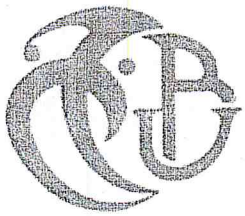
**RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**




**UNIVERSITÉ SAAD DAHLAB BLIDA**

**Faculté des Sciences**

**Département de Mathématiques**


---

**Mémoire de Fin D'études**
**en vue de l'obtention du Master.**

**Option : Recherche Opérationnelle.**

# *Étude d'un problème d'ordonnancement à une machine avec fenêtre d'exécution et poids associés à certains travaux : Cas de minimisation de $\Sigma U_i$*


**Présenté par :**

Mlle. AISSIOU Amira

Mlle. MADANI Hanane

**Encadré par :**

Mlle. S. OURARI (CDTA)

Mr. M.N. RAOUTI (USDB)


**Devant le Jury :**

| | | |
|---|---|---|
| Mr O.TAMI | M.A.A, USD de Blida | Président |
| Mr A .DERBALA | M .C, USD de Blida | Examinateur |
| Mr M.AIT AKKACHE | M.A.A , USD de Blida | Examinateur |
| Mlle S. OURARI | M.R.B, CDTA | Rapporteur |
| Mr. M.N. RAOUTI | M.A.A, USD de Blida | Rapporteur |

**Promotion 2011/2012**

## Remerciements

*Le travail qu'on a le plaisir de vous présenter a été réalisé grâce aux personnes que nous avons l'honneur de vous présenter et de remercier:*

*A nos encadreurs* ***MLLE Samia OURARI*** *et* ***Mr Mohamed Nazih RAOUTI*** *pour leurs enthousiasme et créativité.*

*Aux membres du jury Messieurs* ***Omar TAMI, Ali DERBALA, Mustapha AIT AKKACHE*** *pour avoir accepté de participer.*

*Aux professeurs qui nous ont encadrés pour leurs conseils, leurs orientations et leur disponibilité qui nous ont été d'un apport inestimable.*

*Au soutient et encouragement de nos familles et amies.*

*Aux membres du CDTA et du personnel du département mathématiques de l'université SAAD DAHLAB de Blida.*

*A notre camarade* ***Meriem BENYOUCEF*** *pour sa précieuse contrebutions dans la préparation de ce mémoire.*

*A nos camarades étudiants qui ont crée une ambiance propre*

*A la persévérance dans la quête du savoir ...*

*Je leurs présente mes sincères remerciements.*

*Je dédie ce mémoire :*

*A mes parents qui ont éclairé mon chemin et qui m'ont encouragé et soutenue toute au long de mes études.*

*A mes sœurs.*

*A mon frère.*

*A tous mes ami(e)s.*

AMIRA

*Je dédie ce mémoire :*

*A mes parents qui ont éclairé mon chemin et qui m'ont encouragé et soutenue toute au long de mes études.*

*A mes sœurs.*

*A mes frères.*

*A mes belles-sœurs.*

*A mes beaux-frères.*

*A mes nièces et neveux.*

*A tous mes ami(e)s.*

HANANE

# Table des Matières

**Introduction générale** **1**

**1 Introduction aux problèmes d'ordonnancement** **4**

1.1 Système de gestion de production . . . . . 4

1.1.1 Les différentes structures d'un système de gestion de la production . . 6

1.2 Présentation des problèmes d'ordonnancement . . . . . 10

1.2.1 Définition . . . . . 10

1.2.2 Éléments fondamentaux d'un problème d'ordonnancement . . . . . 11

1.2.3 Classification des problèmes d'ordonnancement . . . . . 14

1.3 Approches classiques de résolution . . . . . 17

1.3.1 Les méthodes optimales: . . . . . 18

1.3.2 Les méthodes optimales énumératives: . . . . . 18

1.3.3 Les méthodes heuristiques: . . . . . 19

**2 Problème d'ordonnancement à une machine et théorème des pyramides** **21**

2.1 Etat de l'art . . . . . 21

2.1.1 Hypothèses du problème . . . . . 22

2.1.2 Cas particuliers du problème $(1\ |r_i|\ \sum U_i)$ . . . . . 23

2.1.3 Cas général du problème $(1\ |r_i|\ \sum U_i)$ . . . . . 26

2.2 Structure d'intervalle et dominance . . . . . 27

2.2.1 Structure d'intervalle et algebre d'Allen . . . . . 27
2.2.2 Notions de sommet et pyramide . . . . . 28
2.2.3 Notion de dominance en ordonnancement . . . . . 29
2.2.4 Théorème des pyramides . . . . . 30
2.2.5 Exemple . . . . . 31
2.2.6 Notion de séquence maître-pyramide . . . . . 34
2.3 Condition de dominance pour la minimisation de $\sum U_i$ . . . . . 35
2.3.1 Théorème des pyramides et minimisation de $\sum U_i$ . . . . . 36

**3 Rappel sur la modélisation mathématique du pb 1 $|r_i| \sum U_i$** **38**
3.1 Problème d'admissibilité . . . . . 38
3.1.1 Problème mono-pyramidal . . . . . 38
3.1.2 Une condition nécessaire et suffisante d'admissibilité . . . . . 39
3.1.3 une condition numérique de dominance . . . . . 39
3.1.4 Un modèle de PLNE pour le problème multi-pyramidal interdépendant 41
3.1.5 Un modèle de PLNE pour le problème multi-pyramidal quelconque . 43
3.2 Problème de minimisation du nombre de travaux en retard . . . . . 44
3.2.1 Utilisation de la séquence maître-pyramide pour la modélisation . . . 45

**4 Nouvelle approche de la modélisation mathématique** **52**
4.1 Modèle PLNE pour le cas général(formule améliorée) . . . . . 52
4.2 Minimisation du nombre de travaux en retard . . . . . 54
4.2.1 Modèle PLNE pour l'obtention d'une borne supérieure . . . . . 54
4.2.2 Modèle PLNE pour l'obtention d'une borne inférieure . . . . . 55
4.3 Modèle avec poids associés aux variables sommets et non-sommets. . . . . 59

**5 Implémentation et Résolution** **61**

5.1 Présentation du ILOG CPLEX . . . . . 61
5.2 Implémentation . . . . . 62
5.3 Résultats des expérimentations . . . . . 68
5.3.1 Cas des modèles sans poids . . . . . 68
5.3.2 Cas des modèles avec poids . . . . . 72

**Conclusion et perspectives** **79**

**Bibliographie** **81**

# Liste des Illustrations graphiques


1.1 Organisation hiérarchique d'un système de gestion. . . . . 5

1.2 Structure générale d'un système de gestion de la production . . . . . 7

1.3 Système centralisé de gestion . . . . . 9

1.4 système décentralisé de gestion. . . . . 9

1.5 Les caractéristiques d'une tâche. . . . . 12

2.1 Algorithme de Moore . . . . . 25

2.2 Algorithme de Lawler . . . . . 26

2.3 Algèbre d'Allen . . . . . 27

2.4 Diagramme des intervalles, sommets et pyramides. . . . . 29

2.5 Diagramme des intervalles, sommets et pyramides . . . . . 32

2.6 Ensemble dominant de séquences du problème de la figure 2.5 . . . . . 33

2.7 Structure des séquences dominantes . . . . . 35

3.1 Algorithme de relaxation des dates de disponibilité. . . . . 50

3.2 Algorithme de relaxation des dates d'échéance. . . . . 51

4.1 Exemple illustratif du changement de variable . . . . . 53

4.2 Structure d'une instance de donnée à traiter. . . . . 57

5.1 Structure du fichier résultat. . . . . 64

5.2 Structure des tâches non sommets dans le fichier résultat. . . . . 65
5.3 Structure des sommets dans le fichier résultat. . . . . 66
5.4 Affectation des tâches selon la séquence dominante dans le fichier résultat. . 67

# Liste des Tables


1.1 Quelques valeurs du champ $\alpha$ . . . . . 15

1.2 Quelques valeurs du champ $\beta$ . . . . . 16

1.3 Quelques valeurs du champ $\gamma$ . . . . . 17

2.1 Une instance d'un problème à une machine . . . . . 32

5.1 Comparaison des résultats(Borne) pour un échantillon. . . . . 70

5.2 Pourcentage des instances résolues pour les bornes supérieures. . . . . 71

5.3 Pourcentage des instances résolues pour les bornes inférieures. . . . . 71

5.4 Pourcentage des instances résolues optimalement. . . . . 72

5.5 comparaisons de 80 et 100 tâches pour 10 instances . . . . . 73

# Introduction générale

Dans le contexte actuel d'évolution des entreprises à l'étranger, il faut disposer de méthodes et d'outils de plus en plus performants pour l'organisation et la conduite de la production. Cette organisation de la production doit être vue au niveau de l'entreprise elle-même, mais aussi au niveau de la chaîne logistique dont elle constitue l'un des maillons. Pour atteindre ses objectifs, l'organisation repose en général sur la mise en œuvre d'un certain nombre de fonctions assurées par des méthodes sophistiquées d'autant plus que les systèmes d'information sont de plus en plus fiables et dynamiques. Parmi ces fonctions, l'ordonnancement joue un rôle essentiel dans la gestion de l'entreprise.

