (Hen, 1999), les problèmes d'ordonnancement ayant quatre éléments fondamentaux interviennent : les tâches, les ressources, les contraintes et les objectifs, donc la formulation et la description de problème se fait par la détermination de ces quatre éléments de base.

- Les tâches

Définition 1.7

D'après (Gia.1988), une tâche est un travail mobilisant des ressources et réalisant un progrès significatif dans l'état d'avancement du projet (Gia.1988).

Plus précisément c'est une entité élémentaire de travail localisé dans le temps par une date de début t_i ou de fin c_i dont la réalisation nécessite une durée p_i telle que $p_i = c_i - t_i$ (LopetEsq.1999).

De façon générale, (T'Ki et Bil.2006) détermine trois paramètres caractérisant une tâche :

- La durée opératoire p_i, c'est la durée d'exécution de la tâche. (processing time).
- La date de disponibilité r_i (release time), c'est la date de début au plus tôt.
- La date d'échue d_i (due date), c'est la date de fin au plus tard.

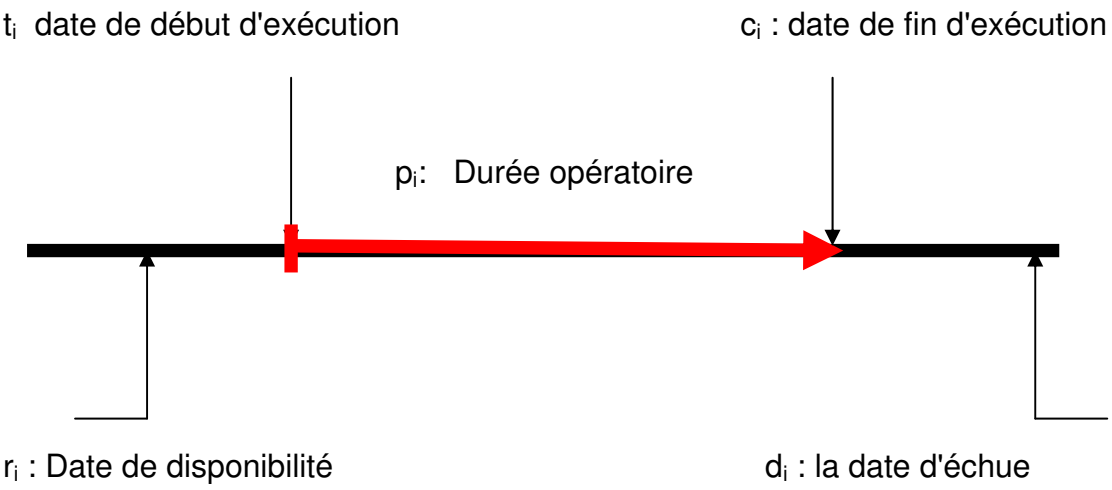


Figure 1 : Caractéristique d'une tâche

Remarque 1.2

On peut donner une relation entre ces paramètres :

$$r_i \le c_i = p_i + t_i \le d_i \le \tilde{d}_i \quad \text{.....(2)}$$

Avec

$\tilde{d}_i$: Le temps mort de la tâche ;

c_i : (completion time) la date d'achèvement de la dernière opération de la tâche i ;

t_i : la date de début d'exécution de la tache i.

Où w_j c'est le poids de la tâche. La réalisation de la tâche i nécessite l'utilisation de ressources.

- Les ressources

Définition 1.8

D'après (Lop et Esq.1999).

Une ressource est un moyen destiné à être utilisé pour la réalisation d'une tâche, et disponible en quantité limitées.

Les ressources peuvent être des machines, des hommes, des ouvriers, des équipements des budgets .... Les ressources sont classifiées en :

- Ressources renouvelable : si après avoir été utilisée par une ou plusieurs tâches elle est à nouveau disponible.

- Ressources disjonctives : si elle ne peut être affecté qu'à une seule tâche à la fois.

- Les contraintes

Définition 1.9

Les contraintes représentent les limites imposées par l'environnement, tandis que l'objectif est le critère d'optimisation, et on a plusieurs types :

- Les contraintes de localisation temporelle ;
- Les contraintes d'enchaînement ;
- Les contraintes de ressources.

- Les objectifs

Chaque ordonnancement est guidé par un critère et on a l'ordonnancement monocritère, ou bien plusieurs critères et on a l'ordonnancement multicritère. (Lope tr Esq.1999) distinguent plusieurs types d'objectifs citons par exemple:

- Les objectifs liés au temps : par exemple la minimisation de plus grand retard algébrique ;
- Les objectifs liés aux ressources: par exemple maximisé la charge d'une ressource ;
- Les objectifs liés au coût par exemple minimisé le coût de lancement de transport ;

- Les critères d'optimisation

Pour chaque problème d'ordonnancement on peut choisir deux types d'objectifs, ou bien trouver la meilleure solution c.à.d. l'optimalité des solutions ou bien trouver une solution c.à.d. l'admissibilité. Pour la recherche d'une solution qui impose à mesurer la qualité d'une solution et dans ce cas là [Hen.99] distingue trois catégories importantes des critères d'optimisation:

- Les critères liés aux dates de fin (C_{max}, $\sum C_i$, L_{max} , T_{max} , ...) ;
- Les critères liés au volume;

- Les critères liés à l'utilisation des ressources.

On définit le retard algébrique (Lateness) noté L_i qui détermine le retard d'exécution de la tâche i par rapport à sa date d'échéance. $L_i = c_i - d_i$.
Le retard absolu de la tâche i noté $T_i = \max(L_i, 0)$.

- Relations entre les critères [34]

Dans la littérature le traitement de l'ordonnancement c'est tout d'abord orienté vers une optimisation monocritère, mais les objectifs des entreprise se sont diversifiés leur processus d'ordonnancement devenu de plus multicritères.

Il existe plusieurs réductions polynomiales entre ces critères qui nous permettent de déduire certains des autres.

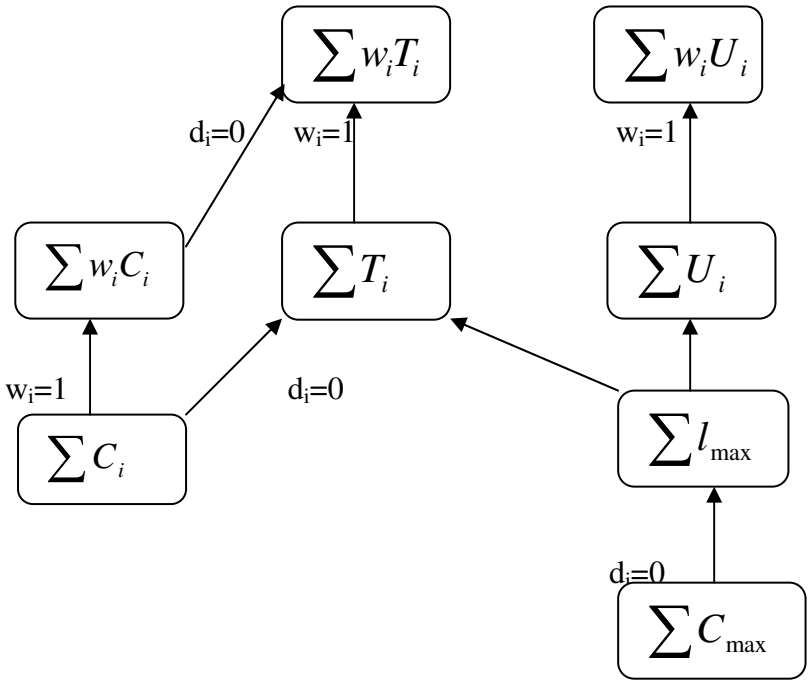


Figure 2 : Relations entre les critères d'optimisation (Pin, 1995) (T'Ki et Bil,2006).

1.2.3 Ordonnancement de groupe [14], [15]

La méthode des ordonnancements de groupes d'opérations permutables, permet d'introduire une flexibilité séquentielle importante tout en garantissant une certaine qualité. Ceci fournit une grande liberté l'ors de

l'exécution de l'ordonnancement, ce qui permet à l'humain d'effectuer des choix sans altérer les performances.

Un groupe d'opérations permutables est un ensemble d'opérations qui seront exécutées successivement sur une même machine, dans un ordre qui n'est pas fixé à l'avance. Un ordonnancement de groupe est défini par une séquence de groupe sur chaque machine. Il est dit faisable si toute permutation des opérations au sein de chaque groupe conduit à un ordonnancement qui satisfait les contraintes du problème. Un ordonnancement de groupe définit ainsi plusieurs ordonnancements réalisables de manière implicite.

Cette représentation implicite d'un ensemble d'ordonnancement possède une propriété très intéressante : le calcul de la qualité de l'ordonnancement dans le pire des cas est calculable en temps polynomial pour les critères de type minmax comme (C_{max}, L_{max} et T_{max}). Ainsi, cette méthode est utilisable pour les problèmes de tailles conséquentes.

Pourquoi l'ordonnancement de groupes est-il intéressant ? [20]
L'ordonnancement de groupes est intéressant puisque :

- La Méthode est prédictive réactive ;
- Flexibilité sur les séquences ;
- La Méthode est bien étudiée durant les 30 dernières années par : (Erschler and Roubellat, 1989, Billaut and Roubellat, 1996, Wu et al. 1999, Artigues et al. 2005) ;
- Méthode permettant de pallier certaines incertitudes : (Wu et al. 1999, Esswein, 2003, Pinot et al. 2007) ;
- évaluation de l'ordonnancement dans le pire des cas en temps polynomial pour les objectifs de type minmax comme le L_{max}.
- Une évaluation du meilleur des cas d'un ordonnancement de groupes pourrait également être utile.

1.2.4 Classification des problèmes d'ordonnancement : [34]

Selon la nature des variables mises en jeu, la nature des contraintes, plusieurs classifications des problèmes d'ordonnancement est proposée, MacCathy & Liu proposent sept types de problèmes d'ordonnancement d'atelier sont ainsi distingués :

- 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 ;
- 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.

Les problèmes d'atelier sont représentés selon une classification (de GRAHAM) à trois champs α /β /γ telle que :

- Le champ α d'écrit l'environnement de la machine ;
- Le champ β d'écrit la tâche et les caractéristiques de la ressource ;
- Le champ γ signifie le critère à utiliser.

CHAPITRE 2
ROBUSTESSE ET FLEXIBILITE EN ORDONNANCEMENT

Introduction

Tous les analystes qui ont traité des problèmes de décision savent que les valeurs numériques utilisées dans les modèles sont discutables. Ces valeurs résultent d'hypothèses concernant le contexte du problème, d'estimations de variables aléatoires ou bien mal connues, de prévisions d'évènements futurs, de choix plus ou moins arbitraires concernant les valeurs de certains paramètres,... Cette incertitude inévitable autour des composants de chaque modèle doit être prise en compte dans le processus d'aide à la décision. A côté des outils traditionnels comme les approches stochastiques, la théories des possibilités ou la logique floue, le concept de robustesse a été introduit pour exprimer que le rôle de l'analyste est de fournir au décideur des solutions, des autorisations ou simplement des informations qui sont valides pour tous ensembles de valeurs considérées et ainsi pour représenter le problème de décision.

L'aide à la décision est un domaine qui provient de la "recherche opérationnelle" dans le domaine militaire. La deuxième guerre Mondiale a stimulé de nombreuses avancées dans les approches de formulation de problèmes et de recherche de solutions les mieux adaptées à des situations données.

2.1 Les étapes et les approches d'aide à la décision [07]

L'aide à la décision est:

L'activité de celui qui, prenant appui sur des modèles clairement explicités mais non nécessairement complètement formalisés, aide à obtenir des éléments de réponse aux questions que se pose un intervenant dans un processus de décision, éléments concourant à éclairer la décision et normalement à prescrire, ou simplement à favoriser, un comportement de

nature à accroître la cohérence entre l'évolution du processus d'une part, les objectifs et le système de valeurs au service des quels cet intervenant se trouve placé d'autre part.

Les approches d'aide à la décision

Les approches d'aide à la décision peuvent se subdiviser selon le type de solution que l'on cherche :

- Une solution optimale, considérant un seul critère et un seul scénario ;
- Une solution de compromis, où l'on a plusieurs critères et un seul scénario ;
- Une solution de compromis robuste, où l'on a plusieurs critères et plusieurs scénarios.

Les étapes de l'aide à la décision

La démarche scientifique d'aide à la décision se compose de quatre étapes principales comme l'indique le schéma suivant :

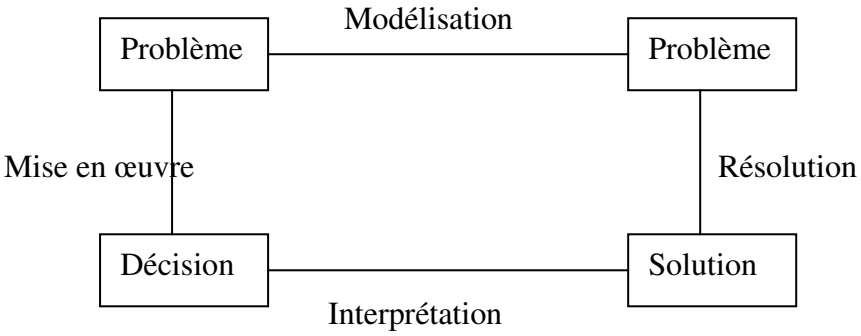


Figure 3 : Démarche scientifique de l'aide à la décision.

La modélisation

Analyse du problème, repérage du champ de la décision, des acteurs et entités concernés, fréquence et horizon de la décision.

- Compréhension du système de valeurs, des objectifs et préférences du décideur ;
- Choix d'une problématique décisionnelle ;

- Traduction des décisions dans le langage formel choisi, définition des variables de décision ;

- Définitions des paramètres, de leur domaine de variation et de leur degré d'incertitude ;

- Enoncé et formalisation des contraintes délimitant le domaine des décisions ;

- Formalisation du ou des critères.

La résolution

Choix d'une démarche algorithmique exacte ou heuristique.

- Mise en œuvre de la procédure algorithmique, recours à des logiciels spécifiques, modeleurs, tableurs ;

- Construction d'un résultat selon la problématique choisie ;

- Analyse de sensibilité ou de robustesse du résultat en fonction de variations des paramètres.

L'interprétation

- Elaboration d'une recommandation sur la base des résultats précédents.

- Présentation de la recommandation aux acteurs.

2.2 L'incertitude :

Roy (1997) a été le premier à introduire le concept de robustesse en recherche opérationnelle et en aide à la décision. Il cite aussi Kouvelis et Yu (1997) qui, dans le contexte de l'optimisation de paramètres discrétisés, donnent des résultats théoriques et des algorithmes pour déterminer une solution montrant le plus faible écart par rapport à celle obtenues dans des conditions optimales.

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 œuvre (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 façon plus ou moins prévisible.

Davenport & Beck proposent Quelques types de perturbations :

Paramètres internes:

- Variation de la capacité d'une ressource ;
- Variation de la durée opératoire p_j d'une tâche ;
- Ajout ou suppression d'une contrainte ;
- Modification de la fonction objective.

Paramètres externes :

- Modification des r_j ou d_j d'une tâche ;
- Insertion ou annulation d'une tâche ;

[Mackay et al. 89] distinguent, trois catégories d'incertitudes :

- les *incertitudes complètement inconnues*, correspondant à des événements totalement imprévisibles, par exemple une grève, une catastrophe naturelle ;

- les *suspicions du futur*, qui sont issues de l'intuition et de l'expérience des décideurs, mais difficilement modélisables.

- Les *incertitudes connues*, ou partiellement connues, pour lesquelles un modèle d'information est disponible ou la construction de l'ordonnancement, ne peuvent être traitée que de façon implicite.

Dans les deux premiers cas. Billaut et al. 2005 distinguent quatre types de modèles :

Les modèles stochastiques, les modèles flous, les modèles par intervalles et les modèles par scénarios.

2.3 L'incertitude en ordonnancement

Esswein, 2003 dits que Les incertitudes décrivent les modifications possibles des données entre la phase prédictive et la phase réactive d'un ordonnancement.

Exemples 2.1 :

Le retard d'une opération ; l'insertion d'une opération ; la distance entre le modèle et la réalité (par exemple, les temps de transport entre deux machines ne sont pas pris en compte dans le modèle) ; la panne d'une machine.

2.4 Les aléas

Définition 2.1 :

Les aléas représentent les incertitudes causés par des événements extérieurs (et donc aléatoires). Donc, Les aléas sont un sous-ensemble des incertitudes.

Exemples 2.2 :

- Les modifications structurelles des données sont des aléas.
- Les valeurs incertaines ne sont pas des aléas.
- L'insertion d'une opération, la panne d'une machine sont forcément des aléas.

2.5 Robustesse

2.5.1 Définitions

Il n'existe pas une définition unique de robustesse, mais il est probablement nécessaire à classer les types de problèmes de la décision et les types d'incertitudes avant de proposer des genres différents de robustesse qui pourrait être opérationnelle. On peut distinguer essentiellement quatre concepts qui pourraient être des points de départ.

Décision robuste

D'après (Gupta et Rosenhead, 1972)

Le concept de décision robuste pourrait aussi être appelée flexibilité dans le sens qu'une décision à un temps donné est robuste si elle reste considérée encore 'bonne' dans le futur. [07]

Solution robuste

D'après (Kouvelis et Yu, 1997)

Le concept de solution robuste où moyens robustes « bonne dans toutes ou la plupart des versions » est une version (i.e. scénario) qui a un ensemble de valeurs valides dans le modèle pour les données. [07]

Conclusion robuste

D'après (Roy, 1998)

Le concept de conclusion robuste où moyens robustes `` valide dans tous ou la plupart des versions '', est une version qui est un ensemble acceptable de valeurs pour les paramètres du modèle. [07]

Méthode robuste

D'après (Vincke, 99 un, b, Sorensen, 01)

Le concept de méthode robuste où moyens robustes `` lequel donne des résultats valide dans tous ou la plupart des versions '', est une version qui est un ensemble possible de valeurs pour les données du problème et pour les paramètres de la méthode. [07]

Définition 2.3

D'après (Sevaux & Sörensen 04)

La robustesse de solution apparaît quand une solution ne change pas beaucoup lorsqu'il y a des petits changements sur les données du problème, c'est-à-dire la méthode de résolution est capable de trouver une solution proche de la solution actuellement utilisée. [07]

Remarque 2.1

Il n'existe aucune contradiction entre ces définitions: ils illustrent seulement le fait que les genres différents de robustesse devraient être introduits dans l'aide à la décision.