La fonction ordonnancement vise à organiser l'utilisation des ressources technologiques et humaines présentes dans les ateliers ou les services de l'entreprise pour satisfaire soit directement les demandes des clients, ou les demandes issues d'un plan de production préparé par la fonction de planification de l'entreprise. Compte tenu de l'évolution des marchés et de leurs exigences, cette fonction doit organiser l'exécution simultanée de multiples travaux sur des délais de réalisation de plus en plus courts, à l'aide de ressources plus ou moins polyvalentes disponibles en quantités limitées. Ceci constitue un problème complexe à résoudre, et le recours à des systèmes informatiques pour la programmation de la production devient une nécessité. En cela, apporter des solutions efficaces et performantes aux problèmes d'ordonnancement constitue surement un enjeu économique important.

Notre travail rentre dans le cadre des travaux de recherche réalisés au sein de la Division Robotique et Productique du Centre de Développement des Technologies Avancées, et plus précisément dans un projet qui s'intitule : Ordonnancement et Pilotage Distribué des Systèmes de Production Contraints.

Dans l'étude menée dans ce mémoire, nous nous intéressons à un problème d'ordonnancement à une ressource. Ce problème est fondamentalement important car très souvent rencontré dans la pratique. En effet, il existe des situations où un problème d'ordonnancement de type job-shop (ou les ateliers à cheminement multiples :chaque travail à sa propre gamme opératoire) peut être équivalent au problème à une seule ressource lorsqu'un poste de l'atelier constitue un goulot d'étranglement. Ceci d'une part, d'autre part et du point de vue théorique, le problème étant difficile, beaucoup de travaux s'intéressent à sa résolution.

Le problème à une ressource étudié dans ce mémoire, considère un ensemble de travaux à réaliser sur des fenêtres d'exécution connues et avec comme critère considéré, la minimisation du nombre de travaux en retard. Ce problème étant NP-difficile [29], plusieurs travaux antérieurs ont été réalisés en se basant sur des méthodes exactes de résolution. Dans notre travail, nous avons repris des travaux basés sur le concept de dominance pour proposer un nouveau modèle mathématique utilisant la programmation linéaire en nombres entiers, à travers de nouvelles formulations mathématiques, puis afin de valider l'approche à travers des expérimentations nous avons implémenté ces modèles sur un solveur.

Ce mémoire est organisé en 5 chapitres, comme suit :

Le premier chapitre présente le contexte dans lequel s'inscrit notre travail. Un aperçu est donné concernant quelques notions élémentaires sur le système de gestion et de production, ainsi que l'ordonnancement avec fenêtre d'exécution et sur leurs classifications. Puis nous donnons un aperçu des approches classiques de résolution des problèmes d'ordonnancements.

Le deuxième chapitre est consacré à des notions nécessaires à la compréhension des modèles mathématiques présentés dans ce mémoire et qui sont en relation avec le concept de dominance, particulièrement, les structures d'intervalles, les concepts de sommet et pyramide qui en découlent ainsi que le théorème de dominance, communément appelé théorème des pyramides. La notion de séquence maître pyramide qui est à la base des formulations mathématiques donnée y'est rappelée.

Dans le troisième chapitre, nous présentons les modèles de programmation linéaire en nombres entiers permettant la formulation mathématique pour la résolution du problème de minimisation du nombre de travaux en retard auquel on s'intéresse. Ces modèles permettent de limiter la solution optimale, quand elle n'est pas atteinte, par des bornes supérieures et inférieures de notre problème $1|r_i|\sum U_i$ en se basant sur des conditions de dominance. Nous y considérons progressivement des problèmes particuliers, jusqu'à arriver à une formulation générale.

Dans le quatrième chapitre nous donnons une nouvelle formulation mathématique plus améliorée en termes du nombre de variables binaires considérées. Ce changement de variable est à l'origine d'une nouvelle formulation et la proposition de nouvelles bornes inférieures et supérieures, toujours en se basant sur les conditions de dominance. Puis, afin d'améliorer

encore plus les bornes, une notion de poids est introduite pour les tâches constituant les sommets de la pyramide, afin d'étudier leur impact sur la solution en termes de valeur du critère et le temps d'exécution.

Enfin, un dernier chapitre est consacré à la partie implémentation de ces modèles sur un solveur. Un exemple simple est donné avant de présenter les expérimentations réalisées sur des instances de la littérature, et ce dans le but d'évaluer l'approche, en considérant à la fois le critère avec et sans le poids.

# Chapitre 1

# Introduction aux problèmes d'ordonnancement

Ce chapitre constitue des notions de base sur les systèmes de gestion et la fonction ordonnancement. Il rappelle aussi les principaux problèmes d'ordonnancement, et donne un aperçu de quelques approches de résolution standards traitées dans la littérature.

## 1.1 Système de gestion de production

La production est une activité opérationnelle utilisant des ressources acquises (matériels & humains) en vue de créer des biens matériels ou d'assurer des services d'utilité moyennant une ou plusieurs transformations. Ces transformations sont caractérisées par un ensemble d'opérations, qui peuvent consister, soit dans une modification de la forme des pièces, soit dans la combinaison de plusieurs pièces. L'ensemble de ces transformations constituant la production seront soumises à des critères impératifs de coût, délai, quantité et qualité [7].

D'une manière classique, tout système de production est décomposé selon une approche systématique en plusieurs sous-systèmes afin de faciliter la maîtrise de la complexité associée à la fonction production. Dans la littérature, Doumeinghts et al [18] proposent une décomposition structurée en fonction de la nature des flux qui la traversent : Système physique (flux de matières, humaines, énergie, etc.), système de décision permettant le pilotage du système physique (flux de décisions), et le système d'information représentant l'ensemble des

informations en stock ou en circulation dans le système (flux d'informations).

-Le système physique de production est constitué par l'ensemble des ressources (humaines et matérielles) permet de convertir des matières premières en produits finis.

-Le système de pilotage est destiné à commander le système physique de production : Des données récupérées par le sous-système informationnel sont traitées par le sous-système décisionnel afin de définir les décisions qui garantissent le bon fonctionnement du système physique compte tenu des multiples contraintes internes (ressources disponibles) ou externes (contraintes des clients et fournisseurs) à l'entreprise.

Le rôle de tout système de gestion est de piloter le système de production à travers des décisions. Ces décisions sont de plusieurs types, et sont structurées hiérarchiquement (figure 1.1) et selon l'horizon sur lequel elles s'appliquent en trois niveaux (typologie introduite par Antony [3]). Chaque niveau de la hiérarchie établit un plan pour des entités bien définies, la décision d'un niveau donné devient une contrainte pour le niveau inférieur :

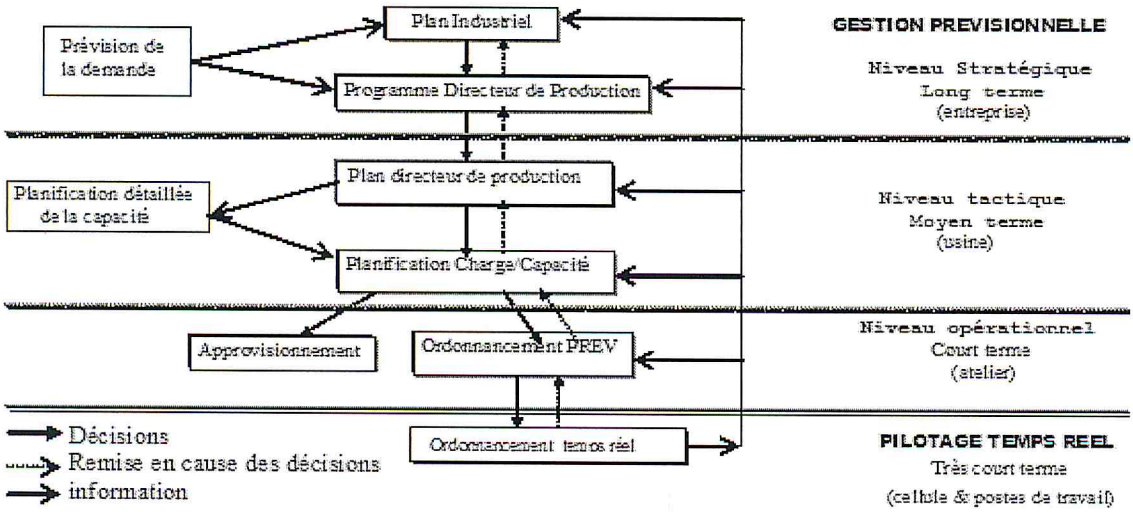



Figure 1.1: Organisation hiérarchique d'un système de gestion.

-Niveau stratégique : Les décisions élaborées à ce niveau définissent la politique de l'entreprise à long terme et concerne généralement les décisions d'investissement ou d'embauche.

-Niveau tactique : Les décisions à moyen terme prises à ce niveau concernent la production et visent l'utilisation efficace des moyens définis au niveau supérieur. Elles se traduisent par la formulation d'un plan directeur de production qui spécifie quels produits fabriquer, en quelles quantités et pour quelles dates. Ce problème est complexe et nécessite généralement une décomposition en niveaux de décision.

-Niveau opérationnel : Ce niveau assure le fonctionnement quotidien de la fabrication et intègre deux types de décisions : Les décisions à court terme élaborées à partir des données statiques. Elles permettent une affectation des ressources aux différentes tâches avec une répartition des tâches dans le temps; et les décisions à temps réel prises à partir des données dynamiques assurent la conduite du système physique quand ce dernier est soumis à des perturbations internes (pannes des machines, etc), ou externes (perturbations de la demande, manque d'approvisionnement, etc). Ces décisions concernent la fonction l'ordonnancement qui sera présentée en détail dans la suite de ce chapitre.

### 1.1.1 Les différentes structures d'un système de gestion de la production

Le système de gestion est destiné à commander le système de production, en d'autre termes, il doit déterminer les activités que devront effectuer les ressources du système de production.

#### A.Structure générale d'un système de gestion de la production

Un système de gestion de la production se compose généralement de quatre sous systèmes.

•**Le sous système de pilotage :** Est destiné à faire fonctionner d'une manière efficace (Commander où Piloter ) le système opérationnel. En d'autre termes, il doit déterminer les activités que devront effectuer les ressources du système opérationnel. Vue la dynamique

du processus de production, les commandes sortantent du système de pilotage se présentent sous forme d'une suite de commande encore appeler «Contrôle».

•**Le sous système hiérarchique de décision :** Est constitué de personnes ou unités administratives à qui l'on a confié les activités de pilotage. Ce sont des unités émettrices de commandes appelées aussi entités décisionnelles.

•**Le sous système opérationnel :** Comprend l'ensemble des ressources destinées à acquérir des intrants, leurs faire subir des transformations pour avoir des extrants utiles.

•**Le sous système d'information :** Relie les sous-systèmes decisionnel et operationel entre eux. Il est destiné à faire transmettre les décisions et les politiques de conduite en temps de commande au système opérationnel pour l'exécution et à communiquer les informations sur l'état du système au moment de la décision (Feed Back).

Il apparaît, d'après cette structure, que les commandes que l'on déduit du système de pilotage sont fonction de l'état du processus de production, et d'autre part que le comportement du système opérationnel est fonction des commandes sortantent du système de pilotage. La relation entre le système opérationnel et le système de pilotage apparaît donc comme un système en boucle fermée, cette forme est appelé système asservi ou asservissement.

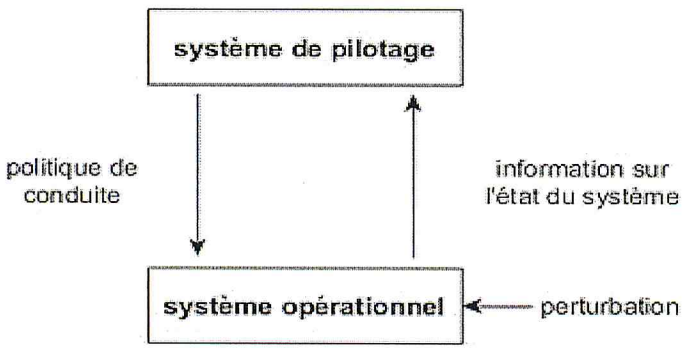



Figure 1.2: Structure générale d'un système de gestion de la production

### B.Autres structures d'un système de gestion de la production

Les objectifs du système de gestion de la production deviennent de plus en plus complexes. En effet, dans un environnement concurrentiel en perpétuel changement, la demande est caractérisée par :

-Son évolution éventuellement saisonnière

-Son évolution aléatoire due aux perturbations auxquelles elle est soumise : annulation de commandes, nouvelles commandes à caractères urgents.

-Sa traduction en termes quantifiables, tels que les quantités des produits désirées et les délais consentis ou en termes non quantifiables, tel que la quantité moyenne désirée, et la fiabilité de son estimation fortement dépendante de la quantité d'information disponible.

D'autres éléments perturbateurs compliquent la tâche de la gestion de la production: il s'agit des événements incertains (rupture d'outils, occurrence de pannes ..etc) ou des événements d'incertitudes relatifs à la durée d'exécution des opérations sur les machines, aux délais de réceptions des imputes, aux durées de transfères de produits entre machines, ...etc.

Les modèles de système de gestion qu'on retrouve dans la littérature se situent entre deux structures extrêmes : les systèmes centralisés et les systèmes décentralisés.Alors que les systèmes hiérarchisés se situent à l'intermédiaire de ces deux structures.

**(a) Les Systèmes Centralisés de Gestion :** Ils sont basés sur l'établissement d'un modèle de gestion monolithique ayant une vue globale sur le système. La détermination d'un plan prévisionnel se fait en tenant compte simultanément de tous les paramètres et informations. Toutes les variables de décision et les relations entre ces variables apparaissent de manière détaillée dans le modèle (figure 1.3).

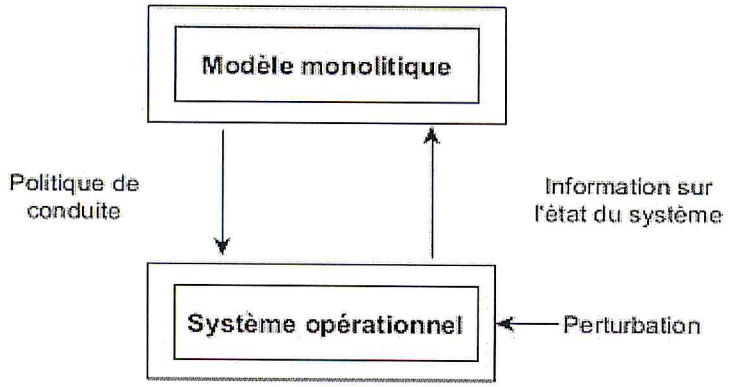



Figure 1.3: Système centralisé de gestion

**(b) Les Systèmes décentralisés de gestion :** Les méthodes décentralisées ont été développées en vue de faire face rapidement aux perturbations. Elles permettent le pilotage dynamique de l'atelier par le biais de règles de priorité simple où complexe (Figure 1.4).

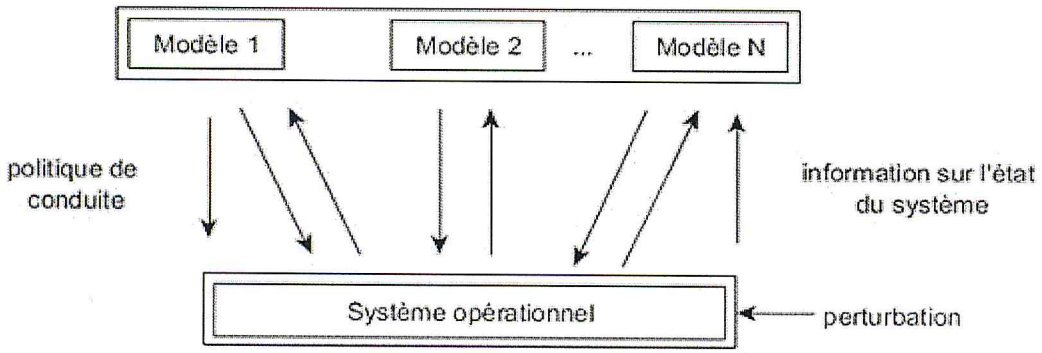



Figure 1.4: système décentralisé de gestion.

La structure répartie des informations entre centres de décisions justifie la vitesse de réaction par le fait que de faible volume de données généralement homogènes est analysé. Ces approches présentent par contre un certain nombre d'inconvénient parmi lesquelles :

-Des réactions non homogènes des centres de décision aux perturbation qui les effectuent.

-Des réactions non optimales, dues à la qualités des méthodes et des règles utilisées.

## 1.2 Présentation des problèmes d'ordonnancement

Dans cette partie, nous faisons un rappel concernant quelques notions de base sur les problèmes d'ordonnancement. Nous abordons en premier, les éléments fondamentaux d'un problème d'ordonnancement, pour ensuite donner une classification de ces problèmes. Enfin, une vue générale des méthodes classiques de résolution est présentée.

### 1.2.1 Définition

On trouve plusieurs définitions dans la littérature consacrées à l'ordonnancement. Nous rappelons celle proposée dans [13]: *"Ordonnancer c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution".*

Le terme réalisation a un sens large. En effet, selon le champ d'application de l'ordonnancement considéré, la réalisation peut concerner:

•en production : les opérations à effectuer sur des machines ;

•en informatique : les morceaux de programmes à exécuter sur des processeurs ;

•ou en gestion de projets : les activités (ou actions) à accomplir nécessitant une ou plusieurs ressources.

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

Un ordonnancement constitue une solution au problème d'ordonnancement. Il décrit

l'exécution des tâches et l'allocation des ressources au cours du temps. Plus précisément, on parle de problème d'ordonnancement lorsqu'on doit déterminer les dates de début et les dates de fin des tâches, alors qu'on réserve le terme de problème de séquencement au cas où l'on cherche seulement à fixer un ordre relatif entre les tâches qui peuvent être en conflit pour l'utilisation des ressources. Un ordonnancement induit nécessairement un ensemble unique de relations de séquencement. En revanche, la solution purement séquentielle d'un problème d'ordonnancement recouvre une famille d'ordonnancement (éventuellement une infinité si le domaine de variation des dates est non borné ou réel).

### 1.2.2 Éléments fondamentaux d'un problème d'ordonnancement

Les problèmes d'ordonnancement sont caractérisés par deux ensembles:

1) L'ensemble de $n$ tâches $T = \{t_1, t_2, ..., t_n\}$;

2) Les ensembles de $m$ machines $P = \{p_1, p_2, ..., p_m\}$.

L'ordonnancement est une allocation des machines et des ressources aux tâches dans l'ordre afin de compléter toutes les tâches sous des contraintes imposées. Il y a deux contraintes générales dans la théorie classique de l'ordonnancement:

$a^\circ$/ Chaque tâche est à exécuter au plus sur une machine.

$b^\circ$/ Chaque machine n'est capable d'exécuter qu'une tâche à la fois.