Il est très important d'éviter toute confusion entre robustesse et la propriété de la stabilité traditionnelle associée à l'analyse de la sensibilité.

2.5.2 La dimension subjective de robustesse

Le fait qu'une décision (solution, conclusion) peut être considéré comme robuste, le décideur est prêt à concéder dans l'information.

2.5.3 La Robustesse en ordonnancement

Le terme robustesse apparaît de plus en plus souvent en ordonnancement, mais aussi en aide à la décision et en recherche opérationnelle.

Définition 2.4

Un des souhaits souvent formulés par les décideurs vis-à-vis de la méthode multicritère qu'ils utilisent est d'avoir une idée de la robustesse du résultat. C'est la raison pour laquelle l'étude de la sensibilité des méthodes multi- critères fait partie des grands axes de recherche du domaine. Cette demande traduit la volonté de savoir dans quelle mesure une variation des données due par exemple à une erreur de mesure ou d'estimation risque d'affecter le résultat donné par la méthode. [15]

Dans la littérature, plusieurs définitions sont données par l'ordonnancement robuste, nous citons :

Définition 2.5

D'après (Jensen 2001)

Un ordonnancement est dit robuste quand on prévoit sa bonne performance par rapport aux autres face à un certain ensemble de scénarios, et quand on utilise la technique de décalage à droite pour réordonnancer. [15]

Définition 2.6

D'après (Esswein 2003)

Un ordonnancement robuste est un ordonnancement dont les performances sont acceptables en présence d'incertitudes. [15]

Définition 2.7

D'après (Kouvelis et al. 2000)

Un ordonnancement robuste est un ordonnancement dont la performance (par rapport à l'ordonnancement optimal associé) est relativement insensible aux réalisations potentielles des durées opératoires. [15]

Définition 2.8

D'après (Davenport & Beck 2000)

Un ordonnancement robuste est un ordonnancement capable d'absorber certaine quantité d'événements inattendus sans avoir à être réordonnancé. [15]

Définition 2.9

D'après (Daniels & Kouvelis 1995)

Un ordonnancement robuste est un ordonnancement ayant la meilleure performance au pire de cas par rapport à la solution optimale correspondante sur toutes réalisations potentielles des durées opératoires.[15]

Définition 2.10

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. [15]

2.5.4 Les difficultés de l'analyse de robustesse [21]

D'après Roy et al, Il faut bien reconnaître qu'une analyse de robustesse peut s'avérer lourde et coûteuse. C'est pourquoi les auteurs

renoncent souvent. Pourtant, ce n'est qu'au prix de ce genre d'analyse qu'il est possible d'aider à construire, transformer ou justifier des préférences et d'entamer une discussion critique pour asseoir une recommandation.
Il est clair que la rigueur d'une conclusion robuste décroît selon qu'il s'agit d'une conclusion parfaitement, approximativement ou seulement pseudo robuste. Mais si on ne se limite pas aux conclusions parfaitement robustes c'est tout d'abord parce qu'elles sont beaucoup plus difficiles à établir que les autres. Ensuite, l'ouverture aux exceptions et, a fortiori, à des énoncés moins formels peut permettre des conclusions plus riches dans leur contenu.

2.5.5 Mesure de la robustesse [21]

Il est difficile de définir une mesure de robustesse puisque cette mesure doit pouvoir tenir compte des différents aléas et incertitudes qui peuvent modifier les prévisions.

Il est important de pouvoir mesurer la robustesse afin de pouvoir déterminer quelle solution est plus robuste qu'une autre, ce qui est utile dans un contexte d'optimisation.

Les mesures diffèrent selon les spécificités du domaine applicatif considéré et selon que l'on s'intéresse à l'insensibilité aux incertitudes de la fonction objectif (quality robustness) ou à l'insensibilité d'une solution d'ordonnancement particulière (solution robustness).

Bertsimas & Sim 2004, Esswein 2003, Briand et al. 2003, 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 vise versa.

2.6 Flexibilité

Le terme flexibilité est très utilisé, notamment en gestion de production, où la recherche de flexibilité apparaît comme un des éléments les plus marquants dans l'évolution des systèmes de production.
Pour générer des ordonnancements robustes, on peut utiliser des méthodes de flexibilité.

2.6.1 Définitions : parmi les définitions existes dans la littérature nous citons

Définition 2.11

La flexibilité c'est la liberté dont on dispose durant la phase de mise en œuvre d'un ordonnancement. [15]

Définition 2.12

La flexibilité d'un objet, d'un système, est une propriété qui recouvre généralement deux aspects complémentaires mais distincts :

- un aspect interne lié à une capacité de changement, de déformation, ou de modification, à une variété d'états possibles.
- un aspect externe lié à une capacité d'adaptation à des modifications de l'environnement, à des perturbations. [15]

Remarque 2.2

Le premier type de flexibilité est lié à la nature même de l'objet. Mais Le second concerne les interactions avec son environnement, sa fonction au sein d'un ensemble plus vaste. Si la capacité de changement peut être mesurée indépendamment de l'environnement de l'objet, sa capacité d'adaptation dépend du type de perturbations qu'il doit être capable d'absorber. On voit donc bien ici que la notion de flexibilité peut être reliée à celle de robustesse.

Remarque 2.3

Notons que les auteurs utilisent respectivement les notions de flexibilité statique et flexibilité dynamique pour désigner ces flexibilités interne et externe respectivement.

2.6.2 Flexibilité en ordonnancement [21]

Dans (GOThA 2002) et pour le domaine de l'ordonnancement, 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 Billaut et al distinguent quatre types de flexibilités.

La flexibilité temporelle

La flexibilité temporelle, concernant les dates de début des tâches, et autorisant sous certaines limites une dérive de ces dernières dans le temps. Autrement dit les dates de début des activités ne sont pas fixées définitivement.

Remarque:2.4

Dès qu'un ordonnancement est déterminé, il est possible de construire un graphe potentiel tâches conjonctif (satisfaisant les contraintes de ressource), où le respect des performances temporelles (valeurs de C_{max} ou L_{max}) est imposé grâce à l'ajout d'arcs de longueur négative, entre certains nœuds et le nœud origine 0.

Cette forme de flexibilité est donc naturellement présente dans une solution, et peut éventuellement être augmentée dans un objectif de robustesse.

La flexibilité séquentielle

Elle autorise une modification de l'ordre des tâches sur les ressources. Autrement dit que les séquences des activités sur les ressources ne sont pas fixées définitivement, elle est plus rarement utilisée.

L'introduction de flexibilité séquentielle impose de manipuler, non pas un seul ordonnancement, mais une famille d'ordonnancement. De ce fait, il est plus difficile, en évitant la combinatoire liée à une énumération exhaustive et systématique des solutions, de déterminer la performance d'une famille d'ordonnancements, ces derniers ne possédant pas tous une performance identique.

Remarque 2.5

Notons que l'introduction de flexibilité séquentielle induit obligatoirement une augmentation de la flexibilité temporelle (l'inverse étant faux).

La flexibilité sur les affectations

La flexibilité sur les affectations laisse libre certains choix d'affectation. Autrement dit que les ressources sur lesquelles les activités s'exécutent ne sont pas déterminées de façon définitive.

La flexibilité sur les affectations, elle existe initialement dans le problème, et rarement conserver à l'issue de l'ordonnancement hors ligne car dans le cas de la flexibilité séquentielle, se pose également ici la difficulté de manipulation d'une famille d'ordonnancements et de son évaluation vis-à-vis de la fonction objective.

La flexibilité sur les affectations est souvent utilisée au cours de la phase d'ordonnancement réactif lorsque, par exemple suite à la défection d'une ressource, une tâche est déplacée de la ressource défectueuse vers une autre de façon dynamique. La position d'insertion est souvent déterminée par analyse de la flexibilité temporelle disponible sur la ressource de destination, afin de minimiser les conséquences sur la performance.

Flexibilité sur les modes d'exécution

Dans ce type de flexibilité le mode d’exécution des activités n’est pas définitivement déterminé.

Remarque 2.6

Ce dernier type de flexibilité implique la flexibilité séquentielle et temporelle et éventuellement la flexibilité sur les affectations.

Remarque 2.7

Les quatre formes de flexibilité précédemment décrites sont de type statique. La flexibilité dynamique d'un ordonnancement réside dans la capacité de l'ordonnancement réactif à adapter l'ordonnancement déterminé hors ligne, en présence de perturbations, de façon à satisfaire au mieux les exigences propres à ce niveau de décision.

2.6.3 Mesure de la flexibilité

La mesure de la flexibilité dépend généralement de :

- du type de flexibilité à mesurer ;
- du problème traité ;
- de la matière d'utiliser la flexibilité ;
- de la méthode utiliser pour générer la flexibilité.

Le but de la flexibilité est d'améliorer la robustesse de l'ordonnancement. On peut utiliser la mesure de la robustesse comme évaluation de la flexibilité.

2.6.4 Classification des approches d'ordonnancements robuste : [06]

Davenport and Beck présentent une classification et qui constitue une référence souvent utilisée dans l'ordonnancement robuste, cette classification distingue trois approches :

- Approche proactives ;
- Approche réactives ;
- Approche proactives-réactives.

2.6.4.1 Les approches proactives

Les approches proactives considère le problème de façon prévisionnelle (hors-ligne) , mais tout en tenant compte des aléas et incertitudes qui peuvent survenir pendant l'exécution de l'ordonnancement .en cela , les méthodes proactives sont également appelés méthodes robustes , car leur but n'est pas de déterminer une solution optimale par apport à un critère usuel ($L_{\max}$ …..) mais d'essayer de garantir un comportement « correct » de la solution proposée si des événements imprévus surviennent.

Différents types de générations d'ordonnancements proactives peuvent être distingués. [12], [13], [14], [15].

- La redondance temporelle ou de ressources ;
- Les techniques probabilistes ;
- Le contingent schelling.

Il peut toujours survenir des perturbations non anticipées lors de la phase proactive, donc il est toujours possible que lors de sa mise en œuvre, les choses ne se passent pas exactement comme elle devrait l'être.

La redondance temporelle

Le problème de l'utilisation de ces méthodes est la détermination des temps morts à insérer et de leur devenir lorsque les perturbations anticipées ne surviennent finalement pas.

Par exemple pour l'évaluation de risque d'un projet, En effet, ordonnancer au plus tôt l'activité réduit le risque de retard mais accroît le coût actualisé du projet.

Exemple 2.2

Pour pouvoir planifier un projet avec un maximum de précision, Goldratt 1997 développe la méthode de la chaîne critique qu'est une méthode proposée dans la gestion de ces aléas. Cette méthode consiste à estimer chaque job avec une durée au plus juste.

2.6.4.2 Les approches réactives

Une approche réactive est une approche qui consiste à trouver une méthode qui doit souvent être rapide pour retrouver une solution réalisable après la survenue d'un aléa.

Remarque 2.8

La rapidité d'exécution n'est pas le seul facteur à considérer, car il est souhaitable d'éviter de trop dévier de la solution initiale.

Exemple 2.3

Perturber trop les dates de début des activités implique de diffuser les modélisations dans toutes les cellules de la chaîne logistique liée.

2.6.4.3 Les approches proactives-réactives

Certains auteurs proposent d'utiliser des méthodes proactives-réactives pour assurer une bonne réactivité du système face aux

événements imprévus ; ces types de méthodes ont une particularité d'être composées de deux méthodes associées, l'une proactive et l'autre réactive.

Il n'existe pas plusieurs méthodes de ce type dans la littérature, la proactivité désignant la volonté d'anticiper les perturbations avant qu'elles ne se produisent, nous distinguons trois types d'approches selon que :

- L'anticipation a lieu lors de la phase hors-ligne et consiste à construire une solution robuste ;
- L'anticipation a lieu lors de la phase hors-ligne et vise à construire une famille flexible de solutions ;

L'anticipation a lieu lors de la phase d'adaptation en ligne d'un ordonnancement de référence.

Exemple 2.4

Approche Orabaid (Ordonnancement d'Atelier Basé sur l'Aide à la Décision) (BILLAUT 1996), la flexibilité séquentielle dans la phase prédictive est proposé un système d'aide à la décision dans la phase réactive qui s'appuyant également sur la flexibilité séquentielle.

Remarque 2.8

Dans la littérature Mehta et Uzsoy présentent une autre classification des ordonnancement robuste cette dernière comporte quatre catégories l'approche totalement réactive, l'approche prédictive - réactive, l'approche robuste et l'approche à base de connaissances, telle que la catégorie des approches à base de connaissance correspond aux méthodes dont l'objectif est de fournir un mécanisme pour la sélection dynamique d'une politique de ré-ordonnancement appropriée, parmi un ensemble d'alternatives possibles.
Donc l'approches à base de connaissance est une sous-catégorie particulière des approches réactives.

D'autre part Herroelen et Leus présentent une autre classification des ordonnancements robuste, les approches réactives, les approches stochastiques, les réseaux de projets stochastiques, les approches floues, les approches proactives et les approches basées sur l'analyse de sensibilité.

CHAPITRE 3
STRUCTURE D'INTERVALLE POUR LA CARACTERISATION DES SOLUTIONS ROBUSTES ET FLEXIBLES EN ORDONNANCEMENT

Introduction :

Dans ce chapitre nous proposons des notions nécessaires pour la recherche d'un ordonnancement robuste, commençant par la définition d'un ordre partiel ensuite une structure d'intervalle ainsi que des rappels sur l'algèbre d'Allen, finalement nous introduisons un théorème démontré dans les années quatre vingt nommé théorème de dominance (théorème de pyramide), qui nous donne des séquences dominantes et terminons par des résultats, des extensions et des remarques sur ce théorème.

3.1 Ordre partiel

Définition 3.1

Nous donnons ici une définition de la notion d'ordre partiel, telle qu'elle est présentée dans (Cheng et al. 2002), ainsi qu'un rappel de quelques propriétés de ces ordres partiels.

Un *ordre partiel* p est caractérisé par une paire $p = (X, \leq_p)$, X étant un ensemble d'éléments, où toute relation $\leq_p$ sur $X \times X$ est *réflexive, antisymétrique et transitive*. Pour $u, v \in X$, $u \leq_p v$ est interprétée comme u est plus petit ou égal à v. Un élément v est minimal (respectivement maximal) dans p s'il n'existe aucun élément u tel que ($u \leq_p v$) respectivement ($v \leq_p u$).

Un ordre partiel $p = (X , \leq_p)$ est un *ordre total* (aussi appelé ordre linéaire ou ordre complet) si pour tout couple $(u , v) \in X \times X$, $u \leq_p v$ ou $v \leq_p u$ est vérifié.

Étant donné deux ordres partiels $p = (X , \leq_p)$ et $Q = (X , \leq_Q)$ sur le même ensemble X , Q est une extension de p ou p est un sous ordre de Q . Si $u <_p v$ implique $u <_Q v$ pour tout couples (u.v.) de X.

Définition 3.2

L'intersection de deux ordres partiels P_1 et P_2 sur le même ensemble X est l'ordre partiel $P_1 \cap P_2 = (X, \leq_{P1 \cap P2})$, $u \leq_{P1 \cap P2} v$ si et seulement si $u \leq_{P1} v$ et $u \leq_{P2} v$ pour tout u, v de X.

3.2 Le but des ordres partiels en ordonnancement : [11], [12], [13]

Les ordres partiels permettent de caractériser des ensembles fléxibles de solutions, en évitant l'explosion combinatoire liée à leur énumération.

Un ordre partiel présente parfois une structure particulière qui favorise le calcul de performance au mieux ou au pire de l'ensemble de solutions qu'il caractérise. Il devient ainsi possible de comparer deux ordres partiels en fonction de leur performance.

Une autre propriété intéressante liée à la structure d'un ordre partiel réside dans la possibilité de calculer rapidement le nombre de solutions caractérisées.

Exemple 3.1

À titre d'exemple, le concept de groupe de tâches définit précédemment est bien un ordre partiel qui possède les deux propriétés précédentes. Rappelons que ce concept permet de caractériser une famille de solutions en associant à chaque ressource une séquence de groupes. Les groupes sont ordonnés mais les tâches de chaque groupe sont totalement permutables. [12]

3.3 Concept de corps d'hypothèses

Fontan dit que lorsque l'on recherche des conditions nécessaires, suffisantes ou dominantes d'admissibilité ou d'optimalité, il est intéressant de préciser le corps d'hypothèses que l'on considère, c'est-à-dire l'ensemble des informations sur les données et leurs relations que l'on suppose disponibles pour la définition des conditions.

Le corps d'hypothèses considéré est alors dit complet.

D'autres conditions ne considèrent pas les valeurs explicites des paramètres, mais plutôt les relations d'ordre existantes entre celles-ci. On parle alors de corps d'hypothèses restreint ou partiel.

Six corps d'hypothèses principaux sont distingués par Fonton, ils sont présentés dans le tableau suivant :

corps d'hypothèses	dates de début et de fin	durées opératoires
CH0	*connues*	*Connues*
CH1	ordre relatif des dates	Inconnues
CH2	ordre relatif des intervalles	Inconnues
CH3	connues	Inconnues
CH4	ordre relatif des dates	ordre relatif
CH5	connues	ordre relatif

Tableau 1 : Les six principaux corps d'hypothèses

3.4 Ordre partiel nécessaire, suffisant et dominant en Ordonnancement [13]

En ordonnancement, les notions de dominance et de condition nécessaire et suffisante, relatives soit à l'admissibilité d'un problème, soit à l'optimisation d'un critère, se traduisent souvent par la définition d'ordres partiels.

- Ordre partiel nécessaire :

Les conditions nécessaires d'admissibilité permettent classiquement d'imposer des ordres partiels nécessaires entre les tâches. Hormis les relations d'ordre entre deux tâches ($i \prec j$) on distingue trois types d'ordre partiel :

- Précédences obligatoires entre une tâche i et un ensemble de tâches S: $i \prec S$ ou $S \prec i$.
- Précédences interdites entre une tâche i et un ensemble de tâches S: $i \not\prec S$ ou $S \not\prec i$.
- Non insérabilité d'une tâche i dans un ensemble de tâches S : $i \times S$.

- Ordre partiel Suffisant :

L'utilisation de conditions suffisantes d'admissibilité ou d'optimalité en ordonnancement est généralement limitée à des problèmes d'ordonnancement simples.

Il n'existe pas de condition suffisante d'admissibilité pouvant s'appliquer de façon générique à des problèmes quelconques.
Les conditions suffisantes d'optimalité variant en fonction du critère considéré et de la nature du problème (flow shop, job shop, open shop).

Exemple : règle de Johnson (Johnson 1954).

- Ordre partiel dominant :

Rappelons qu'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.