**Les tâches:**

Une tâche $i$, appartenant à un ensemble $V$ fini de tâches à ordonnancer, est une entité élémentaire de travail caractérisée par :

La date de disponibilité (release date en anglais) $r_i$: c'est la date de début au plus tôt de la tâche $i$.

La date d'échéance (due date en anglais) $d_i$: c'est la date de fin d'exécution préférentielle de la tâche $i$; lorsque cette date d'échéance doit être respectée, on parlera de date d'échéance stricte (deadline en anglais).

La durée opératoire (processing time en anglais) $p_i$: c'est le temps d'exécution de la tâche $i$.

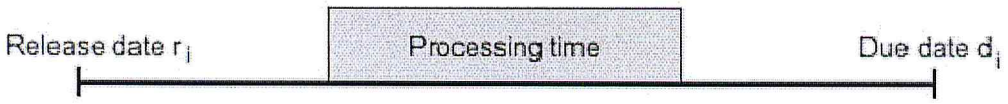



Figure 1.5: Les caractéristiques d'une tâche.

La localisation de la tâche dans le temps est définie par une date de début $t_i \geq r_i$ et/ou une date de fin $f_i \leq d_i$. Dans le cas où la durée opératoire $p_i = f_i - t_i$ est connue, la valeur de la variable $f_i$ peut être déduite de la valeur de $t_i$ (et vice-versa). La réalisation de la tâche $i$ nécessite l'utilisation d'une ou plusieurs ressources $k \in P$, où $P$ est l'ensemble de ressources.

Dans certains problèmes, les tâches peuvent être exécutées par morceaux, l'entrelacement des différents morceaux permet de laisser le moins possible les ressources inactives ; dans d'autres cas contraires, on ne peut pas interrompre une tâche une fois commencée. On parle alors respectivement de problèmes *préemptifs* et *nonpréemptifs*.

**Les ressources:**

Une ressource $k$ est un moyen technique ou humain, disponible en quantité limitée, destiné à être utilisé pour la réalisation de plusieurs tâches. La disponibilité est généralement exprimée par une capacité propre à chaque ressource $k$, notée $Q_k$ avec $(Q_k \geq 1)$.

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

Dans certains problèmes d'ordonnancement, la notion d'état de ressource est également utilisée afin de prendre en compte des contraintes technologiques liées à son utilisation. Dans ce cas, la capacité de la ressource dépend de l'état considéré. Un changement d'état de la ressource peut être produit par l'occurrence d'événements externes incontrôlables (panne, maintenance, ...etc.) ou par le début ou la fin de certaines tâches du probléme.

**Les contraintes:**

Dans la plupart des problèmes d'ordonnancement, les tâches à exécuter sont soumises à des contraintes qu'il faut satisfaire au moment de la recherche d'une solution optimale. On distingue trois types de contraintes :

-Les contraintes potentielles : Ce sont les contraintes de localisation temporelle par exemple "la tâche $i$ doit précéder la tâche $j$" ou la tâche $i$ doit être achevée avant telle date. Avec ce type de contrainte, le problème est dit "problème central d'ordonnancement".

-Les contraintes disjonctives : Lorsque deux tâches ne peuvent pas être exécutées en même temps, on dit qu'il y a une contrainte disjonctive que doivent satisfaire ces deux tâches.

-Les contraintes cumulatives: Elles concernent l'évolution dans le temps du volume total des moyens humains ou matériels consacrés à l'exécution des tâches.

**L'objectif et les critères:**

Selon le domaine d'application, la fonction ordonnancement peut avoir d'autres objectifs que celui de veiller au simple respect des contraintes de temps et de ressources. Il peut s'agir de satisfaire des objectifs économiques, de respecter une législation du travail en vigueur dans une entreprise ou, de gérer au mieux le risque en présence d'incertitudes.

Lorsque ces objectifs sont quantifiables et exprimables en fonction des variables d'ordonnancement, ils sont injectés dans le problème d'ordonnancement, soit par ajout de nouvelles contraintes, soit par ajout d'un ou plusieurs critères d'optimisation. Dans le premier cas, la fonction ordonnancement doit produire une solution admissible, c'est-à-dire satisfaisant l'ensemble des contraintes formulées. Dans le second cas, la solution produite doit non seulement être admissible, mais aussi minimiser ou maximiser la valeur du ou des critères considérés.

Pour les approches d'optimisation, les critères d'évaluation numériques peuvent être :

-soit liés au temps, comme dans le cas de la minimisation du temps total d'exécution ($C_{max}$) ou des retards vis-à-vis des dates d'échéance fixées ($L_{max}$ou $T_{max}$)

-soit liés aux ressources, comme dans le cas de la minimisation de la quantité de ressources nécessaires à la réalisation des tâches ou la charge de chaque ressource.

-soit liés aux coûts (de lancement, de production, de transport,... etc.) qu'un ordonnancement induit.

### 1.2.3 Classification des problèmes d'ordonnancement

Il existe plusieurs notations dans la littérature pour spécifier les problèmes d'ordonnancement. Un schéma de classification proposé par [23], puis repris par [8], structure les données d'un problème d'ordonnancement sur la base d'une notation à trois champs distincts :$\alpha \mid \beta \mid \gamma$

$\alpha$: décrit l'environnement machine.

$\beta$ : précise les contraintes liées aux tâches.

$\gamma$: décrit le ou les critères à optimiser.

**Le champ $\alpha$ :**

Le champ $\alpha$ se décompose dans les environnements machines les plus courants en deux sous champs $\alpha_1$ et $\alpha_2$. Le premier ($\alpha_1$) indique la nature du problème, (job shop, flow shop, open shop, etc.), alors que le second ($\alpha_2$) précise le nombre de machines ou d'ensembles (*pools*) de ressources.

Les différents environnements machines possibles, spécifiés dans le champ $\alpha_1$, sont :

$\alpha_1 = \phi$ (une seul machine (ou ressource unique)): c'est un cas simple où une seule machine (ressource) traite qu'une opération à la fois, on dit que c'est une ressource disjonctive.

$\alpha_1 = P$ (machines parallèles *identiques*): toute tâche peut être executée avec la même durée opératoire sur les machines en paralèlles.

$\alpha_1 = Q$ (machines parallèles $uniformes$): chaque machine a une durée d'exécution propre et la durée d'exécution est la même pour tous les travaux d'une même machine.

$\alpha_1 = R$ (machines parallèles *indépendantes*) : la durée d'exécution est différente pour chaque machine et chaque travail.

$\alpha_1 = F$ ($flow\ shop$) : les travaux d'un même job visitent des machines en série avec le même ordre (même gamme operatoire).

$\alpha_1 = J$ ($job\ shop$) : c'est un cas général de flow shop où la durée d'exécution est différente

| Valeur | description |
|---|---|
| $\varnothing$ | Machine unique |
| $P$ | Machines parallèles identiques |
| $Q$ | Machines parallèles uniformes |
| $R$ | Machines parallèles indépendantes |
| $F$ | Flow shop |
| $J$ | Job shop |
| $O$ | Open shop |
| $FH$ | Flow shop hybride |
| $JG$ | Job shop généralisé |
| $OG$ | Open shop généralisé |

Table 1.1: Quelques valeurs du champ $\alpha 1$

pour chaque travail (chaque job à sa propre gamme operatoire).

$\alpha 1 = O$ (*open shop*) : L'ordre de passage sur les machines est libre pour tous les travaux.

**Le champ $\beta$ :**

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

$\beta = \phi$ : si $\beta$ ne prend aucune valeur, alors tous les travaux sont disponibles à l'instant initial.

$\beta = r_j$ (dates de disponibilité) : indique que les dates de disponibilité des travaux sont distinctes.

$\beta = pmtn$ (préemption) : indique que la préemption est autorisée, i.e. l'exécution d'un travail peut être interrompu pour lancer un autre.

$\beta = prec$ (précédence): indique l'existence des contraintes de précédence entre les travaux, et son absence indique le contraire (les travaux ne sont pas soumis aux contraintes de précédence).

$\beta = nwt$ (No-wait) : indique que les travaux doivent se succéder sans attente.

$\beta = (p_j = p)$ (Durées opératoires égales) : Si la valeur $p_j = p$ apparaît dans le champ $\beta$ cela veut dire que touts les travaux ont la même durée d'exécution.

| Valeur | Description |
|---|---|
| $Prec$ | Il existe des contraintes de précédence générale entre les opérations. |
| $r_i$ | Une date de début au plus tôt $r_i$ est associée à chaque tâche $i$. |
| $d_i$ | Une date d'échéance préféretielle $d_i$ est associée à chaque tâche $i$. |
| $p_i = p$ | les durées opératoires sont identiques |
| $Pmtn$ | la préemption des opérations est autorisée. |
| $no - wait(nwt)$ | les opérations de chaque travail doivent se succéder sans attente. |
| $snsd(rsnd)$ | les ressourses doivent être préparées avant et/ou aprés chaque execution indépendamment de la séquence des taches |

Table 1.2: Quelques valeurs du champ $\beta$

**Le champ $\gamma$ :**

Dans le dernier champ $\gamma$ apparait le critère d'optimisation. Le plus souvent, le critère est exprimé en fonction des variables de décision associées à chaque tâche $j$, qui sont :

- la date de fin d'exécution du travail $j$ sur la dernière machine : $C_j$.
- la durée de séjour du travail $j$ dans l'atelier (*Flow-time*). Son expression est donnée par $F_j = C_j - r_j$.
- le retard algébrique (lateness) qui dénote l'écart par rapport au délai souhaité avec avance favorable et retard défavorable. Il est exprimé par $L_j = C_j - d_j$.
- le retard vrai (*tardiness*) donné par l'expression, toujours positive,

$Tj = max(0; C_j - d_j) = max(0; L_j)$.