Pour un problème d'optimisation, si on utilise la notion d'ordre partiel, on dit qu'un ordre partiel P sur un ensemble de tâches d'un problème d'ordonnancement est un ordre partiel dominant s'il existe une séquence

optimale qui est une extension linéaire de cet ordre partiel P (Cheng et al. 2002).

Exemple 3.3

L'ensemble des ordonnancements semi-actifs (resp l'ensemble des ordonnancements actifs) est dominant pour tout critère régulier.

3.5 Notion de structure d'intervalles : [11]

Une structure d'intervalles est définie par un couple $< I, C >$ ou $I = i_1, i_2, i_3, \ldots, i_n$ est un ensemble d'intervalles et C est un ensemble de contraintes sur $I \times I$. Chaque intervalle l_j défini par un couple de points x_j et y_j tel que la relation d'ordre $x_j \prec_R y_j$ est vérifiée.

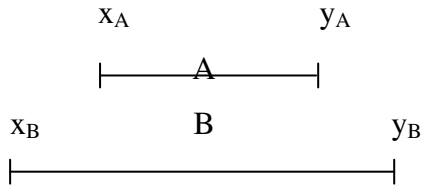
3.5.1 Sommet et base

Par considération de la relation d'Allen during [29], deux types particuliers d'intervalles peuvent être mis en évidence : l'intervalle de type sommet et celui de type base.

Définition 3.7

During (i, b) est une relation d'algèbre d'ALLEN

D'après la définition de la structure d'intervalles précédente on peut donner la relation d'Allen, during (i , b) graphiquement comme suit :



During (A, B): $x_B \prec x_A \leq y_A \prec y_B$

Figure 4 : Illustration de la relation d'Allen

Définition 3.3

Un intervalle s est dit sommet d'une structure d'intervalles $< I, C >$ s'il n'existe pas $i \in I$ tel que during (i, s).

Définition 3.4

Un intervalle b est dit base d'une structure d'intervalles $< I, C >$ s'il n'existe pas $i \in I$ tel que during (b, i).

En utilisant ces notions de sommet et de base, les notions de sommet-pyramide noté (s-pyramide) et de base pyramide noté (b-pyramide) peuvent être définies .[12]

3.5.2 S-pyramide et b-pyramide et algèbre d'ALLEN [13]

Définition 3.5

Une s-pyramide P_s, associée au sommet s d'une structure d'intervalles $< I;C >$, est le sous-ensemble d'intervalles $i \in I$ tel que during(s; i).

Définition 3.6

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

Algèbre d'Allen:

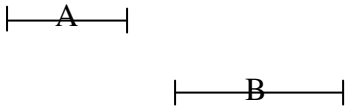
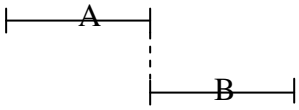
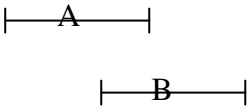
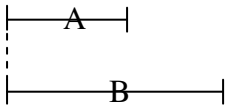
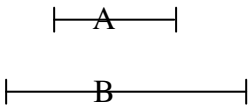
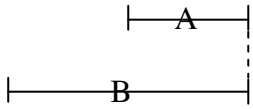
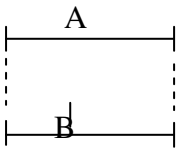
 Precedes (A, B) $x_A \leq y_A \leq x_B \leq y_B$	 Mets (A, B) $x_A \leq y_A = x_B \leq y_B$	 overlaps (A, B) $x_A \prec x_B \prec y_A \prec y_B$
 Starts (A, B) $x_A = x_B \prec y_A \prec y_B$	 during (A, B) $x_B \prec x_A \leq y_A \prec y_B$	 end (A, B) $x_A \prec x_B \leq y_A = y_B$
 Equals (A, B) $x_A = x_B = y_A = y_B$		

Tableau 2 : Les 7 relations d'algèbre d'ALLEN

3.5.3 Pourquoi des structures d'intervalles ?

L'utilisation de structures d'intervalles permet une représentation intéressante des caractéristiques d'un problème. En ordonnancement, il est par exemple possible d'exprimer par un intervalle, les rangs possibles d'affectation d'une tâche sur une ressource. De plus, comme nous le verrons dans les parties suivantes, une structure d'intervalles est bien adaptée à la représentation de corps d'hypothèses restreints, les sommets, bases et pyramides d'une structure pouvant être définie sans avoir connaissance des valeurs explicites des paramètres d'un problème.

Exemple 3.4

Soit la structure d'intervalles suivants :

Les sommets sont : T_3, T_4 et T_5 qui caractérisent les s-pyramides suivantes :

$P_3 = \{ T_1 , T_2 , T_7 \}$, $P_4 = \{ T_7 \}$, $P_5 = \{ T_6 , T_7 \}$.

Ainsi on identifie les bases : T_1, T_2, T_6 et T_7 qui caractérise les b-pyramides suivantes :

$P_1 = \{ T_3 \}$, $P_2 = \{ T_3 \}$, $P_6 = \{ T_5 \}$, $P_7 = \{ T_3 , T_4 , T_5 \}$.

Remarquer par exemple que le sommet T_3 n'appartient pas à la pyramide qu'il caractérise c.à.d. P_3

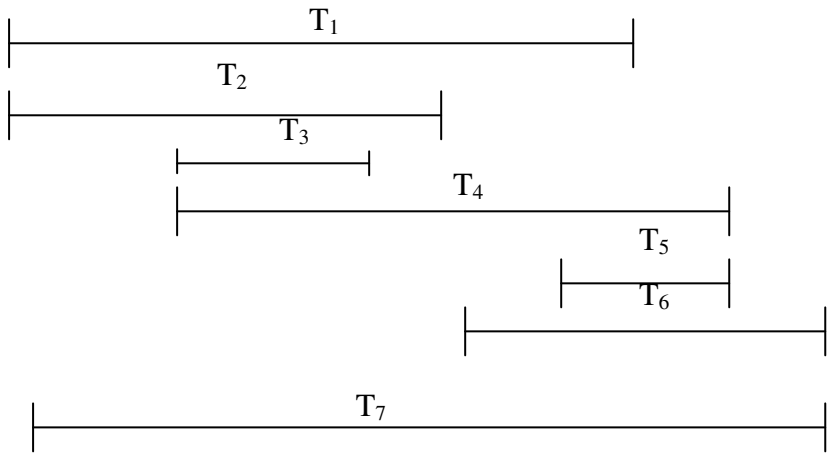


Figure 5 : Exemple d'une structure d'intervalle

CHAPITRE 4

APPLICATION DE LA ROBUSTESSE ET DE LA FLEXIBILITE SUR UN PROBLEME D'ORDONNANCEMENT A UNE SEULE MACHINE

Introduction:

Dans ce paragraphe, nous décrivons un ordre partiel dominant donné par La Hoing Trung pour le problème à une machine, défini par un théorème de dominance démontré dans les années quatre-vingts. Nous introduisons premièrement la description du problème d'ordonnancement à une machine, ainsi qu'une méthode éfficace permettant de le résoudre, puis, le théorème de dominance (théorème de pyramides) est présenté. Nous présentons une version améliorée d'algorithmes de (Tcor) 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 aux retards algébriques au mieux (noté L^{min}) et au pire (noté L^{max}). [13]

4.1 Description du problème

Nous considérons ici un problème P constitué d'un ensemble de travaux $T = \{T_1, T_2, \ldots, T_n\}$ à ordonnancer sur une machine . Chaque travail j est défini par sa durée opératoire p_j , sa date de début au plutôt r_j et sa date de fin au plus tard d_j. Le critère considéré est le plus grand retard algébrique L_{max}, noté $1|r_i|L_{max}$.

Lenstra et al montrent que problème $1|r_i|L_{max}$ est NP- difficile au sens fort. [29], mais Carlier 82 découvre une méthode de PSEP (procédure de séparation et d'évaluation progressive) et qui donne une solution optimale ; qui fournit en quelques secondes, une séquence optimale, pour des problèmes comptant plus de 1000 travaux [29]. Il a donné aussi une

méthode pour minimiser la longueur de l'ordonnancement dans un problème à une seule machine avec des dates de début au plus tôt, des durées de latence q_j. Algorithme de CARLIER.

Théorème 4.1 (Lenstra et al .1977)

Le problème 1/ r_j / L_{max} est fortement NP difficile.

Les cas ou le problème 1/ r_j / L_{max} est résolu :

Les trois problèmes particuliers suivants du problème $1/ r_j / L_{max}$ sont résolus en temps polynomial:

– Si $r_j = r$, pour tout j = 1,…, n .Un ordonnancement optimal est obtenu par la règle de Jackson. Ordonnancer dans l'ordre croissant des dates d'échue (EDD).

- Si $d_j = d$, pour tout j = 1, ….. n .Ordonnancer dans l'ordre croissant des dates de début d'exécutions.

- Horn en 1974 montre que si $p_j = 1$, pour tout j = 1,….n en tout instant, une tache disponible avec la plus petite date d'échue c'est en O (nlogn.

4.2 Présentation de l'algorithme de Carlier [11], [12], [13]

La méthode de Carlier construit une arborescence dans laquelle chaque nœud est associé à un problème à une machine partiellement séquencé. Le choix des relations de précédence est donné par l'algorithme de Schrage et exprimé par des actualisations de dates de début au plutôt et de durées de latence. L'algorithme de CARLIER se décompose en trois grandes étapes, la première et l'initialisation de l'algorithme de SCHRAGE, la deuxième le développement de l'arborescence à partir du nœud courant et la troisième est la recherche de nœud prochain pour la séparer.

Algorithme de CARLIER

Proc CARLIER (solution ,L_{max},r,d,p)

$d_{max} \leftarrow \max_{j \in T} d_j$; Pour i de 1 à n faire $q_i \leftarrow d_{max} - d_i$;FinPour ; **(1)**

SCHRAGE(meilleur_sol,F,r,q,p) ;

recherche le chemin critique $C = \lfloor i_1, i_2, ..., i_p \rfloor$ associé à meilleur_sol ;

Si $\min_{i_j \in C} q_j = q_{i_p}$ alors sommet_separable $\leftarrow$ **faux** ;

Sinon

Déterminer j et j_c ; $f_0 \leftarrow h(j); N \leftarrow \circ(f_0, r, q);$

sommet _separable $\leftarrow$; $\beta \leftarrow \circ$

Finsi

Tantque sommet_separable = **vrai** faire

à partir du sommet β : **(2)**

- Considérer le problème β_1 ou $j_c \prec j$:

$r^{\beta_1} \leftarrow r^{\beta}; q^{\beta_1} \leftarrow q^{\beta}; q^{\beta_1}_{j_c} \leftarrow \max(q^{\beta_1}_{j_c}, \sum_{i_j \in j} p_{i_j} + q^{\beta_1}_{i_p})$

$f_{\beta_1} \leftarrow \max \lfloor f_{\beta_1}, h(j \cup j_c) \rfloor$; $Si \quad f_{\beta_1} \prec Falors \quad N \leftarrow N \cup \beta_1$; $FinSi$ (*)

- Considérer le problème β_2 ou $j \prec j_c$:

$f_{\beta_2} \leftarrow \max \lfloor f_{\beta_1}, h(j \cup j_c) \rfloor$; $Sif_{\beta_2} \prec F \ alors \ N \leftarrow N \cup \beta_{21}$; $FinSi$(*)

$N \leftarrow N \setminus \beta$

parcours $\leftarrow$ **vrai ;**

Tantque $N \neq \emptyset$ et parcours = vrai faire **(3)**

recherche le sommet β d'évaluation f_β minimale ;

SCHRAGE (sol_sch,C_sch, r^{β}, q^{β}, p) ;

Si C _ sch < F alors

$F \leftarrow C\_sch; meilleure\_ \leftarrow sol \leftarrow \_sch;$

Pour tout $v \in N$ faire

Si $f_v \prec F \ alors \ N \leftarrow N \setminus v$;FinSi

FinPourTout

FinSi

Recherche le chemin critique $C = \lfloor i_1, i_2, ..., i_p \rfloor$ associé à sol_sch ;

Si $\min_{i_j \in C} q_j = q_{i_p}$ alors sommet _ separable $\leftarrow$ **Faux ;** $N \leftarrow N \setminus \beta$

Sinon

Déterminer j et j_c ; $f_\chi \leftarrow \max(f_\beta, h(j))$;(*)

parcours $\leftarrow$ **Faux ;**sommet_separable $\leftarrow$ **vrai ;**

FinSi

FinTantQue

FinTantQue

Solution_ $\leftarrow meilleure\_sol; L_{max} \leftarrow F - d_{max}$

FinProc.

(*) le calcule de $h(j \cup j_c)$ *et de* $h(j)$ est éffectué avec les les dates de début au plus tot et les durées de latence du sommet considéré.

Figure 6 : Algorithme de CARLIER

4.3 Description de l'algorithme de (Schrage) [12]

L'algorithme de Schrage correspond à une reformulation de l'algorithme EDD (Earliest Due Date) de Jackson (les travaux sont séquencés en ordre croissant des d_j). Il sélectionne à chaque itération le travail prêt de plus grande durée de latence q_j. Donnée par la formule suivante $q_j = \max_{j \in T} d_j - d_i$ avec q_j est un temps incompressible entre la fin de j et la fin de l'ordonnancement.

Il développe une évaluation optimiste du critère et associé à chaque nœud une borne inférieure et une borne supérieure de C_{max} par une formule bien déterminée.

Il est démontré aussi que si le sommet considéré contient une solution optimale unique et par la détermination d'un ensemble des travaux critiques noté J_c ,alors soit J_c est séquencé avant tous les travaux de J, soit tous les travaux de J sont séquencés avant J_c . L'arborescence est donc développée en considérant deux nouveaux problèmes partiellement ordonnancés.

La. Hoing Trung.2005 propose dans un algorithme appelé (TCOR) une solution robuste, incorporant la flexibilité séquentielle, ainsi une nouvelle méthode, basée sur un théorème de dominance, par considération d'un corps d'hypothèses restreint, ce théorème permet de définir un ordre partiel dominant qui caractérise un ensemble de séquences dominantes vis-à-vis de l'admissibilité, ou de l'optimalité (pour un critère de plus grand retard algébrique de type L_{max} par exemple).

Algorithme de (SCHRAGE) :

procSCHRAGE (Solution,C_{max},r,q,p)
$C_{max} \leftarrow 0$;Solution $\leftarrow$ Tableau vide à n positions
Fin $\leftarrow$ 0;A séquencer $\leftarrow$ T;
Pour i de 1 à n faire
Début $\leftarrow$ max (Fin, $\min_{j \in A} seqencer_j^T$) ,Solution [i] $\leftarrow$ Travail
$K \in A$ Seqencer prêt $r_k \leq$ Debut de plus grand durée de latence ;
A Séquencer $\leftarrow$ A séquencer \ { k } ,
Fin $\leftarrow$ Début + p_k ;
$C_{max} \leftarrow$ max (C_{max} Fin + q_k) ;
Fin pour
Fin proc

Figure 7 : Algorithme de SCHRAGE

4.4 Le théorème des pyramides (théorème de dominance)[13]

Ce théorème a été formulé, en dehors de tout contexte de recherche de robustesse.

Pour la caractérisation de l'ensemble de séquences dominantes, considérons la structure d'intervalles < I, C > où un intervalle $i_j \in I$ caractérisé par une date de début r_j et une date de fin d_j. [I_j , r_j] est associé a chaque tâche j du problème .

Remarque 4.1

Remarquons que les sommets sont supposés indicés dans l'ordre croissant de leurs r_j ou en cas d'égalité, dans l'ordre croissant de leurs d_j . Autrement dit que étant donné deux sommets de la structure d'intervalle S_α et S_β alors : $\alpha \prec \beta$ si et seulement si $r_{S_\alpha} \leq r_{S_\beta}$ et $d_{S_\alpha} \leq d_{S\beta}$, et si deux sommets

possèdent des dates de début au plus tôt et de fin au plus tard identiques, alors ils peuvent être indicés de façon quelconque.

Remarque 4.2

Dans le cas ou les deux sommets possèdent des dates de début au plus tôt et de fin au plus tard identique, alors ils peuvent être indicés de façon quelconque.

La s-pyramide notée P_α désigne la pyramide caractérisée par le sommet S_α, notons aussi que u (i_j) est l'indice de la première s-pyramide à laquelle la tâche i_j appartient et v (i_j) est l'indice de la dernière pyramide à la quelle la tâche i_j appartient .

Enoncé de théorème 4.2 [13]

Un ensemble dominant de séquences peut être constitué par les séquences telles que :

1- les sommets sont ordonnés dans l'ordre de leurs indices.

2- avant le premier sommet, seuls sont placés des travaux appartenant à la première s-pyramide, rangés dans l'ordre croissant de leurs r_j ou, en cas d'égalité, dans un ordre arbitraire.

3-après le dernier sommet, seuls sont placés des travaux appartenant à la dernière S -pyramide rangés dans l'ordre croissant de leurs d_j ou, en cas d'égalité, dans un ordre arbitraire.

4-entre deux sommets S_k et S_{k+1} sont placés en premier des travaux appartenant à la s-pyramide P_k. et n'appartenant pas à P_{k+1} dans l'ordre croissant de leurs d_j (dans un ordre arbitraire en cas d'égalité). Puis des travaux communs aux deux pyramides P_k et P_{k+1} dans un ordre arbitraire, et enfin des travaux appartenant à la pyramide P_{k+1} et n'appartenant pas à P_k dans l'ordre croissant de leurs r_j (dans un ordre arbitraire en cas d'égalité).

Le théorème des dominances garantit la dominance de l'ensemble des séquences vis-à-vis de l'admissibilité. Pour cela Erschler et al ont montré dans un nouveau concept de dominance pour l'ordonnancement de travaux

sur une machine que la dominance est aussi garantie vis-à-vis du plus grand vrai retard T_{max} .[12]

Ainsi que Cyril et al .03 montre qu'il existe toujours une séquence optimale vis-à-vis du retard algébrique L_{max} dans l'ensemble dominant de séquences caractérisé grâce au théorème des pyramides. [13]

Proposition 4.1 [Cyril et al .2003] (voir [13])

Il existe toujours une séquence optimale vis-à-vis du retard algébrique dans l'ensemble dominant de séquences caractérisé grâce au théorème de dominance.

Preuve : par l'absurde

Soit p telle que $p = \{j / r_j, p_j, d_j\}$ l'ensembles des tâches à séquencer, L_{max} le retard optimale du problème p, SP l'ensemble dominant de séquences dominée déterminer grâce au théorème de dominance et $L_{max}(Sp)$ le plus grand retard sur S_p .