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

Les indicateurs de performance usuels sont alors :

- la durée totale de l'ordonnancement (*makespan*) $C_{max} = max_j C_j$;
- le plus grand retard algébrique $L_{max} = max_j L_j$;

| Valeur | Le critère considéré |
|---|---|
| $C_{\max}$ | La durée totale (makespan). |
| $L_{\max}$ | Le plus grand retard algébrique. |
| $T_{\max}$ | Le plus grand retard vrai. |
| $\sum U_i$ | Le nombre de tâches en retard. |
| $\sum [w_i] C_i$ | La durée moyenne ou pondérée des tâches. |
| $\sum [w_i] U_i$ | Le nombre moyen ou pondéré des tâches en retard. |
| $\sum [w_i] T_i$ | Le retard moyen ou pondérée |

Table 1.3: Quelques valeurs du champ $\gamma$

- la plus grande durée de séjour $F_{max} = max_j F_j$;
- le plus grand retard vrai $T_{max} = max_j T_j$ ;
- le nombre de tâches en retard $N_T = \sum_{j=1}^{n} U_j$.
- le retard moyen $T = \frac{1}{n} \sum_{j=1}^{n} T_j$.
- la durée moyenne de séjour $F = \frac{1}{n} \sum_{j=1}^{n} F_j$.

À partir de ces mesures, des critères d'appréciation ou d'évaluation des solutions du problème d'ordonnancement sont construits. Pour celles que nous avons présentées plus haut, l'objectif consiste dans tous les cas à minimiser ces mesures de performance.

## 1.3 Approches classiques de résolution

La résolution d'un problème d'ordonnancement consiste à trouver une séquence d'exécution des tâches de sorte qu'un certain critère d'optimisation atteigne sa valeur optimale. Les méthodes de résolution des problèmes d'ordonnancements puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, théorie des graphes, etc.). Ces méthodes garantissent en général l'optimalité de la solution fournie.

Pour certains problèmes donnés avec des critères bien définis, on a pu trouver des solutions optimales en des temps de calcul polynomiaux. On les appelle les méthodes optimales efficaces.

Pour les autres problèmes, on utilise des méthodes énumératives. Bien que leur complexité temporelle soit généralement exponentielle, elles donnent en pratique, sur des problèmes de taille moyenne, des solutions optimales en un temps raisonnable.

Cependant, pour les problèmes de grande taille dont on sait que la complexité est très grande, les méthodes exactes ne peuvent pas être utilisées, à cause du temps de calcul qui est très grand, d'où la nécessité de construire des méthodes de résolution approchées, qu'on appelle méthodes heuristiques, qui peuvent donner de bonnes solutions en un temps raisonnable.

### 1.3.1 Les méthodes optimales:

Parmi les plus connues, citons: [1]

-la règle $SPT$ (*Shortest Processing Time*) pour le problème $1 \parallel \sum C_j$ ;

-la règle $WSPT$ (*Weighted Shortest Processing Time*) pour le problème $1 \parallel \sum w_j C_j$;

-la règle $EDD$ (*Earliest Due Date*) pour le problème $1 \mid d_j \mid L_{max}$ ;

-l'algorithme de *Moore et Hodgson* pour le problème $1 \mid d_j \mid \sum U_j$;

-l'algorithme de *Johnson* pour le problème $F_2 \mid prmu \mid C_{max}$.

### 1.3.2 Les méthodes optimales énumératives:

Ce sont des méthodes qui permettent de garantir, en général, pour des instances pas trop grandes, l'optimalité de la solution fournie. Dans cette classe, on peut distinguer :

-La méthode par séparation et évaluation ($PSE$): C'est une méthode de recherche arborescente qui énumère à travers des nœuds des sous-ensembles de solutions. Le principe est d'éliminer de l'arbre de recherche les branches prouvées non-optimales afin d'éviter l'énumération exhaustive des solutions, en utilisant les bornes inférieures et supérieures du

critère.

-La programmation linéaire (PL) où on modélise les contraintes et les critères sous formes de fonctions linéaires en variables mixtes.

-La programmation linéaire en nombres entiers: La PLNE est souvent difficile à résoudre du fait que l'espace de recherche n'est plus convexe mais discret. Dans ces cas-là on utilise principalement deux approches de recherche de la solution optimale entière : Les méthodes arborescentes par séparation et évaluation (branch and bound ) et les méthodes de coupes (branch and cut).

-Relaxation lagrangienne : C'est une méthodologie générale qui permet d'obtenir des bornes inférieures de bonne qualité pour certains problèmes d'optimisation combinatoire. L'idée de la technique consiste à supprimer (relaxer) une partie des contraintes (en principe celles qui rendent le problème difficile) en les introduisant dans la fonction objectif sous forme d'une pénalité (combinaison linéaire des contraintes relaxées) si les contraintes relaxées ne sont pas respectées.

-Génération de colonnes : Cette méthode repose sur la décomposition du modèle original par la méthode de Dantzig et Wolfe [15]. L'idée principale consiste à décomposer le modèle d'origine en un ou plusieurs sous-problèmes plus faciles à résoudre, l'ensemble étant généralement coordonné par un programme linéaire appelé maître. Les méthodes utilisant la génération de colonnes rencontrent souvent des problèmes de convergence. En effet, la rapidité de la méthode dépend, à la fois, du nombre d'itérations nécessaires jusqu'à preuve d'optimalité et du temps de calcul consacré à chaque étape de résolution.

-La programmation dynamique consiste à procéder à une décomposition du problème initial en plusieurs sous-problèmes plus facile à résoudre optimalement, puis pour obtenir la solution optimale, un parcours à rebours des décisions prises dans chaque sous-problème est effectué.

### 1.3.3 Les méthodes heuristiques:

Les heuristiques se distinguent des méthodes exactes par le fait qu'elles proposent des solutions de bonne performances mais non optimales pendant un temps de calcul raisonnable.

On compte plusieurs types d'heuristiques, généralement classifiées en trois grandes familles:

-Les algorithmes gloutons dans lesquels on construit une solution étape par étape, sans jamais remettre en cause une décision déjà prise, avec l'espoir de se rapprocher le plus vers la solution optimale. En ordonnancement, les algorithmes gloutons sont des méthodes de construction par règles de priorité simples de type $SPT$ (*Shortest Processing Time*), $EDD$ (*Earliest Due Date*), etc...

-Les méthodes de recherche locale (*tabou, recuit simulé*). Ces méthodes partent d'une solution de départ unique, puis définissent un voisinage, qui est exploré pour trouver une meilleure solution. Parmi les méthodes de voisinage, on peut citer les méthodes de descente pour lesquelles une solution voisine doit obligatoirement améliorer le critère, i.e., s'arrête dès qu'il n'existe plus de meilleure solution dans le voisinage. On note toutefois, qu'afin de pouvoir s'échapper des minima locaux, d'autres approches autorisent de dégrader provisoirement le critère dans le but de pouvoir obtenir un optimum global.

-Les métaheuristiques à base de population, connues aussi sous l'appellation d'algorithmes évolutionnistes, sont des méthodes de recherche qui travaillent sur une population de solutions plutôt que sur une solution unique. Les plus connues dans ce genre sont : les algorithmes génétiques développés par *Holland* [24], et l'algorithme de colonies de fourmis introduit initialement par *Dorigo* et *al.* [17] pour la résolution du problème du voyageur de commerce, puis élargie à d'autres domaines d'application.

**Conclusion:**

Après ce rappel sur quelques notions de bases relatives aux problèmes d'ordonnancement, nous entamerons dans le chapitre qui suit, l'étude du problème d'ordonnancement à une machine. De plus, nous énonçons le théorème des pyramides.

# Chapitre 2

# Problème d'ordonnancement à une machine et théorème des pyramides

Dans ce chapitre nous présentons le problème à une machine et quelques notions utiles à la compréhension de la suite du mémoire, ainsi qu'un théorème de dominance qui est utilisé dans la modélisation de notre problème.

## 2.1 Etat de l'art

Les problèmes d'ordonnancement à une machine, d'apparence simples, sont dans la plupart des cas NP-difficiles dès que des fenêtres d'exécutions sont associées aux travaux. Ils sont fondamentalement très importants car souvent rencontrés en pratique. Leur modélisation sert de base et permet la compréhension et la résolution des problèmes à plusieurs ressources plus complexes. En effet, les problèmes d'ordonnancement mettant en jeu des environnements à plusieurs machines peuvent être décomposés en sous-problèmes d'ordonnancement plus simples à une machine. L'exemple, souvent cité, est celui d'un problème d'atelier se réduisant au seul problème à une machine lorsqu'un poste de travail constitue un goulot d'étranglement [36].

Le problème consiste à séquencer un ensemble $V$ fini de tâches sur une seule machine (une seule ressource), tout en respectant des contraintes et en optimisant un objectif bien précis, comme la minimisation du nombre de travaux en retard,$\sum U_i$, etc...

Dans notre étude, nous nous intéressons à un problème avec fenêtre d'exécution (les dates de disponibilité $r_i$ sont différentes) avec comme critère la minimisation du nombre de tâches en retard, problème qui s'écrit selon la classification de Blazewicz : $(1|\ r_i|\sum U_i)$.

où $U_i = 1$ si le travail $i$ est achevée aprés sa date échue. Ce problème est NP-difficile [29].