Supposons qu'il n'existe aucune solution optimale dans Sp , c.a.d $L_{max}(Sp) - L_{max} > 0$ considérons dans ce cas un nouveau problème p' telle que $p' = \{j' / r_j = r_{j'}, p_j = p_{j'}, d_{j'} = d_j + L_{\max}\}$ et on a $S^p = S^{p'}$.d'autre part $L^{p'}_{\max}(S^{p'}) = L^{p}_{\max}(S^{p}) - L^{P}_{MAX}$ .comme les dates de fin d'exécutions des tâches du problème p' sont incrémentées de la valeur du meilleur retard algébrique alors le problème p' est admissible d'où $L^{p'}_{\max}(S^{p'}) \le 0$ d'où $L^{p}_{\max}(S^{p}) - L^{p}_{\max} \le 0$ d'où la contradiction avec l'hypothèse . Donc on a au moins une séquence optimale.

4.5 Quelque Remarques sur le théorème

Le théorème des pyramides définit un ordre partiel dominant et il donne des relations de précédence entre les sommets, et entre les travaux non sommets et les sommets d'autre part.

Nous remarquons aussi que le théorème, donne une classe entre les travaux attribués devant un sommet par ordre croissant des dates de

début d'exécution , et à ceux attribués derrière un sommet par ordre croissant des dates de fin d'exécution. D'ou l'observation suivante:

Observation [M. Abbas, M.Boukedroun]

La flexibilité séquentielle pouvant potentiellement être mise en évidence par le théorème de dominance recouvre celle pouvant être offerte par le concept de groupes des tâches permutables.

Preuve :

Un groupe de tâches totalement permutable correspond à un ensemble de sommets possédant des dates de débuts et de fin égale dans le sens ou le dernier caractérise exactement le même ensemble de séquences que le premier.

Proposition 4.2 [M. Abbas, M.Boukedroun]

Etant donné deux travaux non sommet i et j tels que $r_i < r_j$, et que i et j appartenant à une seule pyramide de sommet s, sont tous deux séquencés avant s alors, les deux ordres suivantes sont interdit:
$i < j$ et $j < i < s$.

4.6 Exemples : Calcul des sommets et des pyramides et leur structure d'intervalles associée

Exemple : 4.1 [Annexe]

Considérons le problème à sept travaux dans les dates de début, les dates de fin et les durées opératoires .Dans l'application du théorème des pyramides, seul l'ordre relatif entre les dates de fin au plus tôt et de fin au plus tard sont nécessaire .

Tâches	T_1	T_2	T_3	T_4	T_5	T_6	T_7
r_i	10	13	11	20	30	0	30
p_i	5	6	7	4	3	6	2
d_i	44	25	27	30	43	34	51

Tableau 3 : Une instance de problème à une seule machine (7Tâches)

Pour la structure d'intervalles associée à cet exemple, on distingue quatre sommets : $s_1 = T_2$, $s_2 = T_4$, $s_3 = T_5$ et $s_4 = T_7$, et qui caractérise les quatre pyramides : $P_1 = \{T_1, T_3, T_6\}$, $P_2 = \{T_1, T_6\}$, $P_3 = \{T_1\}$ et $P_4 = \emptyset$.

Remarquons que les sommets n'appartiennent pas à la pyramide qu'ils caractérisent.

Par l'application de théorème de dominance à cette structure d'intervalles, nous obtenons des ensembles de séquences dominantes, par exemple :

$T_6 < T_1 < T_3 < T_2 < T_4 < T_5 < T_7$, $T_1 < T_2 < T_3 < T_6 < T_4 < T_5 < T_7$,

$T_2 < T_3 < T_4 < T_6 < T_5 < T_1 < T_7$.

Attention

La séquence $T_1 < T_2 < T_6 < T_3 < T_4 < T_5 < T_7$ ne fait pas partie de l'ensemble dominant de séquences car le travail T_3 n'appartenant qu'à la pyramide P_1 qui ne peut pas être placé après le travail T_6 .car possédant une date de fin plus grande ($d_{T6} > d_{T3}$).

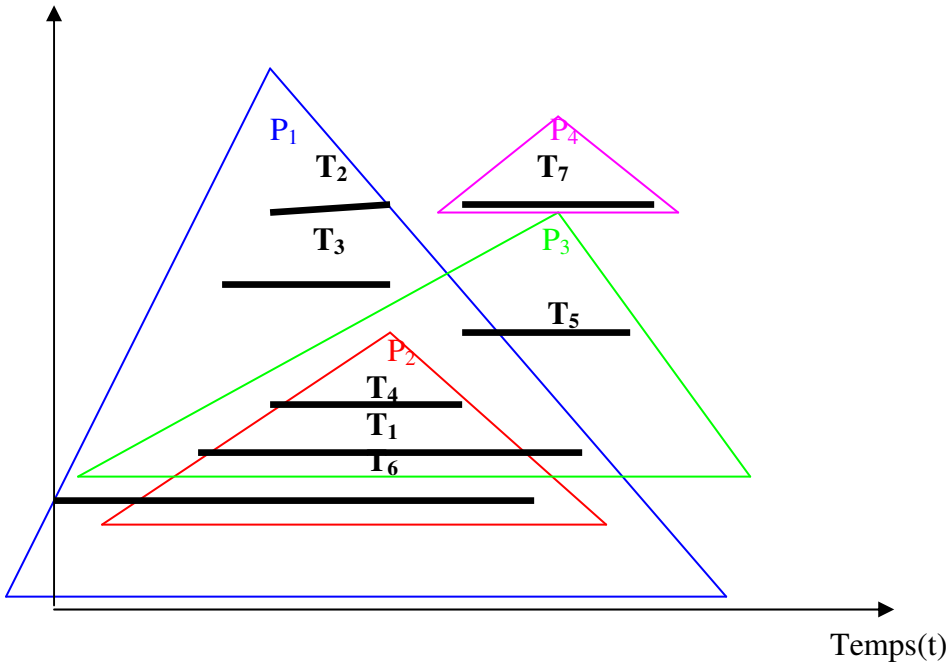


Figure 8 : Structure d'intervalles, sommets et pyramides du tableau 3

Exemple : 4.2

Considérons le problème d'ordonnancement à six travaux dans le but de minimiser le plus grand retard algébrique L_{max}.

Tâches	T_1	T_2	T_3	T_4	T_5	T_6
r_i	2	1	0	6	4	8
d_i	3	5	10	7	9	11

Tableau 4 : Une instance de problème à une seule machine (6Tâches)

La structure d'intervalle contient trois sommets $S_1 = T_1$, $S_2 = T_4$, $S_3 = T_6$ qui caractérisent trois pyramides $p_1 = \{T_2, T_3\}$ et $p_2 = \{T_5, T_3\}$, $p_3 = \emptyset$ remarquons qu'un sommet n'appartient pas à la pyramide qu'il caractérise.

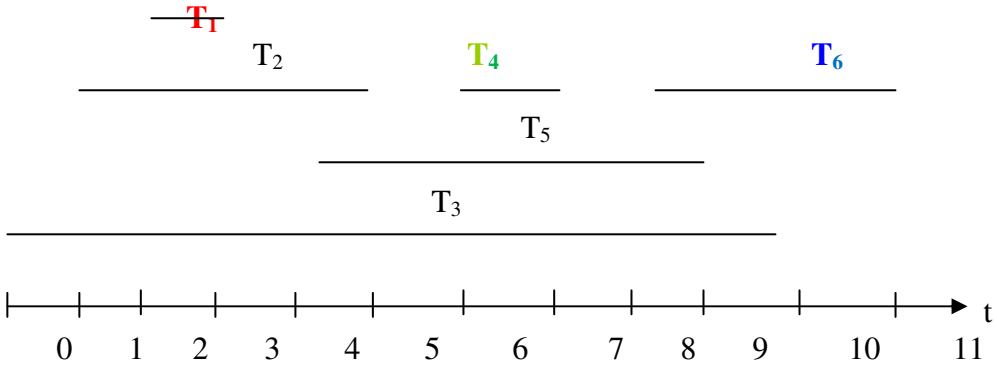


Figure 9 : Illustration d'une structure d'intervalles

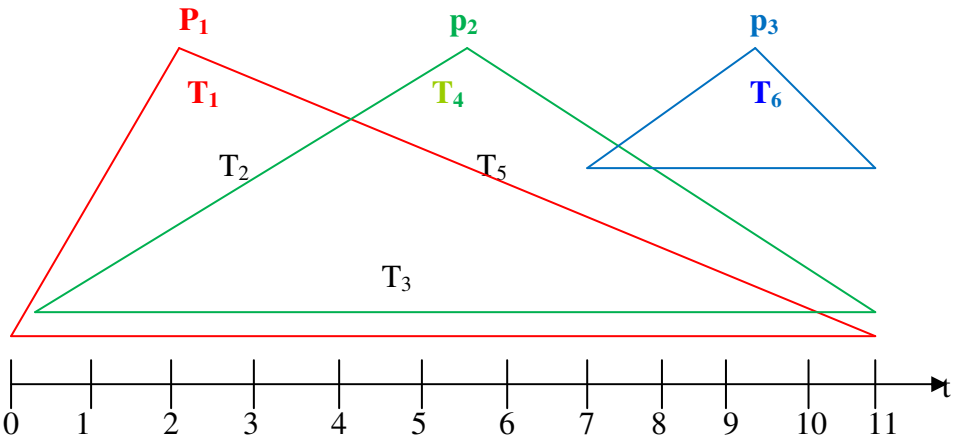


Figure 10 : Illustration d'une structure d'intervalles (pyramides)

Exemple 4.3

Considérons le problème d'ordonnancement à cinq travaux dans le but est de minimiser le plus grand retard algébrique L_{max}.

Tâches	T_1	T_2	T_3	T_4	T_5
r_i	6	1	21	24	4
p_i	4	5	6	8	7
d_i	13	37	33	31	17

Tableau 5 : Une instance de problème à une seule machine (5 Tâches)

La structure d'intervalle contient deux sommets $S_1 = T_1$, $S_2 = T_4$, qui caractérisent deux pyramides $p_1 = \{T_2, T_5\}$ et $p_2 = \{T_2, T_3\}$, remarquons que un sommet n'appartient pas à la pyramide qu'il caractérise.

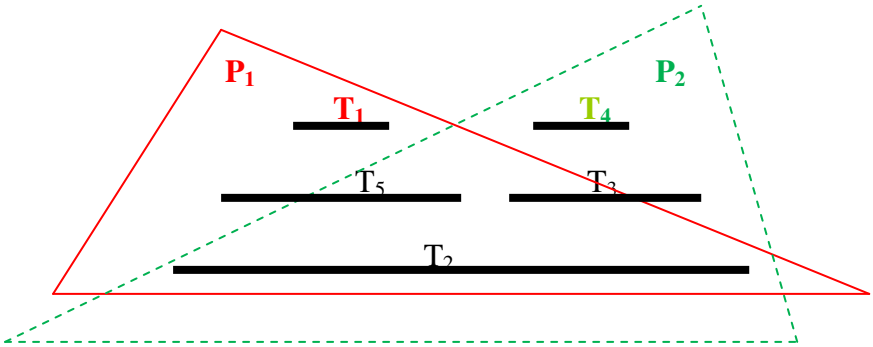


Figure 11 : Structure d'intervalle de problème du tableau.5

Par l'application de théorème de dominance à cette structure d'intervalles, nous obtenons des ensembles de séquences dominantes, par exemple :

$T_2 < T_5 < T_1 < T_3 < T_4$, $T_1 < T_5 < T_3 < T_4 < T_2$, $T_5 < T_1 < T_4 < T_3 < T_2$.

4.7 Extensions du théorème [11], [12], [13]

Proposition 4.3 (La Hoing Trung.2005) [13]

Si un travail j appartient à deux pyramides P_1 et P_2 dont les sommets s_1 et s_2 possèdent des dates de début égales, alors toute séquence plaçant j entre s_1 et s_2 est dominée par au moins une autre séquence de l'ensemble dominant.

Preuve

La preuve est facile, il suffit de dire que pour tout travail j appartenant à deux pyramides p_1 et p_2 de sommets respectivement s_1 et s_2 avec $r_{S_1} = r_{S_2} = r_S$ et $d_{S_1} \prec d_{S_2}$, nous avons aussi la relation $r_i \prec r_S$ et $d_i \succ d_{S_2}$ .Par l'application de théorème ,toute séquence dominante est telle que $S_1 < S_2$.
Si le travail j est placé après le sommet S_1 alors sa date de début d'exécution vérifier la relation suivante $r_j = r_{S_1} = r_S$. D'autre part les trois intervalles associés aux travaux S_1, S_2 et j forment donc une structure d'intervalles (Une structure en escalier) telle que $r_j = r_{S_1} = r_{S_2}$ et $d_{S_1} \prec d_{S_2} \prec d_j$, et pour une telle structure on sait que $S_1 < S_2 < J$ rangeant les travaux par date de fin d'exécution croissante est dominante, en particulier devant la séquence $S_1 < j < S_2$ plaçant j entre les deux sommets S_1 et S_2.

Proposition 4.4 (La Hoing Trung. 2005) [13]

Si un travail j appartient à deux pyramides P_1 et P_2 dont les sommets s_1 et s_2 possèdent des dates de fin égales, alors toute séquence plaçant j entre s_1 et s_2 est dominée par au moins une autre séquence de l'ensemble dominant.

Preuve

La preuve de cette proposition est analogue à la preuve précédente, il suffit d'inverser la structure, ($d_j = d_{S_1} = d_{S_2}$ et $r_j \prec r_{S_1} \prec r_{S_2}$) et on déduit que la séquence $S_1 < J < S_2$ est dominée par la séquence dominante $j < S_1 < S_2$ rangeant les travaux par ses dates de début d'exécution croissante.

Proposition 4.5 [M. Abbas, M.Boukedroun]

Si un travail j appartient à deux pyramides P_1 et P_2 dont les sommets s_1 et s_2 possèdent des dates de début et de fin égales, alors toute séquence plaçant j entre s_1 et s_2 est dominée par au moins une autre séquence de l'ensemble dominant.

Preuve :

Le travail j appartenant aux deux pyramides P_1 et P_2 de sommets s_1 et s_2, possède des dates de début égales et de fin égales.

D'une part les trois intervalles associés aux travaux s_1 et s_2 et j forment une structure en escalier c.a. d $r_j = r_{s_1} = r_{s_2}$ et $d_{s_1} \prec d_{s_2} \prec d_j$ (puisque les dates de début sont égales), d'où la séquence $s_1 \prec j \prec s_2$ plaçant j entre les deux sommets.

D'autre part les trois intervalles associés aux travaux s_1 et s_2 et j forment une structure en escalier inversée c.à.d. $d_j = d_{s_1} = d_{s_2}$ et $r_j \prec r_{s_1} \prec r_{s_2}$ (puisque les dates de fins sont égalent), d'où la séquence $s_1 \prec j \prec s_2$ est dominée par la séquence dominante $j \prec s_1 \prec s_2$, donc de la première et la deuxième en déduit que toute séquence plaçant j entre s_1 et s_2 est dominée par au moins une autre séquence de l'ensemble dominant.

4.8 Une borne inférieure pour l'ensemble des séquences dominantes

Propriété 4.1

Grâce au théorème de dominance l'ensemble de séquences dominantes est caractérisé par la formule suivante :

$$\left|S_{dom}\right| = \prod_{q=1}^{q=N} (1+q)^{n_q} \quad \text{.....(3)}$$

n_q : le nombre de travaux non sommets appartenant à q pyramides

N : le nombre de pyramides.

Application :

On appliquant la formule précédente sur les trois exemples on trouve la cardinalité de l'ensemble de séquences dominante de chaque exemple et on a:

Exemple 4.1 : $\left|S_{dom}\right| = (1+1)^1.(2+1)^1.(3+1)^1 = 24$

Exemple 4.2 : $\left|S_{dom}\right| = (1+1)^2.(2+1)^1.(3+1)^0 = 12$

Exemple 4.3 : $\left|S_{dom}\right| = (1+1)^2.(2+1)^1 = 12$ Les séquences dominantes sont :

$T_2 \prec T_5 \prec T_1 \prec T_3 \prec T_4$*$T_2 \prec T_5 \prec T_1 \prec T_4 \prec T_3$*$T_5 \prec T_1 \prec T_2 \prec T_3 \prec T_4$*$T_5 \prec T_1 \prec T_2 \prec T_4 \prec T_3$
$T_5 \prec T_1 \prec T_3 \prec T_4 \prec T_2$* $T_5 \prec T_1 \prec T_4 \prec T_3 \prec T_2$* $T_2 \prec T_1 \prec T_5 \prec T_3 \prec T_4$* $T_2 \prec T_1 \prec T_5 \prec T_4 \prec T_3$
$T_1 \prec T_5 \prec T_2 \prec T_3 \prec T_4$* $T_1 \prec T_5 \prec T_2 \prec T_4 \prec T_3$* $T_1 \prec T_5 \prec T_3 \prec T_4 \prec T_2$* $T_1 \prec T_5 \prec T_4 \prec T_3 \prec T_2$

Tableau 6: Les séquences dominantes pour l'exemple 4.3

Proposition 4.6 : [M. Abbas, M.Boukedroun]

Les permutations des travaux T_j avec j = 1,2,…, n placées entre deux sommets S_k et S_{k+1} appartenant en même temps aux S-pyramides P_k et P_{k+1} respectivement sont strictement équivalentes en terme de qualité.

Preuve :

En effet, pour le premier exemple précédent les trois travaux non sommet T_1, T_3, T_6 sont placé entre les deux sommets T_2 et T_4 et comme T_3 appartenant seulement à la pyramide s_1 du sommet T_2 elle doit nécessairement précéder les deux autres travaux T_1 et T_6 alors la séquence $T_2 \prec T_3 \prec T_6 \prec T_1 \prec T_4 \prec T_5 \prec T_7$ est comptabilisée. Par contre la séquence $T_2 \prec T_3 \prec T_1 \prec T_6 \prec T_4 \prec T_5 \prec T_7$ n'est pas comptabilisée malgré qu'elle respecte bien le théorème de pyramide. D'où la proposition suivante

Proposition 4.7:

La valeur $|S_{dom}|$ est une borne inférieure du nombre réel de séquences dans l'ensemble dominant.

Preuve : La preuve découle de la proposition précédente

Illustration de l'ensemble dominant de séquences **:**

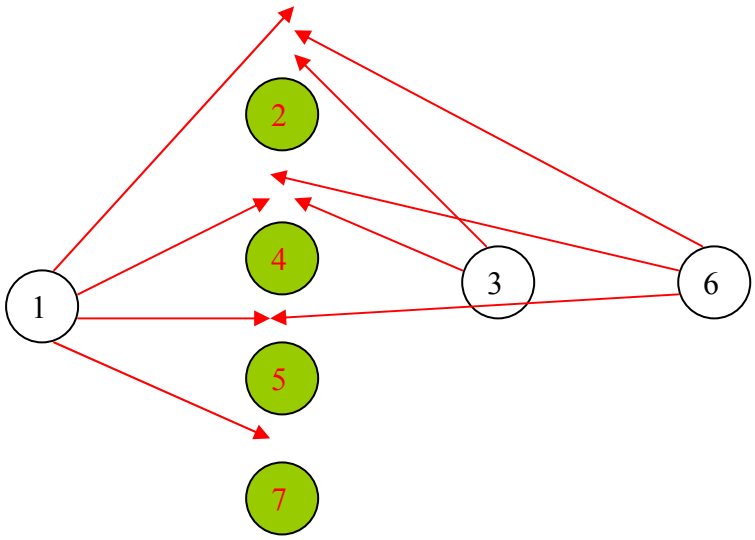


Figure 12: Illustration de l'ensemble dominant de séquences de l'exemple 4.1

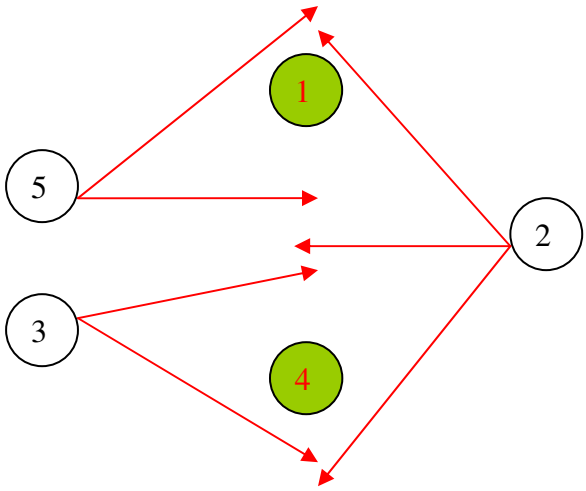


Figure 13: Illustration de l'ensemble dominant de séquences de l'exemple 4.3

4.9 Séquence maître pyramide [43]

Pour éviter d'énumérer toutes les séquences S_Δ que le théorème des pyramides caractérise. Cyril Briand et al montrent qu'il est possible de même que la condition de dominance de (Dauzeres –pérés et Sevaux) de définir une séquence particulière noté σ_Δ " maître pyramide " de la forme:

$\sigma_\Delta = \alpha_0 s_0 \beta_0 \ \alpha_1 s_1 \beta_1 .... \alpha_m s_m \beta_m$ .....(4)

Telle que : - m est le nombre total de sommets.

- α_i est l'ensemble des travaux appartenant à la pyramide Pi classés par ordre croissant de leur date de disponibilité et β_i est l'ensemble des travaux appartenant à la pyramide Pi classés par ordre croissant de leur date de échue.

Proposition 4.8 (Erschler et al .1983) [13]

Toute séquence de la forme $\sigma_\Delta = \alpha_0 s_0 \beta_0 \ \alpha_1 s_1 \beta_1 .... \alpha_m s_m \beta_m$ est dominante vis-à-vis de l'admissibilité.

S_k sommet de la pyramide k (m sommets)

α_{ik} Séquence de travaux appartenant à la pyramide, p_k classée par ordre croissant de r_i β_i séquence de travaux appartenant à la pyramide, p_k classée par ordre croissant de d_i.

Remarque :

Un travail non sommet peut se trouve soit dans α_{ik} soit dans β_i s'il appartient à p_k.

4.10 Quelques propriétés de dominance :

►s'il existe une séquence admissible $\sigma_1 \notin \sigma_\Delta$ alors il existe une solution $\sigma_2 \in \sigma_\Delta$ de sorte que $\sigma_1 admissible \Rightarrow \sigma_2 admissible$.

►s'il n'existe pas de solution admissible parmi les solutions dominantes alors il n'existe pas de solution de problème posée. Dans ce cas la réduction de l'espace de recherche est obligatoire. « Corps d'hypothèse restreint ».

Exemple 4.4

On Considère un problème de cinq travaux

Travaux	T_1	T_2	T_3	T_4	T_5
r_i	0	5	8	12	14
P_i	8	6	5	6	10
d_i	16	26	24	22	32

Tableau 7 : Problème d'une seule machine

Il y a trois sommets , $S_1 = T_1$, $S_2 = T_4$, $S_3 = T_5$ .les pyramides associées à ces sommets sont : $P_1 = \emptyset$, $P_2 = \{ T_2 , T_3 \}$, $P_3 = \emptyset$ .Dans cet exemple il y a donc quatre séquences dominantes pour la recherche de séquence admissible à cinq travaux donc $S_\Delta = \{ (T_1,T_2,T_3,T_4,T_5) , (T_1,T_2,T_4,T_3,T_5) , (T_1,T_3,T_4,T_2,T_5) , (T_1,T_4,T_3,T_2,T_5) \}$.

La séquence maître pyramide est $\sigma_\Delta = (T_1 , T_2 , T_3 , T_4 , T_3, T_2 , T_5)$, elle caractérise bien les quatre séquences précédentes .

CHAPITRE 05

AMELIORATION D'UNE APPROCHE D'ORDONNANCEMENT ROBUSTE POUR UN PROBLEME D'UNE SEULE MACHINE

5.1 Notation

U (j) : L'indice de la première pyramide pour laquelle le travail j est affecté

V (j) : L'indice de la dernière pyramide pour laquelle le travail j est affecté

$L_j^{\max}$: Le retard au pire de travail j ;

$L_j^{\min}$: Le retard au mieux de travail j ;

$\Pr ed_j^{\min}$: L'ensemble de travail devant précéder j ;

$\Pr ed_j^{\max}$: L'ensemble de travail j restant ;

$Seq_j^{\min}$: Séquence optimale obtenue par la règle de Jackson.

5.2 Caractérisation de l'ensemble $\mathrm{seq_j^{min}}$

Proposition 5.1 (La Hoing Trung. 2005) [13]

Une meilleure séquence favorable $seq_j^{\min} \in S_{dom}$ pour un travail j est $\sigma_1 \prec t_1 \prec ..... \prec \sigma_{u(j)-1} \prec t_{u(j)-1} \prec j$ Avec $\sigma_k = \left\{ i \in pred_j^{\min} \middle| u(i) = k \right\}$ sont séquencées dans l'ordre croissant de leur due date.

Preuve

Pour déterminer $L^{\min}$, supposons seules les travaux séquencés avant le travail j sont considérés.

Supposons $j \in u(j)$, aussi l'ensemble des travaux k tells que $v(k) \prec u(j)$ noté $\Pr ed_j^{\min}$ est considéré aussi.

Pour la détermination de $L^{\min}$, il suffit de minimiser $C_{\max}$ de l'ensemble composé de travaux de $\Pr ed_j^{\min}$ et pour la détermination de l'ensemble $Seq_j^{\min}$ il suffit aussi d'appliquer la règle de Jackson qui consiste à classer les travaux

par ordre croissant de dates de début d'exécution, finalement remarqué bien que cette procédure respecte bien le théorème de dominance.[12]

Exemple

Déterminons $Seq_5^{\min}$ de la tâche T_5 de l'exemple 4.1, en effet , le travail T_5 est un sommet qui n'appartient qu' à la pyramide qu'il caractérise de plus $u(T_5) = 3$ d'où $pred_5^{\min}$ comprenant des travaux placés avant T_5 de toute séquence dominante alors , pour tout travail k on a $v(k) < 3$, donc $pred_5^{\min} = \{T_2, T_3, T_6, T_4\}$ .Par l'utilisation de la règle de Jackson on obtient $seq_5^{\min} = T_6 \prec T_3 \prec T_2 \prec T_4 \prec T_5$.

5.3 Evaluation des performances d'un tel ensemble de séquences dominantes du point de vue de retard algébrique [13]

5.3 .1 Calcul de retard au mieux [Annexe]

Déterminer $L_j^{\min}$ revient alors à minimiser la durée totale $C_{\max}$ du problème d'ordonnancement constitué des travaux de l'ensemble $\Pr ed_j^{\max}$, mais en respectant tout d'abord le théorème de dominance. Par l'application de la règle de JACKSON et pour minimiser le $C_{\max}$ une séquence $Seq_j^{\min}$ optimale est déterminée.

Pour l'évaluation du retard au mieux (La Hoing Trung.2005) a donné un algorithme de complexité de O (nlogn), ainsi une formule de détermination de $L_j^{\max}$ est décrite par :

$$L_j^{\min} = \max(C_{\max}^{seq_j^{\max}}, r_j) + p_j - d_j \text{ ....(5)}$$

$$C_{\max}^{seq_j^{\min}} = \max(S_k^{seq_j^{\min}} + p_k), \forall k \in pred_j^{\min} \text{ ....(6)}$$

Algorithme de calcule L^{min}:

Proc CalculerL$_j$ Min (j , T) (*)

$Pred_j^{min} = \{\}$;

Pour chaque $i \in T$ faire

Si v(i) < u(j) alors $Pred_j^{min} \leftarrow Pred_j^{min} \bigcup i$; finSi

Fin pour

Pour chaque $i \in Pred_j^{min}$ faire

Affecter i à la pyramide d'indice u (i) ;

Fin pour

Pour chaque pyramide P_k , telle que k < u (j) faire

$Seq_{pk}^{min} \leftarrow$ Travaux séquencés par ordre croissant des r_i

$$Seq_j^{min} \leftarrow Seq_{p_0}^{min} \prec ..... \prec Seq_{p_{u(j-1)}}^{min} \prec j$$

Calculer L_j^{min} pour la Seq_{pk}^{min} ;

Finproc

Figure 14 : Algorithme de calcule de L_j^{min}

5.3.2 Calcul du retard au pire [Annexe]

Déterminer L_j^{max} revient alors à maximiser le makespan C_{max} des travaux de l'ensemble $Pred_j^{max}$, mais en respectant tout d'abord le théorème de dominance.

En effet Pour maximiser le C_{max} une version inversée de la règle de JACKSON est utilisée pour classer les travaux par ordre décroissant de r_i.

Algorithme de calcule L^{max}

Proc CalculerL$_j$Max (j , T) (*)

$pred_j^{s1} = \{\}$;

$pred_j^{s2} = \{\}$_;

A = { };

B = { };

Pour chaque $i \in T$ faire

Si v(i) < u(j) alors $Pred_j^{s1} \leftarrow Pred_j^{s1} \cup i$; finSi

Si (u i) ≤ v (j) ≤ v (i) alors $Pred_j^{s2} \leftarrow Pred_j^{s2} \cup i$; FinSi

Fin pour

Pour chaque $i \in pred_j^{s1}$ faire

Affecter i à la pyramide v i) ;

Fin pour

Pour chaque pyramide P_k , k < v (j) faire

$Seq_{pk}^{max} \leftarrow$ Travaux séquencés par ordre croissant des d_i ;

Fin pour

$$S_1 \leftarrow Seq_{p_0}^{\max} \prec ..... \prec Seq_{p_{v(j)-1}}^{\max}$$

Pour chaque $i \in pred_j^{s2} - \{S_{v(j)}\}$ faire

Si $d_i > d_j$ alors

$A \leftarrow A + \{i\}$;

SiNon

$B \leftarrow B + \{i\}$;

FinSi

Fin pour

Séquence A par ordre croissant des r_i ;

SéquenceB 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.

Figure 15 : Algorithme de calcule L_j^{max}

Proposition 5.2 (La Hoing Trung. 2005) [13]

Soit une structure d'intervalles caractérisées par une unique s-pyramide P de sommet s, la séquence la plus défavorable vis-à-vis du retard algébrique d'un travail j appartenant à P est obtenue en plaçant tout travail k $\in$ P –{s} tel que $d_k > d_j$ avant s par ordre de r_i croissant, sinon après s par ordre croissant de d_i.

Preuve

Même démonstration que celle de proposition précédente.

La Hoing Trung donne pour l'évaluation de retard au pire un algorithme de complexité de O (nlogn), ainsi une formule de détermination de $L_j^{\max}$ est décrite

$$L_j^{\max} = \max(C_{\max}^{seq\ j^{\max}}, r_j) + p_j - d_j \quad \text{..... (7)}$$

5.3.3 Diagramme de retard [13]

Remarque 5.1

Il est important de disposer d'une représentation visuelle permettant de mettre en évidence les performances de l'ensemble des séquences dominantes, on introduit un intervalle de type [L^{min} , L^{max}] pour chaque travail j de problème p , cet intervalle est appelé diagramme de retard.

Par l'application du théorème de dominance et les deux algorithmes de calcul du L^{min} et L^{max} on trouve deux bornes pour chaque séquence de l'exemple 4.1.

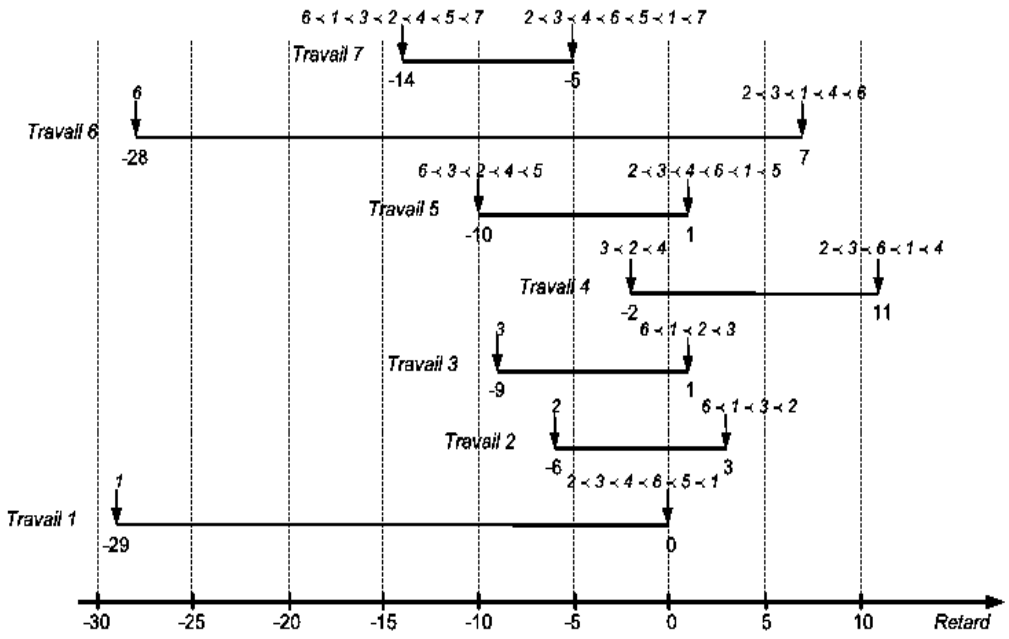


Figure 16 : Diagramme de retard de l'exemple 4.1

5.4 L'intérêt de diagramme de retard

Ce diagramme permet à un décideur de visualiser les performances associées à l'ensemble de séquences dominantes S_{dom} d'un problème, du point de vue du retard algébrique. De plus il n'existe pas de séquences permettant de garantir l'obtention simultanée pour tous les travaux de leur , retard algébrique au mieux ou au pire.

Autrement dit, imposer une séquence partielle permettant à un travail de se terminer au plutôt se traduit nécessairement par une augmentation des retards au mieux de certains travaux. Inversement, imposer une séquence partielle, la pire pour un travail donné, tend à réduire les retards algébriques au pire des autres travaux.

5.5 Exemple illustratif des calculs

Application : sur l'exemple 4.1

L'exemple 4.1 est repris de l'article de CARLIER [29]

Tâches	T_1	T_2	T_3	T_4	T_5	T_6	T_7
r_i	10	13	11	20	30	0	30
p_i	5	6	7	4	3	6	2
d_i	44	25	27	30	43	34	51

Tableaux 8 : Une instance du problème d'une seule machine

Etape01

Le théorème de dominance dans cet exemple nous permet de distinguer quatre sommets :

$s_1 = T_2$, $s_2 = T_4$, $s_3 = T_5$ et $s_4 = T_7$, qui caractérisent les quatre pyramides :

$P_1 = \{T_1, T_3, T_6\}$, $P_2 = \{T_1, T_6\}$, $P_3 = \{T_1\}$ et $P_4 = Ø$. D'où les 24 séquences dominantes.

Par l'application de l'algorithme de CARLIER sur cette exemple il s'agit d'un problème de 7 tâches à ordonnancer, nous donne une séquence optimale définie par

($T_6 < T_3 < T_2 < T_4 < T_1 < T_5 < T_7$), et parmi tous les nœuds de l'ensemble de séquences dominantes on prend le nœud racine de l'arborescence N_0 ayant $\max_{j\in T} L_j^{\min} = -2$ et $\max_{j\in T} L_j^{\max} = 11$.

Etape02

Grâce au théorème de dominance et la formule de la détermination des séquences dominantes

$$\left|S_{dom}\right| = \prod_{q=1}^{q=N} (1+q)^{n_q}$$

$\left|S_{dom}\right| = (1+1)^1.(2+1)^1(3+1)^1 = 24$. Par l'application du théorème précédent on trouve 24 séquences dominantes qui sont :

$T_6 < T_1 < T_3 < T_2 < T_4 < T_5 < T_7$ * $T_1 < T_3 < T_2 < T_6 < T_4 < T_5 < T_7$ * $T_6 < T_3 < T_2 < T_4 < T_6 < T_5 < T_7$
$T_6 < T_1 < T_2 < T_3 < T_4 < T_5 < T_7$ * $T_1 < T_2 < T_3 < T_6 < T_4 < T_5 < T_7$ * $T_1 < T_2 < T_2 < T_4 < T_6 < T_5 < T_7$
$T_6 < T_3 < T_2 < T_1 < T_4 < T_5 < T_7$ * $T_3 < T_2 < T_6 < T_1 < T_4 < T_5 < T_7$ * $T_3 < T_2 < T_1 < T_4 < T_6 < T_5 < T_7$
$T_6 < T_2 < T_3 < T_1 < T_4 < T_5 < T_7$ * $T_2 < T_3 < T_6 < T_1 < T_4 < T_5 < T_7$ * $T_2 < T_3 < T_1 < T_4 < T_6 < T_5 < T_7$
$T_6 < T_3 < T_2 < T_4 < T_1 < T_5 < T_7$ * $T_3 < T_2 < T_6 < T_4 < T_1 < T_5 < T_7$ * $T_3 < T_2 < T_4 < T_6 < T_1 < T_5 < T_7$
$T_6 < T_2 < T_3 < T_4 < T_1 < T_5 < T_7$ * $T_2 < T_3 < T_6 < T_4 < T_1 < T_5 < T_7$ * $T_2 < T_3 < T_4 < T_6 < T_1 < T_5 < T_7$
$T_6 < T_3 < T_2 < T_4 < T_5 < T_1 < T_7$ * $T_3 < T_2 < T_6 < T_4 < T_5 < T_1 < T_7$ * $T_3 < T_2 < T_4 < T_6 < T_5 < T_1 < T_7$
$T_6 < T_2 < T_3 < T_4 < T_5 < T_1 < T_7$ * $T_2 < T_3 < T_6 < T_4 < T_5 < T_1 < T_7$ * $T_2 < T_3 < T_4 < T_6 < T_5 < T_1 < T_7$

Tableau 9 : Les séquences dominantes de l'exemple 4.1

Etape03

Pour le Calcul de L^{min} et L^{max} de chaque tâche et pour la détermination de ces dernières appliquons les formules (5), (6) et (7) de BRIAND.