### 2.1.1 Hypothèses du problème

Pour notre problème (minimisation du nombre de travaux en retard), nous considérons les hypothèses suivantes :

- La machine est prête à l'instant $t = 0$.
- La capacité de traitement des tâches est suffisamment grande.
- La machine ne tombe pas en panne et ne nécessite aucun réglage.
- Le nombre de tâches est fini (ordonnancement statique)
- Le temps de préparation des tâches est inclu dans leurs durées d'exécution.
- L'ensemble $V$ des tâches est connu à l'instant $t = 0$ :

Chaque tâche $i$ est caractérisée par :

- Une date de disponibilité $r_i$ une date échue $d_i$ et une durée opératoire $p_i$, comme mentionné au premier chapitre.
- une tâche $i$ ne peut pas commencer son exécution avant sa date de disponibilité et doit vérifier $t_i \geq r_i$ .
- si une tâche termine son exécution après sa date échue (*i.e.* $f_i > d_i$ ) alors elle est en retard.
- lorsqu'une tâche $i$ de $V$ vérifie la contrainte suivante : $t_i + p_i \leq d_i$ , sa réalisation est assurée et est considérée comme une tâche qui n'est pas en retard.
- Aucune contrainte entre les tâches n'est imposée
- Pour les tâches en retard $(t_i + p_i > d_i)$ on peut les séquencer arbitrairement après les tâches non en retard.

### 2.1.2 Cas particuliers du problème $(1\ |r_i|\ \sum U_i)$

Le problème $(1\ |r_i|\ \sum U_i)$ du fait de sa complexité et de son intérêt pratique, a suscité beaucoup d'intérêt de la part de la communauté scientifique. De nombreux résultats ont été obtenus concernant soit directement ce problème, soit une de ses nombreuses extensions :

J.M .Moore a étudié le problème dans le cas particulier où les dates de début au plus tôt sont identiques [31]. Ce problème dénommé par $(1\ |r_i = r|\ \sum U_i)$ est résolu en $O(n \log n)$ par un algorithme simple.

C.L. Monma [32] décrit lui aussi un algorithme optimal dans le cas où les temps opératoires sont tous identiques et de durée (1) i.e. pour le problème$(1\ |p_i = 1|\ \sum U_i)$; l'algorithme est polynomial, et s'exécute en $O(n)$.

Le problème général $(1\ |r_i|\ \sum U_i)$, dans lequel les dates de début au plus tôt sont différentes, a été prouvé NP-difficile [29] . Cependant H. Kise et al. [26] ont donné un algorithme en $O(n^2)$ pour résoudre le problème dans le cas particulier où les dates de début au plus tôt et les dates échues sont ordonnées identiquement *i.e* telles que $r_i < r_j \Longrightarrow d_i \leq d_j$. Pour le même cas particulier E. L. Lawler [27] a ramené la complexité à $O(n \log n)$.

Dans un autre article [28] E. L. Lawler a montré comment appliquer des techniques de programmation dynamique pour résoudre le problème dans le cas général.

Quand des contraintes de précédence apparaissent entre les tâches, le problème avec durées opératoires unitaires $(1|prec, p_i = 1|\ \sum U_i)$ devient NP-difficile [22].

Quand les tâches élémentaires peuvent être regroupées en lots pour être exécutées simultanément sur la machine, et que les dates de début au plus tôt sont toutes identiques, le problème reste NP-difficile [29].

•Algorithme de J. M. Moore

L'algorithme de Moore résout optimalement le problème $(1||\sum U_i)$ définit comme suit:

Un ensemble de $n$ tâches disponibles toutes à l'instant initial, doit être séquencé sur une ressource (machine) unique.

Chaque tâche $i$ a une date échue $d_i$ et sa durée d'exécution est $p_i$, l'objectif est de minimiser le nombre de tâches en retard.

J. M. Moore a proposé un algorithme de résolution dans [31] (cf. figure 2.1). Le principe de la procédure consiste à séparer l'ensemble des tâches en deux sous-ensembles; les tâches à l'heure et les tâches en retard. Les tâches sont ajoutées dans l'ensemble des tâches à «l'heure» dans l'ordre croissant de leur date échue.

L'algorithme de Moore est une simple procédure de balayage en avant, nommée en anglais forward procedure. On note $J$ l'ensemble des tâches «à l'heure» et $J^d$ celui des tâches «en retard». $J^c$ est l'ensemble des tâches qui n'ont pas encore été considérées. Si une tâche ajoutée $J_{j^*}$ est en retard $(\sum_{j\epsilon J} p_j > d_j)$, alors la tâche avec la plus grande durée opératoire est enlevée de l'ensemble ≪à l'heure≫ et placée dans l'ensemble «en retard» $J^d$.

La complexité de cet algorithme est $O(n \log n)$. La complexité est donnée par le tri des dates échues dans l'ordre croissant, $O(n \log n)$. A chaque itération, dans la boucle «repeat», une tâche est enlevée de $J^c$. Dans ce cas, l'algorithme finit après $n$ boucles. Trouver la tâche avec la durée d'exécution maximale se fait en $O(\log n)$ avec une structure de tas maintenue à jour à la ligne 4 de l'algorithme (cf. figure 2.1) en $O(\log n)$. Donc la complexité totale est de $O(n \log n)$. La preuve d'optimalité peut être trouvée dans [31].

Cet algorithme est souvent appelé: algorithme de Moore et Hogson.

**Algorithme de Moore :fordward procedure( )**

1. $J := \emptyset$ ; $J^d := \emptyset$ ; $J^c := \{J_1.....J_n\}$
2. repeat
3. $J_j* :=$ $J_j$ that satisfies $d_j * := \min_{J_j \in J^c}(d_j)$
4. $J := J \cup \{J_j*\}$
5. $J^c := J^c \backslash \{J_j*\}$
6. if $\sum_{J_j \in J} P_j > d_j*$
7. *then* $J_j* := J_k$ that satisfies$P_k* := \max_{J_k \in J}\ (P_k)$
8. $J\ := J - \{J_k*\}$
9. $J^d := J^d \cup \{J_k*\}$
10. *until* $J^c = \emptyset$

Figure 2.1: Algorithme de Moore

- **Algorithme de Lawler**

Dans l'algorithme de Lawler, les tâches sont supposées ordonnées de telle sorte que $r_1 \leq r_2 \leq ..... \leq r_n$ et $d_1 \leq d_2 \leq .... \leq d_n$.La solution au problème de minimisation du nombre de travaux en retard se fait par un algorithme de tri en $O(n \log n)$ [27].

**Algorithme de Lawler**

Lawler procedure()

1. $S' := \emptyset$; $S := \emptyset$; $r_0 := 0$; $P := 0$
2. *for* $J_j := J_1$ *to* $J_n$
3. *do* $r := r_j - r_{j-1}$
4. while $P > 0$ and $r > 0$

5. do $J_i := J_h$ that satisfies $P_k \min_{J_k \in S}(P_k)$
6. if $r \geq P_i$
7. then $S := S - \{J_i\}$
8. $S' := S' \cup \{J_i\}$
9. $P := P - P_i$
10. $r := r - P_i$
11. else $P_i := P - r$
12 $P := P - r$
13. $r := 0$
14. $S := S \cup \{J_i\}$
15. $P := P + P_i$
16. if $r_j + P > d_j$
17. then $J_i := J_h$ that satisfies $P_h = \max_{J_K \in S}(P_k)$
18. $S := S - \{J_i\}$
19. $P := P - P_i$

Figure 2.2: Algorithme de Lawler

### 2.1.3 Cas général du problème $(1 \mid r_i \mid \sum U_i)$

Dans le cas général du problème $(1 \mid r_i \mid \sum U_i)$, une méthode arborescente, basée sur la résolution du problème préemptif et sur des techniques de propagation de contraintes, est proposée par P.Baptiste, L.Peridy et E.Pinson [6] . Une autre méthode exacte est décrite par Dauzère-Pérès et Sevaux [16], très efficace et peut être appliquée à des problèmes comptant jusqu'à 200 travaux, et c'est la première fois que des problèmes de cette taille sont abordé et résolus de façon exact.

Recemment Ourari et al dans [35] proposent de résoudre le problème par des modèles de programmations mathématique en fournissant des bornes au problème et ce en utilisant des conditions de dominance.

## 2.2 Structure d'intervalle et dominance

La notion de structure d'intervalles, est en effet au cœur d'une approche fondée sur un théorème des pyramides pour la caractérisation d'un ensemble dominant de solutions que nous présentons dans ce qui suit.

### 2.2.1 Structure d'intervalle et algèbre d'Allen

Une structure d'intervalle est définie par le couple $< I, C >$ où $I = \{i_1, ...i_n\}$ est un ensemble d'intervalles et $C$ est un ensemble de contraintes sur $I \times I$ . Chaque intervalle est défini par une date de début $x_j$ et une date de fin $y_j$. Une contrainte entre deux intervalles peut être exprimée soit en spécifiant un ordre total entre les dates de début et de fin des deux intervalles, soit de façon équivalente, en utilisant une des relations de l'algèbre d'Allen [2] récapitulées sur la Figure 2.3.