Pour la tâche T_4

- Le travail T_4 est un sommet il n'appartient donc qu'à la pyramide qu'il caractérise d'où $u(T_4)=2$ et l'ensemble $pred_j^{\min} = \{T_2, T_3, T_4\}$ qui sont des travaux placés avant la tâche T_4 dans toute séquence dominante et on a

$seq_4^{\min} = \{T_3 \prec T_2 \prec T_4\}$

$C_{\max}^{seq_4^{\min}} = \max(S_2 + p_2, S_3 + p_3, S_4 + p_4) \forall k \in pred_4^{\min}$

$= \max(19, 18, 24) = 24$

Finalement $L_4^{\min} = \max(C_{\max}^{seq_4^{\min}}, r_4) + p_4 - d_4$

$= \max(24, 20) + 4 - 30 = -2$

Pour la tâche T_5

Le travail T_5 est un sommet il n'appartient donc qu'à la pyramide qu'il caractérise d'où $u(T_5)=3$ et l'ensemble $pred_j^{\min} = \{T_2, T_3, T_4, T_6\}$ qui sont des travaux placés avant la tache T_5. $seq_5^{\min} = \{T_6 \prec T_3 \prec T_2, T_4\}$

$C_{\max}^{seq_5^{\min}} = \max(S_2 + p_2, S_3 + p_3, S_4 + p_4, S_6 + p_6) \forall k \in pred_5^{\min}$

$= \max(19, 18, 24, 06) = 24$

$L_5^{\min} = \max(C_{\max}^{seq_5^{\min}}, r_5) + p_5 - d_5$.

$= \max(24, 30) + 3 - 43 = -10$

Remarque

De la même manière calculons les autres bornes des autres tâches.

Pour les calcule L^{max}

$C_{\max}^{seq_j^{\max}} = \max(S_k^{seq_j^{\max}{}^n}) \forall k \in pred_j^{\max}$ avec $S_k^{seq_j^{\max}}$ sont les dates de débuts des travaux k dans l'ensemble $pred_j^{\max}$ et le $C_{\max}^{seq_j^{\max}}$ est le makespan correspondant.

Déterminer L^{max} revient alors à maximiser la durée totale C_{max} du problème d'ordonnancement constitué des travaux de l'ensemble $pred_j^{\min}$, pour minimiser le C_{max} une $seq_j^{\max}$ optimale est obtenue.

Par exemple la tâche T_6 :

Le travail T_6 appartient aux pyramides caractérisées par les sommets T_2 et T_4, d'où $v(T_6)=2$ L'ensemble $pred_j^{\max}$ qui contient des travaux k $v(k) \le v(T_6) = 2$, donc $pred_6^{\max} = \{T_1, T_2, T_3, T_4\}$.Déterminer le retard algébrique au pire revient à maximiser le $C_{\max}$ des travaux de l'ensemble $pred_6^{\max}$ .comme les deux travaux T_2 et T_3, $v(2) = v(3) = 1$ alors $S_1^{\max} = T_2 \prec T_3$ .Pour maximiser le $C_{\max}$ des travaux T_1 et T_4 $u(k) \le v(T_6) \le v(k)$ ou $v(T_6) = 2$ avec les travail T_6, d'où $S_2^{\max} = T_1 \prec T_4 \prec T_6$

A partir de $S_1^{\max}$ et $S_2^{\max}$ on a la séquence $T_2 \prec T_3 \prec T_1 \prec T_4 \prec T_6$, d'où $L_6^{\max} = 7$.

Après tous les calculs on trouve :

Résumé :

Tâches	T_1	T_2	T_3	T_4	T_5	T_6	T_7
r_i	10	13	11	20	30	0	30
p_i	5	6	7	4	3	6	2
d_i	44	25	27	30	43	34	51
$L_j^{\min}$	-29	-6	-9	-2	-10	-28	-14
$L_j^{\max}$	0	3	1	11	1	7	-5

Tableau 10: Résumé des calculs de deux bornes au pire et au mieux

Pour l'exemple 4.1 le diagramme de retard associé est l'intervalle fermé borné de la forme :

$-2 \leq L^*_{\max} \leq 11$ c.à.d. $L^*_{\max} \in [-2,11]$. Ces valeurs sont comparées au retard optimal $L^{opt}_{\max} = -1$ établi grâce à la méthode de Carlier.

5.6 Détermination d'un ordre partiel dominant et suffisant par une méthode de séparation et évaluation d'ordonnancement robuste [13]

La Hoing Trung.2005 présente une PSEP, il considère un problème déterministe dont l'objectif est de trouver un ordre partiel dominant et suffisant tel que le théorème de dominance caractérise des séquences dominantes optimales. [13]

5.6.1 Principe de séparation des nœuds de la première procédure La Hoing Trung.2005

Principe

Briand et al proposent un mécanisme de séparation qui nous permet de séparer un ensemble dominant de séquences en une bipartition sans créer de nouvelles séquences pour chaque nœud de l'arborescence, un chemin critique associé à la séquence produisant la borne supérieure du retard est déterminé. À partir de ce chemin qui détermine un pivot, correspondant à un sommet de ce chemin, et un travail i, non-sommet, que l'on décalera à droite ou à gauche du pivot et il distingue trois cas. Notons j le travail engendrant la borne supérieure.

- Si j est un travail non sommet, alors le pivot est le sommet de la pyramide à laquelle j appartient, le travail i choisi pour séparer est celui le plus proche du sommet.

- Si j est le sommet de la dernière pyramide du chemin critique, et si cette dernière pyramide contient d'autres travaux, alors le pivot à séparer est j et le travail i choisi est le premier travail situé à gauche de j.

- Si j est le sommet de la dernière pyramide, et si cette pyramide ne contient aucun autre travail, alors le pivot est le sommet de la première pyramide précédente rencontrée contenant encore des travaux non sommets ; le travail i choisi est le plus proche du sommet de cette pyramide.

Deux branchements sont envisagés : celui où la date de début de tache libre est actualisé par la date de début de la tache pivot et celui où la date de fin de tache libre est actualisé par la date de fin de la tache pivot. L'évaluation de chaque branchement est ensuite effectuée par le calcul des bornes inférieures et supérieures du retard. Le branchement possédant la meilleure borne supérieure est choisi pour la séparation suivante.

La séparation d'un nœud de l'arborescence s'arrête lorsque le chemin critique associé ne contient plus que des sommets ou lorsque les deux bornes sont égales.[13]

5.6.1.1 Améliorations évidentes

Notre amélioration basée sur l'étape de branchement des nœuds, le branchement adopté pour continuer la recherche est celui ou le nœud ayant meilleure borne inférieure, La séparation d'un nœud de l'arborescence s'arrête lorsque les deux bornes L^{min} et L^{max} sont égales. Nous remarquons que dans cette amélioration , la borne inférieure est évidemment peu efficace pour éliminer rapidement les nœuds de l'arborescence ,et donc d'améliorer les temps de calcul.

Remarque :

Une heuristique basée sur la détermination d'une borne supérieure au mieux qu'est inferieure à $L^{\max}$ et qui nous permet de couper plutôt l'arborescence de recherche et donc d'améliorer les temps de calcul.

Application

Traitons l'exemple 4.1 repris de l'article de CARLIER [29]

Tâches	T1	T2	T3	T4	T5	T6	T7
ri	10	13	11	20	30	0	30
pi	5	6	7	4	3	6	2
di	44	25	27	30	43	34	51
$L_j^{\min}$	-29	-6	-9	**-2**	-10	-28	-14
$L_j^{\max}$	0	3	1	**11**	1	7	-5

Tableau 11: Résumé de calculs de deux bornes au pire et au mieux de l'exemple 4.1

Première séparation :

$N_0 = \{T_2<T_3<T_6<T_1<T_4\}$, La racine N_0 est évaluée par : $L^{min} = -2$ et $L^{max} = 11$

Le chemin critique est : $T_2 < T_3<T_6<T_1 <T_4$ .Le travail pivot est le travail T_4 puisqu'il est un sommet de p_2 et ce dernière, il est contient des travaux non sommets, Le travail libre est le travailT_1 puisque le plus proche de T_4 à gauche.

Deux branchements sont effectués :

$N_1= \{T_2<T_3<T_6<T_4\}$ et $N_2= \{T_2<T_3<T_6<T_1<T_4\}$.

N_1 est évaluée par: $L^{min}= -2$ et $L^{max}= 6$ N_2 est évaluée par : $L^{min} = 2$ et $L^{max} = 11$

Comme la borne inférieure de N_2 est plus grande que celle de N_0 et de N_1, alors N_2 est coupé et le branchement sera effectué suivant N_1.

Deuxième séparation :

Le chemin critique est : $T_2<T_3<T_6<T_4$, Le travail pivot est le travail T_4, Le travail libre est le travail T_6.

Deux branchements sont effectués :

$N_3 = \{T_2<T_3<T_4<T_6\}$ et $N_4= \{T_2<T_3<T_6<T_4\}$.

N_3 est évaluée par $L^{min} = 0$ et $L^{max} = 2$ N_4 est évaluée par $L^{min} = -2$ et $L^{max} = 6$.

Comme la borne inférieure de N_3 est plus grande que celle de N_1 et de N_4 alors, N_3 est coupée et le branchement sera effectué suivant N_4.

Troisième séparation :

Le chemin critique est :{ $T_2<T_3<T_6<T_4\}$, Le travail pivot est le travail T_2, Le travail libre est le travail T_3.

Deux branchements sont effectués :

$N_5 = \{T_2<T_3<T_6<T_4\}$ et $N_6 = \{T_3<T_2<T_6<T_4\}$,

N_5 est évaluée par $L^{min} = 0$ et $L^{max} = 6$ N_6 est évaluée par $L^{min} = -1$ et $L^{max} = 4$

Comme la borne inférieure de N_5 est plus grande que celle de N_6 et de N_4 alors, N_5 est coupée et le branchement sera effectué suivant N_6

Quatrième séparation ;

Le chemin critique est :{ $T_3<T_2<T_6<T_4\}$, le travail pivot est le travail T_2, le travail libre est le travail T_6.

Deux branchements sont effectués :

$N_7 = \{T_3<T_2<T_6<T_4\}$ et $N_8 = \{T_6<T_3<T_2\}$.

N_7 est évaluée par $L^{min} = 4$ et $L^{max} = 4$ N_8 est évaluée par $L^{min} = -1$ et $L^{max} = -1$
D'ou la solution optimale (**$L^{min} = -1$ et $L^{max} = -1$**) , et une séquence optimale N_8 .
Les séquences optimales de l'ensemble dominant associé à ce problème sont donc toutes caractérisées par le nœud N_8. Le retard optimal est donc égal à –1. Le chemin critique est **$T_6<T_3<T_2$** et l'ensemble dominant de solutions associé à la structure d'intervalles obtenue pour le nœud N_8 est :
$T_6 < T_3 < T_2 < T_4 < (T_5 - T_1) < T_7$. Où $(T_5–T_1)$ est un groupe de travaux permutables.

Comparaison

Pour notre amélioration on a trouvé une séquence dominante et une solution optimale à partir de 4 itérations possibles par contre pour la procédure de TCOR la séquence dominante est une solution optimale trouvée à partir de 5 itérations possibles.

Illustration de L'arborescence de la première version améliorée :

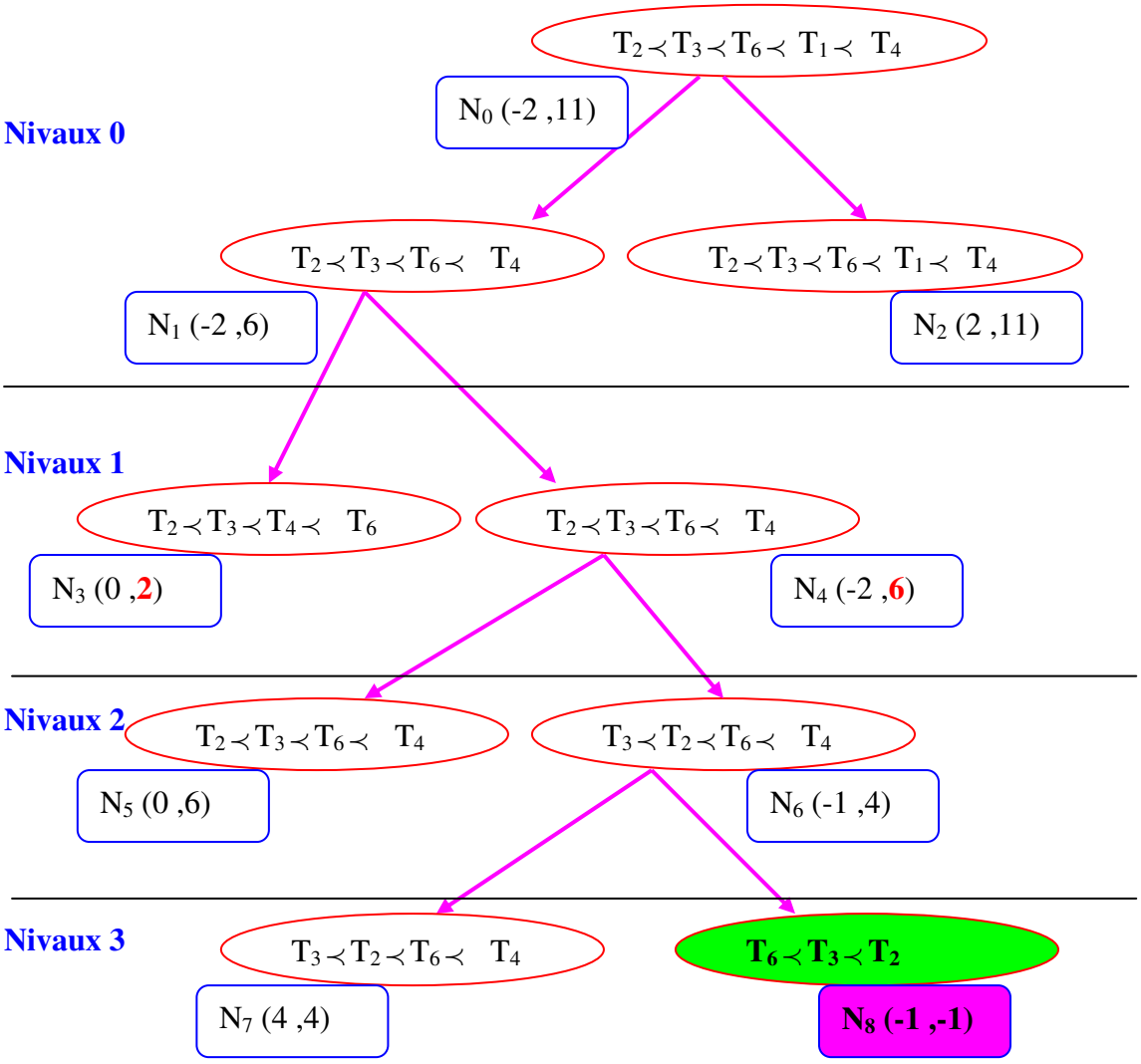


Figure17 : L'arborescence de la version améliorée de l'algorithme de TCOR1

5.6.1.2 Améliorations

Principe général :

Notre contribution basée cette fois sur l'amélioration de l'approche d'ordonnancement robuste précédente, dans le but est la détermination d'une solution dominante, robuste et flexible vis-à-vis du plus grand retard algébrique.

Dans ce cas, nous proposons une amélioration de l'approche d'ordonnancement robuste basée sur la PSEP de CARLIER.

Premièrement l'algorithme de CARLIER est appliqué pour la détermination d'une solution optimale et une séquence optimale, cette dernière est utilisée pour la détermination des travaux pivots et des travaux libres de la séquence. Deux bornes l'une au mieux et l'autre au pire sont calculés, ces dernières sont utilisées pour couper les nœuds de l'arborescence.

Pour l'évaluation de cette procédure on calcule toujours les deux bornes $\max_{j\in T} L_j^{\min}$ et $\max_{j\in T} L_j^{\max}$ (borne au mieux et au pire respectivement) de chaque tache de la séquence. Et parmi tous les nœuds on définit le nœud racine de l'arborescence comme étant le nœud ayant $\max_{j\in T} L_j^{\min}$ et $\max_{j\in T} L_j^{\max}$ parmi toutes les séquences déterminées par le théorème de dominance et pour chaque nœud de l'arborescence on détermine le chemin critique C qui donne le plus grand retard au pire $\max_{j\in T} L_j^{\max}$.

Première étape : séparation des nœuds

Concernant la séparation des nœuds on applique l'algorithme de calcul des travaux pivots et des travaux libres qui s'appelle algorithme setpivotlibre amélioré, dans les cas ou le travail pivot précède le travail libre nous actualisons les dates de début d'exécution de taches de la séquence, par contre dans le cas ou le travail libre précède le travail pivot nous actualisons les dates de fin d'exécution de tache de chaque séquence.

Ensuite nous calculons les deux bornes précédentes de chaque nœud. C.-à-d. pour chaque tache de la séquence on calcule les deux bornes et on choisit le nœud racine de l'arborescence ayant $\max_{j\in T} L_j^{\min}$ et $\max_{j\in T} L_j^{\max}$.

Première étape : branchement des nœuds

Le branchement adopté pour continuer la recherche après une séparation est celui ou le nœud ayant meilleures borne supérieures L^{max} c.à.d. nous comparons les retards au pire de chaque nœud de l'arborescence et le nœud adopté pour continuer la recherche est celui ayant la meilleure borne au pire de retard algébrique .D'autre part , Les nœuds sont coupés dés

que la borne supérieure de l'un des nœuds séparés est plus grande que l'autre, d'où les deux bornes supérieure et inférieure sont égales.

Cette stratégie nous donne une solution optimale robuste et flexible et une séquence optimale à partir de trois itérations possibles d'où la convergence rapide par apport à l'algorithme de TCOR1.

5.6.2 Application

Prenons l'exemple du tableau 10

Application sur la contribution

Appliquons tout d'abord la procédure de CARLIER pour l'obtention de la solution optimale $L_{\max}^{opt} = -1$, La séquence ayant $\max_{j \in T} L_j^{\max}$ est la suivante

$T_2 < T_3 < T_6 < T_1 < T_4$, avec $\max_{j \in T} L_j^{\max} = 11$.

Première séparation :

Pour le chemin critique N_0, le travail T_2 est choisi comme un travail pivot et le travail T_3 est choisi comme un travail libre. Donc deux nœuds se présentent N_1 et N_2 tels que :

$N_1 = \{T_2<T_3<T_6<T_1<T_4\}$, $N_2 = \{T_3<T_2<T_6<T_1<T_4\}$, calculons le retard au pire et au mieux de chaque nœud on trouve :

N_1 Est évaluée par $L^{min} = 0$ et $L^{max} = 11$.

N_2 est évaluée par : $L^{min} = -1$ et $L^{max} = 9$.

Le branchement est effectué sur N_2 car le nœud N_2 ayant une meilleure borne au pire. Alors, que le nœud N_1 est coupé.

Deuxième séparation :

Pour le chemin critique N_2 le travail T_3 est choisi comme un travail pivot et le travail T_6 est choisi comme un travail libre. Et donc deux nœuds se présentent N_3 et N_4 tels que :

$N_3 = \{T_3<T_2<T_6<T_1<T_4\}$, $N_4 = \{T_6<T_3<T_2<T_1<T_4\}$, calculons le retard au pire et au mieux de chaque nœud, on trouve :

N_3 Est évaluée par $L^{min} = -1$ et $L^{max} = 9$.

N_4 est évaluée par : $L^{min} = -1$ et $L^{max} = 3$.

Dans ce cas Le branchement est effectué sur le nœud N_4, car le nœud N_4 a une meilleure borne au pire. Alors, que le nœud N_3 est coupé.

Troisième séparation :

Pour le chemin critique N_4, le travail T_1est choisi comme un travail pivot et le travail T_4 est choisi comme un travail libre. Et donc deux nœuds se présentent N_5 et N_6 tels que :

$N_5 = \{T_6<T_3<T_2\}$, $N_6 = \{T_6<T_3<T_2<T_1<T_4\}$, calculons le retard au pire et au mieux de chaque nœud, on trouve :

N_5 est évalué par : $L^{min} = -1$ et $L^{max} = -1$.

N_6 est évaluée par : $L^{min} = 2$ et $L^{max} = 3$.

Dans ce cas le nœud **$N_5 = \{T_6<T_3<T_2\}$** est la solution optimale du problème avec **$L^{min} = -1$ et $L^{max} = -1$** de N_5 est le retard optimal associée, alors, que le N_6 est coupé.

On remarque que la séquence optimale et la solution optimale sont trouvées à partir de trois itérations, en utilisant seulement la solution optimale de CARLIER pour couper les nœuds.

On Peut continuer la séparation des nœuds restant mais on ne peut pas appliquer l'algorithme setpivotlibre amélioré. D'autre part il est clair que cette procédure converge plus rapidement vers la solution optimale et la séquence optimale après trois étapes, de plus on détermine le nombre de nœuds coupés au même temps.

Illustration de L'arborescence de la version améliorée :

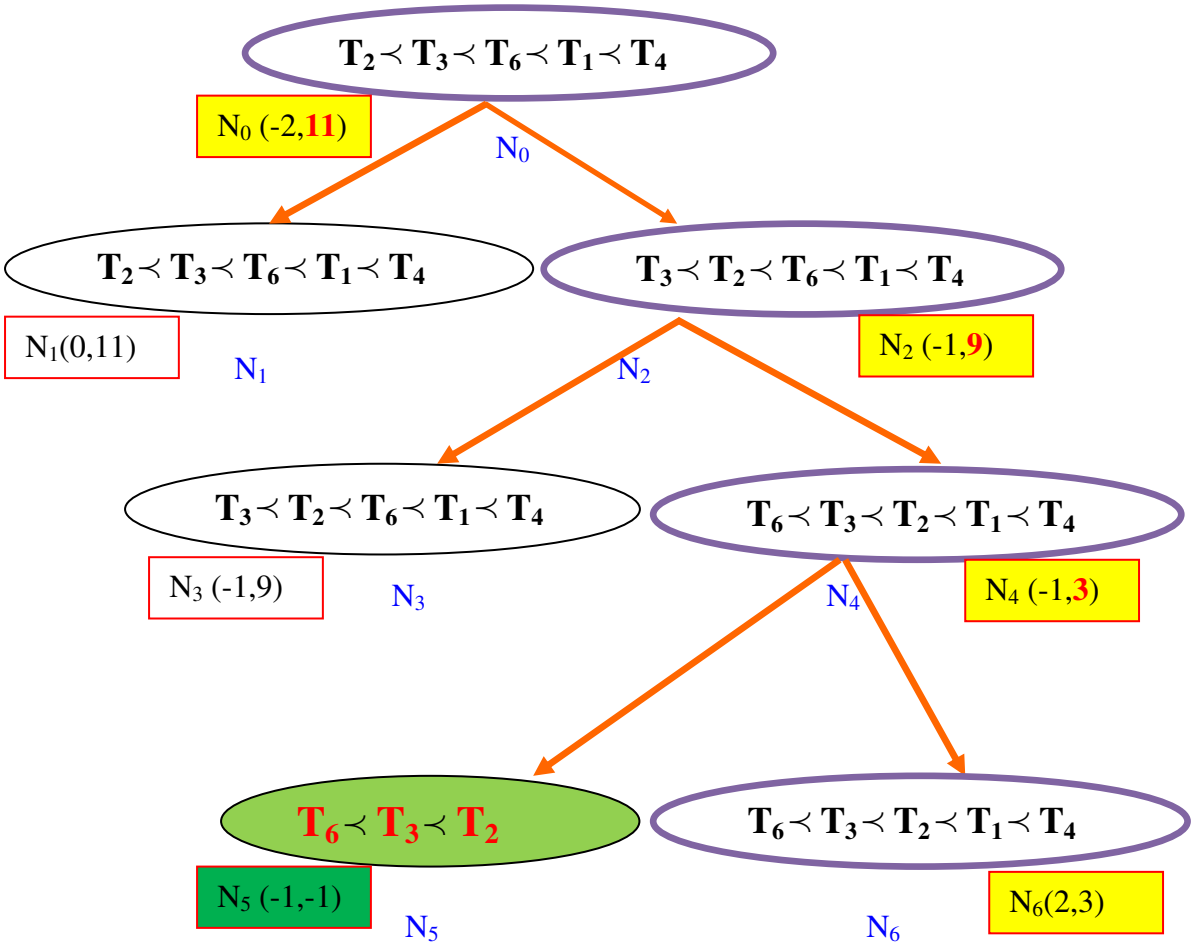


Figure 18: L'arborescence de la version améliore de l'algorithme de TCOR2

5.6.3 Comparaison entre les trois procédures

On Peut continuer la séparation des nœuds restant mais en peut pas appliquer l'algorithme setpivotlibre. D'autre part, il est clair que cette procédure converge plus rapidement vers la solution optimale et une séquence optimale après trois étapes, de plus on diminue le nombre des nœuds coupés au même temps sur l'arborescence.

Conclusion et perspective

Plusieurs études consacrées à l'ordonnancement ont montré l'écart important entre la recherche théorique et les besoins pratiques, la plupart des méthodes d'ordonnancement sont déterministes c.à.d. ne prennent pas le caractère incertain de leur environnement, pour cela une solution déterministe peut s'avérer rapidement. Pour régler ce problème, des méthodes d'ordonnancement robuste ont été développées, ces méthodes ayant une capacité de résister à la perturbation de l'environnement.

Dans ce mémoire, nous sommes focalisés sur l'étude du problème à une machine dans la littérature noté 1, ri, di/ L_{max} qu'est démontrer NP difficille par Lenstra dans les années 77, avec ri le temps de début d'exécution, di temps de fin d'exécution et L_{max} le plus grand retard algébrique, mais Carlier en 1982 à donné une procédure de séparation et d'évaluation progressive pour ce problème. En 2005 La Hoing Trung propose une version améliorée pour le problème 1, r_i, d_i/ L_{max} basée sur la solution et la séquence optimale de Carlier où cette dernière est utilisée pour calculer les travaux pivots et les travaux libres, ainsi cette procédure prend le nom d'algorithme de TCOR. Nous avons donné une observation sur le théorème de dominance démontré dans les années 80, cette observation basée sur les groupes de tâches permutables, un troisième cas limite de l'extension de ce théorème, basé sur le cas où les taches ayant même date de début et de fin d'exécution sont données, dans ce cas là toutes les séquences plaçant un travail entre deux sommets S_1 et S_2 de pyramides P_1et P_2 sont dominées par au moins une autre séquence de l'ensemble dominant.

Nous avons donné également une proposition sur les permutions des travaux dans l'ensemble de groupe tâche qui sont strictement équivalentes en terme de qualité c.à.d. robustesse.

Enfin, nous avons obtenu une version améliorée de l'algorithme de TCOR basé sur la détermination d'une solution flexible et robuste et une séquence dominante vis-à-vis du plus grand retard algébrique. Cette version basée surtout sur les étapes de séparation et dévaluation des nœuds de l'arborescence, dans le premier cas l'algorithme de Carlier est appliqué pour

la détermination d'une solution optimale et une séquence optimale, cette dernière est utilisée pour la détermination des tâches pivots et des tâches libres de la séquence , deux bornes des nœuds l'une au mieux et l'autre au pire sont calculées à partir de chaque séparation et chaque étape de l'arborescence.

Concernant le branchement des nœuds après une séparation on prend le nœud ayant meilleure borne supérieure L^{max} .les nœuds sont coupés dés que la borne supérieure de l'un des nœuds séparé est plus grand que l'autre. Pour un exemple de 07 taches, la version nous donne une solution robuste et flexible ainsi qu'une séquence optimale à partir de trois itérations possibles d'où la convergence rapide de la version par rapport à celle qui existe.

Les résultats obtenus dans ce mémoire, permettent d'envisager de nouvelles perspectives de recherche:

1. Détermination des nouveaux ordres partiels dominants basant sur des corps d'hypothèse restreint qui sont plus riches que c'elle existe.
2. Détermination des conditions de dominance nécessaires applicable a tout problème d ordonnancement.

Annexes

Vous trouvez ci-dessous les algorithmes et leur programmes élaborés en utilisant le langage C#.

Des algorithmes et leur programme par le C#

La fenêtre Principale du programme contient tous les calculs effectués (les sommets, les pyramides, les indices des pyramides, les deux retards c.à.d. le retard au mieux ainsi que le retard au pire)

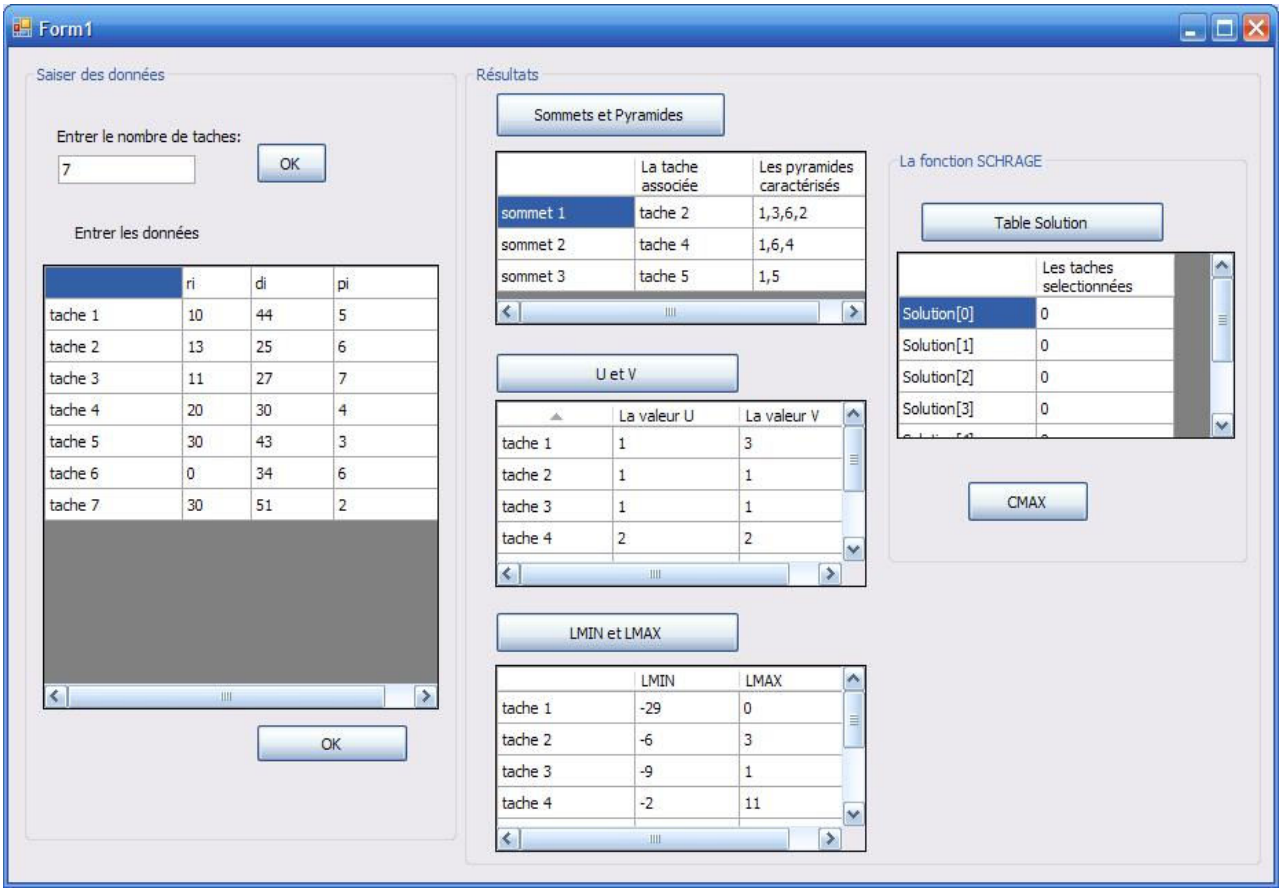


Figure 19 : Fenêtre principale du programme

Algorithme pour déterminer le travail pivot et le travail libre

Proc SetPivotLibre (travail_pivot, travail_libre, T)

Mettre le pointeur du chemin critique à la fin

Sommet _i ← dernier sommet sur le chemin critique ;

Tant Que la pyramide du sommet_i ne contient pas de travaux non sommets

sommet ← sommet précédent du sommet_i ;

Fin Tant Que

Travail_ pivot ← sommet _i ;

Si travail i à gauche du travail pivot n'est pas un sommet alors travail_libre ← ce travail i ;

Si Non

Chercher le travail i à droite du travail pivot, qui est le plus proche et qui n'est pas un sommet ;

Travail _libre ← travail i ;

Fin Si

Fin Proc.

Figure 20: Algorithme de la détermination des travaux libres et les travaux pivot

1. Le programme de sommet

```
void affecterSommet()
       {
            for (int i = 0; i < nbreTache; i++)
            {
                T[i].numTache = i;
                T[i].sommet = 1;
            }

            int somet = 0;
            int s = 0;
            for (int i = 0; i < nbreTache; i++)
            {
                s = 0;
                for (int j = 0; j < nbreTache; j++)
```

```
                    if (j != i)
                        if (T[j].r >= T[i].r && T[j].d <= T[i].d)
                        {
                            s = -1;
                            break;
                        }

                if (s >= 0)
                {
                    T[i].sommet = somet;
                    sommet[somet++] = i;
                }
                else
                    T[i].sommet = -1;

            }
            nbreSommet = somet;
            int temp = 0;
            for (int i = 0; i < nbreSommet - 1; i++)
            {
                int min = T[sommet[i]].r;
                for (int j = i + 1; j < nbreSommet; j++)
                {
                    if (min > T[sommet[j]].r)
                    {
                        temp = sommet[i];
                        sommet[i] = sommet[j];
                        sommet[j] = temp;
                    }
                     if (min == T[sommet[j]].r && T[sommet[i]].d <
            T[sommet[j]].d)
                    {
                        temp = sommet[i];
                        sommet[i] = sommet[j];
                        sommet[j] = temp;
                    }
                }
            }
        }
```

2. La fontion des pyramides

```
void affecterPyramide()
        {

            for (int i = 0; i < nbreSommet; i++)
            {
                int t1 = 0;
                pyramide[i] = new Pyramide();
                pyramide[i].ensTache = new int[200];
                for (int j = 0; j < nbreTache; j++)
                {
                    if (j != sommet[i])
                        if (T[j].r < T[sommet[i]].r && T[j].d >
T[sommet[i]].d)
                            pyramide[i].ensTache[t1++] = j;

                }
                pyramide[i].ensTache[t1++] = sommet[i];
                pyramide[i].nbreTache = t1;
            }
        }
```

Figure 21 : Fenêtre de la détermination des sommets et des pyramides

Cette fenêtre présente les tâches, les sommets et leurs pyramides caractérisées en utilisant le langage C#.