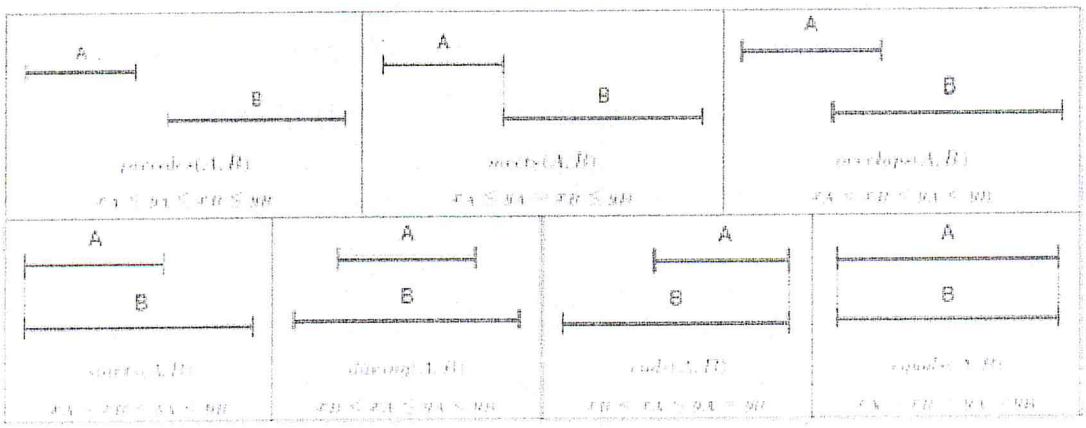



Figure 2.3: Algèbre d'Allen

### 2.2.2 Notions de sommet et pyramide

Par considération des relations d'Allen, deux types particuliers d'intervalles peuvent être mis en évidence : l'intervalle de type base et, au tour des laquels est définie la notion de pyramide.

**Définition 1** *:Un travail t est dit sommet si son intervalle d'exécution est tel que :*

$\ll \nexists\, i \epsilon V$ tel que during $(i,t) \gg \Longleftrightarrow \ll \nexists\, i \in V$ tel que $r_i > r_t \wedge d_i < d_t \gg$ .

notons que si on considère un ensemble de sommets, ces derniers sont indicés dans l'ordre de leur $r_i$ *croissant* ou, en cas d'égalité, dans l'ordre de leur $d_i$ croissant.

**Définition 2** *: Une pyramide $P_t$, associée au sommet t d'une structure d'intervalle, est le sous-ensemble de tâches ayant leur intervalle d'exécution tel que :*

$\ll \text{during}(t,i) \gg \Longleftrightarrow \ll r_i < r_t \wedge d_i < d_i \gg$ .

Pour illustrer les deux définitions précédentes, et définir les sommets et les pyramides à partir d'un exemple concernant un ensemble de travaux, considérons la structure d'intervalle représentée sur la figure 2.4. Sur cet exemple, on identifie les sommets: $s1$ et $s2$ qui carectérisent respectivement les pyramides: $P_1 = \{d\}$ et $P_2 = \{b, c, d\}$. Remarquons que, compte tenu de la définition d'une t-pyramide, les sommets n'appatiennent pas à la pyramide qui caractérisent.

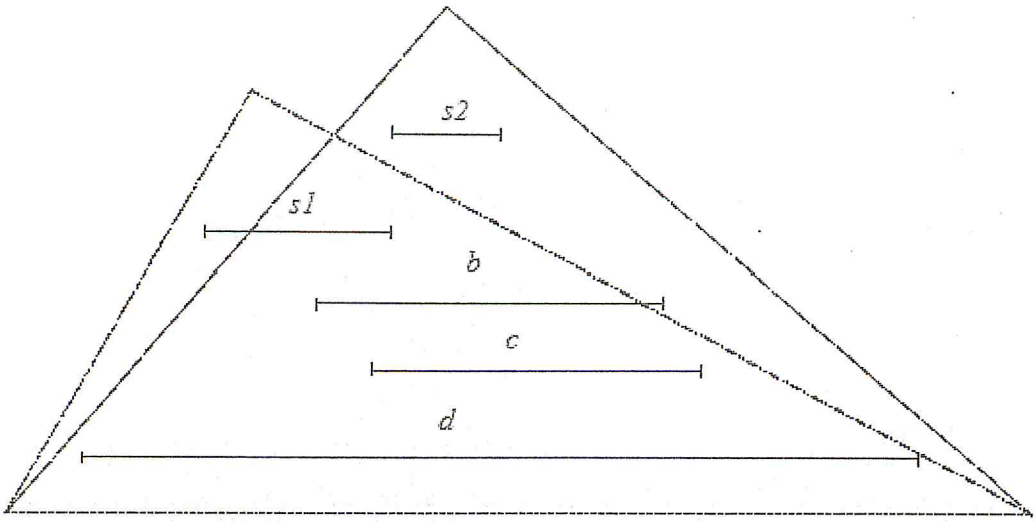



Figure 2.4: Diagramme des intervalles, sommets et pyramides.

### 2.2.3 Notion de dominance en ordonnancement

Pour résoudre un problème d'ordonnancement, il est possible de réduire l'espace de recherche à considérer en utilisant des conditions de dominance vis-à-vis de l'optimalité ou de l'admissibilité.

Dans le cadre du problème à une machine, si on s'intéresse à l'étude de l'admissibilité (i.e.trouver une solution réalisable),une séquence seq 1 domine une séquence seq 2 si et seulement si seq 2 admissible implique seq 1 admissible [20] les séquences dominantes peuvent être considérées comme potentiellement meilleures que les autres dans le sens où, si aucune n'est réalisable, alors cela implique qu'il n'existe pas de solution réalisable pour le problème considéré.

Un sous ensemble de solution est dit **dominant** pour l'optimisation d'un critère donné, s'il contient au moins un optimum pour ce critère. Un sous ensemble de solution dominantes est caractérisé à l'aide de **condition de dominance**. Mettre en évidence des conditions de dominance éfficaces, vis-à-vis de la résolution d'un problème d'optimisation, conduit généralement à la conception de méthodes de résolution elles mêmes très efficaces.

### 2.2.4 Théorème des pyramides

Le théorème des pyramides permet de limiter l'espace de recherche d'une solution réalisable sous certaines hypothèses. Les hypothèses considérées prennent en compte seulement l'ordre relatif des dates de début au plut tôt $r_j$ et de fin au plus tard $d_j$ des $n$ travaux .Les durées opératoires $p_j$ ainsi que les valeurs explicites des $r_j$ et $d_j$ de chaque travail ne sont pas considérées. Cela fait que les résultats suivant restent valides quelles que soient les durées opératoires et quelles que soient les valeurs de $r_j$ et $d_j$ tant que l'ordre relatif entre ces derniéres est respecté [25].

Pour caractériser un ensemble de séquences dominantes, nous utilisons les notions de sommet et de pyramide cf. sous section 2.2.2.

Nous rappelons 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 $d_j$. Ceci est équivalent à affirmer que $s_\alpha$ et $s_\beta$ étant deux sommets de la structure d'intervalles, alors $\alpha \prec \beta$ si et seulement si $r_{s_\alpha} \leq r_{s_\beta}$ et $d_{s_\alpha} \leq d_{s_\beta}$. Si deux sommets possédent des dates de début au plus tôt et de fin au plus tard identiques, alors ils peuvent être indicés de façon quelconque. La pyramide notée $P_\alpha$ désigne la pyramide caractérisée par le sommet $s_\alpha$.

On note $u(i)$ (respectivement $v(i)$) l'indice de la premiére pyramide à la quelle le travail $i$ appartient (respectivement l'indice de la derniére pyramide à la quelle le travail $i$ appartient). Par la notation $i \prec j$, on disigne que l'éxecution du travail $i$ précède celle de $j$. Un ordre partiel dominant peut alors être caractérisé grâce au théorème suivant:

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

1. *Les sommets sont ordonnés dans l'ordre de leur indice;*
2. *Avant le premier sommet, seuls sont placés des travaux appartenant à la première pyramide, rangé dans l'ordre croissant de leurs $r_i$ ou en cas d'égalité, dans un ordre arbitraire;*
3. *Aprés le dernier sommet seuls sont placés des travaux appartenant à la dernière pyramide rangé dans l'ordre croissant de leurs $d_i$ 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 pyramide $P_k$ et n'appartenant pas à $P_{k+1}$ dans l'ordre croissant de leurs $d_i$ (dans un ordre arbitraire en cas d'énigalité), 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 $r_i$ (dans un ordre arbitraire en cas d'égalité).*

Le théorème des pyramides définit un ordre partiel dominant dans la mesure où il impose des relations de précédence entre les sommets d'une part,puisque $s_k \preceq s_{k+1}$, et entre les travaux non-sommets et les sommets d'autre part, puisque $s_{u(j)-1} \prec j \prec s_{v(j)+1}$. Remarquons que, si deux travaux non-sommets $i$ et $j$, telque $r_i < r_j$, appartenant à une seule pyramide de sommet $s$, sont tous deux séquencés avant alors $i \prec j$ et l'ordre total $j \prec i \prec s$ est interdit. Ce théorème introduit donc, outre un ordre partiel dominant, des relations de dominance conditionnelles, liées à la position des travaux relativement aux sommets, et à l'ordre relatif des dates de début au plus tôt et de fin au plus tard.

Une propriété intéressante du théorème des pyramides est qu'il permet de mesurer la cardinalité de l'ensemble des séquences dominantes $S_{dom}$ obtenu lors de l'application du théorème. Cette cardinalité est calculée grâce à la formule suivante:

$$card(S_{dom}) = \Pi_{q=1}^{N}(q+1)^{n_q} \tag{2.1}$$

Où :

$n_q$:désigne le nombre de travaux non sommet appartenant exactement à $q$ pyramides.

$N$ :le nombre totale de pyramides.