3. Programme de La déterminaion de l'indice de la 1ère pyramide et la 2ème

```
int u(int j)
        {
            int found = 0;
            for (int i = 0; i < nbreSommet && found == 0; i++)
                for (int k = 0; k < pyramide[i].nbreTache && found ==
0; k++)
                    if (j == pyramide[i].ensTache[k])
                    {
                        found = 1;
                        T[j].u = i;
                        return i;
                    }
            return -1;

        }
```

Le V

```
int v(int j)
        {
            int found = 0;
            for (int i = nbreSommet - 1; i >= 0 && found == 0; i--)
                for (int k = 0; k < pyramide[i].nbreTache && found ==
0; k++)
                    if (j == pyramide[i].ensTache[k])
                    {
                        found = 1;
                        T[j].v = i;
                        return i;
                    }
            return -1;
        }
```

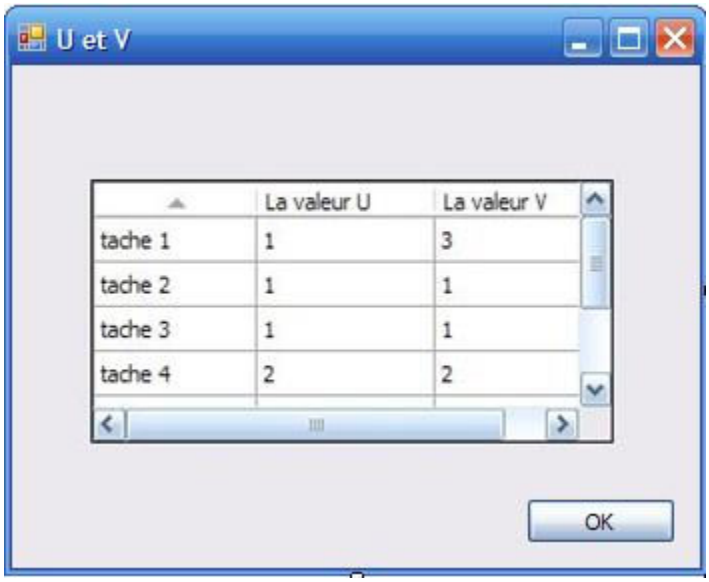


Figure 22 : Fenêtre de la détermination des indices de pyramides

Cette fenêtre présente les indices des pyramides caractérisé en utilisant le langage C#.

4. Programme de calcul L^{MIN} et L^{MAX}

```
int lmin(int j)
        {

            indice_pred[j] = 0;
            for (int i = 0; i < nbreTache; i++)
                if (v(i) < u(j)) predmin[j, indice_pred[j]++] = i;

            string ss="";
            for (int i = 0; i < indice_pred[j]; i++)
                ss = ss + "," + (predmin[j, i]+1);

            predmin[j, indice_pred[j]++] = j;

            int temp=0;
            for (int k = 0; k < indice_pred[j]; k++)
            {
                int min = T[predmin[j, k]].r;
                for (int i = k; i < indice_pred[j]; i++)

                    if (min > T[predmin[j, i]].r)
                    {
                        temp = predmin[j, k];
                        predmin[j, k] = predmin[j, i];
                        predmin[j, i] = temp;
                    }
            }
```

5. Programme de La fonction L^{MAX}

```
int lmax(int j)
        {
            indice_predx[j] = 0;
            for (int i = 0; i < nbreTache; i++)
            {
                if (u(i) <= v(j)) predmax[j, indice_predx[j]++] = i;
            }

            indice_predS1[j] = 0;
            indice_predS2[j] = 0;

            for (int i = 0; i < indice_predx[j]; i++)
            {
                if (v(i) < v(j)) predS1[j, indice_predS1[j]++] = i;
                if (u(i) <= v(j) && v(j) <= v(i)) predS2[j,
indice_predS2[j]++] = i;
            }
            int temp = 0;
            for (int k = 0; k < indice_predS1[j]; k++)
            {
                int min = T[predS1[j, k]].d;
                for (int i = k; i < indice_predS1[j]; i++)

                    if (min > T[predS1[j, i]].d)
                    {
                        temp = predS1[j, k];
                        predS1[j, k] = predS1[j, i];
                        predS1[j, i] = temp;
                    }
            }
            indice_A[j] = 0;
            indice_B[j] = 0;

            for (int i = 0; i < indice_predS2[j]; i++)
            {
                {
                    if (T[predS2[j, i]].d > T[j].d)
                        A[j, indice_A[j]++] = predS2[j, i];
                    else
                        B[j, indice_B[j]++] = predS2[j, i];
                }
            }

            int tempA = 0;
            for (int k = 0; k < indice_A[j]; k++)
            {
                int min = T[A[j,k]].r;
                for (int i = k; i < indice_A[j]; i++)

                    if (min > T[A[j,i]].r)
                    {
                        tempA = A[j,k];
                        A[j,k] = A[j,i];
                        A[j,i] = tempA;
                    }
            }

            int tempB = 0;
            for (int k = 0; k < indice_B[j]; k++)
            {
```

```
                int min = T[B[j,k]].d;
                for (int i = k; i < indice_B[j]; i++)

                    if (min > T[B[j,i]].d)
                    {
                        tempB = B[j,k];
                        B[j,k] = B[j,i];
                        B[j,i] = tempB;
                    }
            }
            for (int i = 0; i < indice_predS2[j]; i++)
            {
                if (i < indice_A[j])
                {
                    S2[j,i] = A[j,i];
                }
                else
                {
                    S2[j, i] = B[j, i - indice_A[j]];
                }
            }
            indice_seq_max[j] = indice_predS1[j] + indice_predS2[j];

            for (int i = 0; i < indice_seq_max[j]; i++)
            {
                if (i < indice_predS1[j])
                    seq_max[j,i] = predS1[j, i];
                else
                    seq_max[j, i] = predS2[j, i - indice_predS1[j]];
            }
            int c_seq_max = T[seq_max[j, 0]].r + T[seq_max[j, 0]].p;

            for (int i = 0; i < indice_seq_max[j]; i++)
                if (c_seq_max < T[seq_max[j,i]].r +
T[seq_max[j,i]].p)
                    c_seq_max = T[seq_max[j,i]].r +
T[seq_max[j,i]].p;

            int lmax = Math.Max(c_seq_max, T[j].r) + T[j].p - T[j].d;

            return lmax;

        }
```

Figure 23 : Fenêtre de la détermination des deux bornes L^{min} et L^{max}

Cette fenêtre présent les retards au mieux et au pire des travaux en utilisant le langage C#.

6. Programme de La fonction SCHRAGE

```
int SCHRAGE()
        {
            bool[] a_sequencer = new bool[nbreTache];
            bool SelectTache = false;
            int fin = 0;
            int debut = 0;


            for (int i = 0; i < nbreTache; i++)
                a_sequencer[i] = true;

            for (int i = 0; i < nbreTache; i++)
            {
                SelectTache = false;
                int min_rj = 0;
                for (int j = 0; j < nbreTache; j++)
                    if (a_sequencer[j]) min_rj = T[j].r;

                for (int j = 0; j < nbreTache; j++)
                    if (a_sequencer[j]) min_rj = Math.Min(min_rj,
T[j].r);

                debut = Math.Max(fin, min_rj);
                int k = -1;
                int tempMin = 0;
                for (int j = 0; j < nbreTache; j++)
                    if (a_sequencer[j] = true && T[j].r <= debut)
tempMin = debut - T[j].r;

                for (int j = 0; j < nbreTache; j++)
```

```
                    if (a_sequencer[j] = true && T[j].r<=debut)
                    {
                        if (tempMin > debut - T[j].r)
                        {
                            tempMin = debut - T[j].r;
                            k = j;
                            solution[i] = k;
                            SelectTache = true;
                        }
                    }

                if (SelectTache == true)
                {
                    nbreTacheSelectionne++;
                    a_sequencer[k] = false;
                    fin = debut + T[k].p;
                    cmax = Math.Max(cmax, fin + T[k].q);
                }
            }
            return 1;
}
```

Algorithme de TCOR

Proc TCOR(solution, $L_{\max}^{optimal}$, T)(*)

$L_{\max}^{optimal} \leftarrow +\infty$;
Solution $\leftarrow$ Ø ; / * solution est une liste de Sis optimales * /
A_traiter $\leftarrow$ Ø ; / * A_traiter est la liste de Sis non encore traitées * /
Calculer les bornes pour le nœud_racine ;
A_traiter $\leftarrow$ A_traiter $\cup$ nœud_racine ;

TantQue A_traiter $\neq$ Ø faire
Nœud_courant $\leftarrow$ dépiler A_traiter ;
Si $L^{\max} = L^{optimale}$ alors / * une Sioptimale locale * /
Si $L^{\min} \prec L_{\max}^{optimal}$ alors
$L_{\max}^{optimal} \leftarrow L^{\min}$;
Solution $\leftarrow$ Ø ;
Solution $\leftarrow$ Solution $\cup$ Nœud_courant ;
FinSi
Si $L^{\min} = L_{\max}^{optimale}$ alors
Solution $\leftarrow$ Solution $\cup$ Nœud_courant ;
FinSi
$L^{\min} \succ L_{\max}^{optimal}$ alors
Couper le Nœud_courant ;
FinS
SiNon
SetPivotLibre (Travail_Pivot,Travail_Libre, T) ;
Séparer Nœud_courant en deux fils :Fils_gauche et Fils_droit ;
Evaluer Fils_gauche et Fils_droit ;
/* Traiter Fils_gauche possédant la plus grand borne superieure */
Si $L_{Fils\_gauche}^{\min} \leq L_{\max}^{optimale}$ alors
A_traiter $\leftarrow$ A_traiter $\cup$ Fils_ gauche ;
SiNon
Couper Fils_gauche ;
FinSi
/* Traiter Fils_droit possédant la plus petite borne superieure */
Si $L_{Fils\_drot}^{\min} \leq L_{\max}^{optimale}$ alors
A_traiter $\leftarrow$ A_traiter $\cup$ Fils _droit ;
SiNon
Couper Fils_droite ;
FinSi
FinSi
Fin TanTantQue
FinProc

Figure 24 : Algorithme de TCOR

REFERENCES

1. Alexis Tsoukias "On the concept of decision aiding process. " Lamsade CNRS Université Paris Dauphine. June 2005.

2. Adam Kasperski ".Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion ".Wyspianskiego, 27, 50-370, Wroclaw .Poland. (431-436).2005.

3. Alein Berro, "optimisation multiobjectif et stratégies d'évolution en environnement dynamique", thèse de doctorat, université des sciences sociales Toulouse I, France .2001.

4. Abbas, M. et Vincke, PH. (1993), Preference structures and threshold models, Journal of Multicriteria Decision Analysis 2, 171–178.

5. Abbas, M. (1995), any complete preference structure without circuit admits an interval representation, Theory and Decision 39, 115–126.

6. A. Rossi. Ordonnancement en milieu incertain, mise en œuvre d'une démarche robuste. Thèse de Doctorat, Institut National Polytechnique de Grenoble, France, 2002.

7. B. Roy. Robustesse de quoi et vis-à-vis de quoi mais aussi robustesse pourquoi en aide à la décision ? Newsletter of the European Working Group - Multicriteria Aid for Decisions, vol. 3, no. 6, pages 1 6,2002.

8.Cstaitin-Zopounidis et Michael Douspos "Newsletter of European working group" (multicriteria aid for decision) .series 3, n= 10 Fall 2004. Technical university of Crete, dept .of production engineering and management financial engineering.

09. Christina Vassiadou, Zeniou. "Robust optimization models for managing callable bond portfolio ".Cyrus. (264-273) .1996.

10. Clarisse Dhaenens-Flipo, "Optimisation Combinatoire Multi Objectif : Apport des Méthodes Coopératives et Contribution à l'Extraction de Connaissances", thèse de Habilitation à Diriger des Recherches de l'U.S.T.L, Laboratoire d'Informatique Fondamentale de Lille, 2005.

11. C. Briand & H.T. La. "Une procédure par séparation et évaluation progressive pour l'ordonnancement robuste de problèmes à une machine". Dans Recherche Informatique Vietnam & Francophonie 2003 (RIVF2003), pages 11_16, Hanoi, Vietnam, 2003.

12. C. Briand, H.T. La & J. Erschler. "Ordonnancement de problèmes à une machine : une aide à la décision pour un compromis flexibilité performance".

Dans 5ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'03), pages 146_147, Avignon, France, 2003.

13. C. Briand, H.T. La & J. Erschler. "Une approche pour l'ordonnancement robuste de tâches sur une machine". Dans 4ème Conférence Francophone de modélisation et simulation (MOSIM'03), pages205, 211, Toulouse, France, 2003.

14. C. Esswein, J.-C. Billaut & C. Artigues. "Une approche multicritères pour un apport de flexibilité séquentielle". Dans Flexibilité et robustesse en ordonnancement sous la direction de J-C. Billaut et A.Moukrim et E. Sanlaville, pages 209_232. Hermès, 2005.

15. C. Esswein. "Un apport de flexibilité séquentielle pour l'ordonnancement Robuste". Thèse de Doctorat, Université François Rabelais, Tours, France, Décembre 2003.

16. Dimitris . Bertsimas, Dessislava Pachamanova ."Robust linear optimisation Under général noms. Combridge, MA 02142-1347-USA.(510-516).2004.

17. Frédéric Gardi, "Ordonnancement avec exclusion mutuelle par un graphe d'intervalles ou d'une classe apparentée : complexité et algorithmes", THÈSE De doctorat, École Doctorale de Mathématiques et Informatique de Marseille, 2005.

18. Freerk A. Lootsma . "MultiCriteria Decision Analysis via Ratio and Difference Judgement. Delft University of technology. Delft, The Netherlands.

19. George j. Kyparisist et Christos Koulamas "Flexible flow shop scheduling with .uniform parallel machins .MIAMI fl 331997. (985-997) .2006.

20. Guillaume Pinot Nasser Mebarki, "coperation homme machine pour la mise œuvre d'un ordonnancement de groupe". CNRS 6597 Nantes, France FRANCORO V / ROADEF 2007.

21. GOThA. "Flexibilité et Robustesse en Ordonnancement". Le bulletin de la ROADEF, vol. 8, pages 10,12, 2002.

22. HERVé MEUNIER " algorithmes évolutionnaires parallèles pour l'optimisation multéobjéctif de réseaux de télécommunication mobiles ". Thèse de doctorat. Université des sciences technologies de Lille. U.F.R D' I.E.E.A 2002.

23. H.T. La, J.L. Santamaria & C. Briand. "Une aide à la décision pour L'ordonnancement robuste à une machine : un compromis flexibilité /performance, à paraître". Dans 6ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'05), Tours, France, 2005.

24. Igor Averbakh, Oded Bersman . "Algorithms for the robust 1- center problem on a Tree. Wa 98225-9063. USA. (292-302). 2000.

25. Julien Cegarra ." la gestion de la complexité dans la planification : le cas de l'ordonnancement ". Thèse de doctorat, Université Paris 8 –Vincennes-saint –Denis.octobre 2004.

26. Juid Wang."A fuzzy robust scheduling approach for product development project .Taiwan roc. (180-194). (2004).

27. Jacques Pictet. "Newsletter of european working group" (multicriteria aid for decisions) .Forum. Robustness analysis. Robustness analysis in practice. Series 3, n=14. Fall 2006.

28. Jens Clausen. "Branch and Bound Algorithms Principles and Examples". March 12, 1999.

29. J. Carlier. "The one machine sequencing problem". European Journal Of Operational Research, vol. 11, pages 42_47, 1982.

30. J. Erschler & 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.

31 . J. Erschler, G. Fontan & C. Merce. "Un nouveau concept de dominance pour l'ordonnancement des travaux sur une machine". RAIRO Recherche Opérationnelle/Opérations Research, vol. 19, no. 1, pages1913, 1985.

32. J. Yang & G. Yu. On the robust single machine scheduling problem. Journal of Combinatorial Optimization, vol. 6, no. 2, pages 18 33, 2002.

33. Lionel Eyraud." Théorie et pratique de l'ordonnancement d'applications sur les systèmes distributes. Thèse de doctorat, l'École Doctorale de Mathématiques, Sciences et Technologies de l'Information, Informatique, laboratoire Informatique et Distribution.2006.

34. Kebabla. Mebarek. Utilisation des stratégies métaheuristique pour l'ordonnancement des ateliers de type job shop. Université de Batna 2008.

35. M.Aaloulou.R. kalai,D.Vanderpooten. R. Kalai, D. Vanderpooten «newsletter of european working group " (multicriteria aid for decisions) une nouvelle approche de robustesse α-lexcicographique. Series 3, n=12 Fall 2005. LAMSADE université Paris dauphine.

36. Mohamed Ali Aloulou et Federico Della Croce, "Complexity of single machine scheduling problems under scenario-based uncertainty" , LAMSADE, Université Paris Dauphine, France,2007.

37. Matthieu DUPUY. "Contributions à l'analyse des systèmes industriels et aux problèmes d'ordonnancement à machines parallèles flexibles application aux laboratoires de contrôle qualité en industrie pharmaceutique". Thèse de doctorat .2005 Centre de Génie Industriel de l'École des Mines d'Albi-Carmaux.

38. Philippe Vincke. Forum ." About robustness analysis ". SMG. Université de Lille de Bruxelles.

39. Pawel Zielinski. "The computational complexity of the relative robust shortest path problem with interval data .polond (570-576).2004.

40. Pl Vincke. "Robust and neutral methodes for aggregation performance into an outranking relation .Brussels Belgium. (405-412) .1999.

4**1.** Philippe Baptiste , Laurent Peridy Eric Pinson , "A Branch and Bound to Minimize the Number of Late Jobs on a Single Machine with Release Time Constraints " , CNRS, UMR 6599 HEUDIASYC, Univ. de Tech. de Compiègne, F-60200 Compiègne , 2001 .

42. R. Montermanni et L.Mogambardella "a branch and bound algorithm for the robust spannibg tree. Problem witheinterval data
Swizzer land. (771-779). 2005.

43. R.Montemanni, L, M.Gambardella Av, Donali."A branch and bound algorithm for the robust shortest path problem with interval data". Galleria 2, CH-6928 Manno, Switzerland. (225-232) .2004.

44. Samia Ourari1, Cyril Briand2 et Brahim Bou, "Une Condition de Dominance pour l'Ordonnancement à une Machine avec Minimisation du Nombre de Travaux en Retard" , 1 CDTA d'Alger, Lot. du 20 août 1956, Baba Hassen, Alger, Algérie, LAAS-CNRS, 7 av. du Colonel Roche 31077, Toulouse, France.

45. S.F. Smith. "Intelligent scheduling systems, chapter Reactive scheduling Systems". Kluwer Press, 1995.

46. T. Vidal et al, "Gestion de projets sous incertitudes : un modèle de génération de plans flexibles en horizon glissant", LGP – ENI de Tarbes, 47, avenue d'Azereix, B.P.1629, 65016 Tarbes Cedex,

47. V.J. Leon, S.D. Wu & R.H. Storer. "Robustness measures and robust Scheduling for job shops". IIE Transactions, vol. 26, no. 5, pages 32 43, 1994.

48. Vincke, P.h "Problème multicitères". Cahiers du Centre d'Etudes de Recherche Opérationnelle, 16, p.425-439, 1974.

49. Yunpeng Pan, Leyuan Shi." Branch-and-bound algorithms for solving hard instances of the one-machine sequencing problem. Department of Industrial Engineering, University of Wisconsin, Madison, WI 53706, USA.2004.