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

### 2.2.5 Exemple

Pour illustrer le théorème des pyramides, nous utilisons l'exemple extrait de l'article [12].

Dont les dates de début, les dates de fin et les durées opératoires sont indiquées dans le tableau 2.1. Dont l'ordre relatif entre les dates au plus tôt ($r_i$) et de fin au plus tard ($d_i$) est le suivant: $r_6 < r_1 < r_3 < r_2 < r_4 < d_2 < d_3 < d_4 = r_5 = r_7 < d_6 < d_5 < d_1 < d_7$.

Le diagramme des intervalles associé à cet exemple est donné sur la figure 2.5, ainsi que la décomposition en pyramides correspondante.

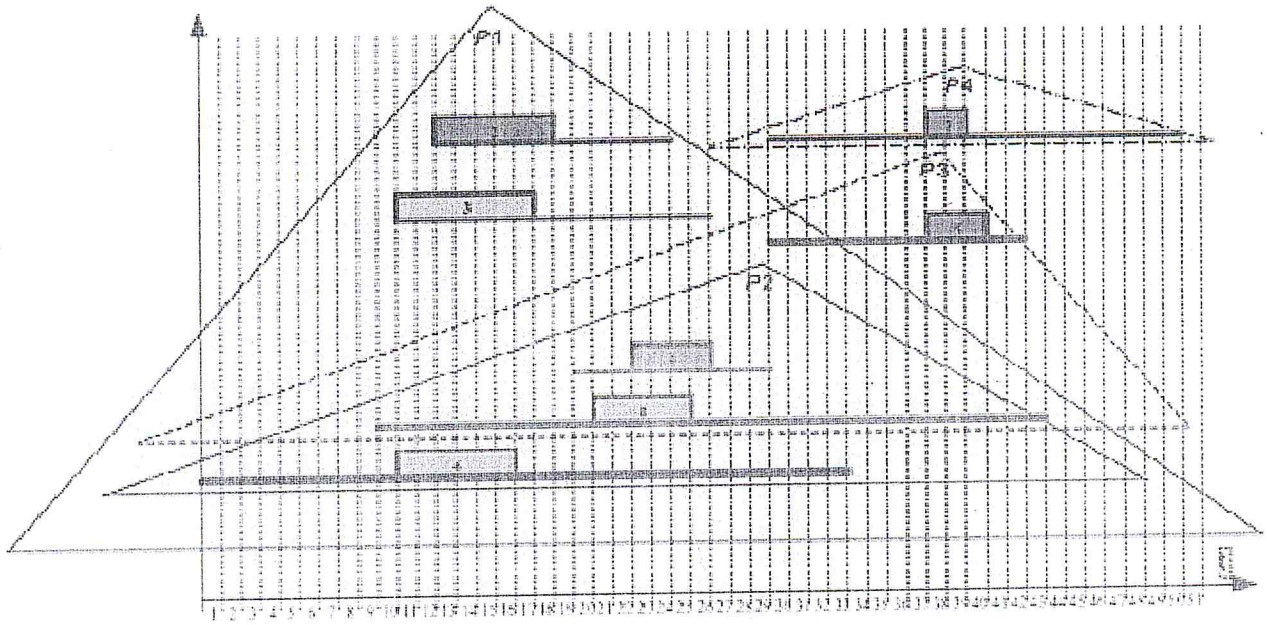



Figure 2.5: Diagramme des intervalles, sommets et pyramides

| tâche | $r_j$ | $p_j$ | $d_j$ |
|---|---|---|---|
| 1 | 9 | 5 | 43 |
| 2 | 12 | 6 | 24 |
| 3 | 10 | 7 | 26 |
| 4 | 19 | 4 | 29 |
| 5 | 29 | 3 | 42 |
| 6 | 0 | 6 | 33 |
| 7 | 29 | 2 | 50 |

Table 2.1: Une instance d'un problème à une machine

Compte tenu de la structure d'intervalles associée à cet exemple, on distingue quatre sommets : $s_1 = 2, s_2 = 4, s_3 = 5, s_4 = 7$ caractérisant les quatre pyramides: $P_1 = \{1, 3, 6\}$, $P_2 = \{1, 6\}$, $P_3 = \{1\}$, $P_4 = \{\varnothing\}$.

Rappelons que, compte tenu de la définition d'une pyramide, les sommets n'appartiennent pas à la pyramide qu'ils caractérisent.

L'application du théorème des pyramides à cette structure d'intervalles permet de caractériser un ensemble dominant $S_{dom}$ de cardinalité:

$card(S_{dom}) = (1+1)^1 \times (2+1)^1 \times (3+1)^1 = 24$ (sur 7!= 5040 cas possibles) constitué des séquences données sur la Figure 2.6

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

Par exemple, considérons le cas de deux travaux $u, v$ et deux sommets $t_k, t_{k+1}$ ayant l'ordre relatif entre les $r_i$ et $d_i$ suivant: $r_u < r_v < r_{t_k} < r_{t_{k+1}} < d_{t_k} < d_{t_{k+1}} < d_u < d_v$.

| | | |
|---|---|---|
| 6 ≺1≺3 ≺2 ≺4 ≺5 ≺7, | 1 ≺3 ≺2 ≺6 ≺4 ≺5 ≺ 7, | 1 ≺3 ≺2 ≺4 ≺6 ≺5 ≺7, |
| 6 ≺1 ≺2 ≺3 ≺4 ≺5 ≺7, | 1 ≺2 ≺3 ≺6 ≺4 ≺5 ≺7, | 1 ≺2 ≺3 ≺4 ≺6 ≺5 ≺7, |
| 6 ≺3 ≺2 ≺1 ≺4 ≺5 ≺7, | 3 ≺2 ≺6 ≺1 ≺4 ≺5 ≺7, | 3 ≺2 ≺1 ≺4 ≺6 ≺5 ≺7, |
| 6 ≺2 ≺3 ≺1 ≺4 ≺5 ≺7, | 2 ≺3 ≺6 ≺1 ≺4 ≺5 ≺7, | 2 ≺3 ≺1 ≺4 ≺6 ≺5 ≺7, |
| 6 ≺3 ≺2 ≺4 ≺1 ≺5 ≺7, | 3 ≺2 ≺6 ≺4 ≺1≺ 5 ≺7, | 3 ≺2 ≺4 ≺6 ≺1 ≺5 ≺7, |
| 6 ≺2 ≺3 ≺4 ≺1 ≺5 ≺7, | 2 ≺3 ≺6 ≺4 ≺1 ≺5 ≺7, | 2 ≺3 ≺4 ≺6 ≺1 ≺5 ≺7, |
| 6 ≺3 ≺2 ≺4 ≺5 ≺1 ≺7, | 3 ≺2 ≺6 ≺4 ≺5 ≺1 ≺7, | 3 ≺2 ≺4 ≺6 ≺5 ≺1 ≺7, |
| 6 ≺2 ≺3 ≺4 ≺5 ≺1 ≺7, | 2 ≺3 ≺6 ≺4 ≺5 ≺1 ≺7, | 2 ≺3 ≺4 ≺6 ≺5 ≺ 1≺7. |

Figure 2.6: Ensemble dominant de séquences du problème de la figure 2.5

Les deux travaux $u$ et $v$ appartiennent donc aux deux pyramides $P_{t_k}$ et $P_{t_{k+1}}$ et peuvent être placés dans n'importe quel ordre entre les deux sommets. Considérons la séquence dominante $t_k \prec u \prec v \prec t_{k+1}$, si cette séquence est comptabilisée dans $S_{dom}$, la séquence $t_k \prec v \prec u \prec t_{k+1}$ bien qu'elle respecte le théorème des pyramides, ne le sera pas. La valeur

calculée par la formule $card(S_{dom})$ est donc une borne inférieure du nombre réel de séquences de l'ensemble dominant.

### 2.2.6 Notion de séquence maître-pyramide

Dans cette partie, et pour éviter d'énumérer toutes les séquences dominantes de $S_{dom}$ que le Théorème des pyramides caractérise, nous exploitons une séquence particulière, définie dans [9], nommée séquence **maître-pyramide** et notée $\sigma_\Delta\ (V)$ dans la suite du texte. Cette séquence est de la forme [35] :

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

- $m$ : est le nombre total de sommets;

- $\alpha_i$ : est l'ensemble des travaux appartenant à la pyramide $P_i$ du sommet $t_i$ et n'appartenant pas à la pyramide $P_{i-1}$, classés par ordre croissant de leurs dates de disponibilité;

- $\beta_i$ : est l'ensemble des travaux appartenant à la pyramide $P_i$ classés par ordre croissant de leurs dates échue.

Une séquence $S$ est dite compatible avec la séquence **maître-pyramide** $\sigma_\Delta\ (V)$ si l'ordre des travaux dans S ne contredit pas les ordres possibles définis par $\sigma_\Delta\ (V)$.

Toute séquence de $n$ travaux compatible avec $\sigma_\Delta\ (V)$ respecte le Théorème des pyramides. L'ensemble des séquences de $n$ travaux compatibles avec $\sigma_\Delta\ (V)$ représente l'ensemble dominant $S_{dom}$.

Par exemple, la séquence maître-pyramide associée au problème de l'exemple précédant est représentée dans la figure 2.7 :

$\sigma_\Delta = (6, 1, 3, 2, 3, 6, 1, 4, 6, 1, 5, 1, 7)$

On peut facilement vérifier que cette séquence maître caractérise bien les 24 séquences dominantes déduites du théorème des pyramides et qui sont décrites plus haut.

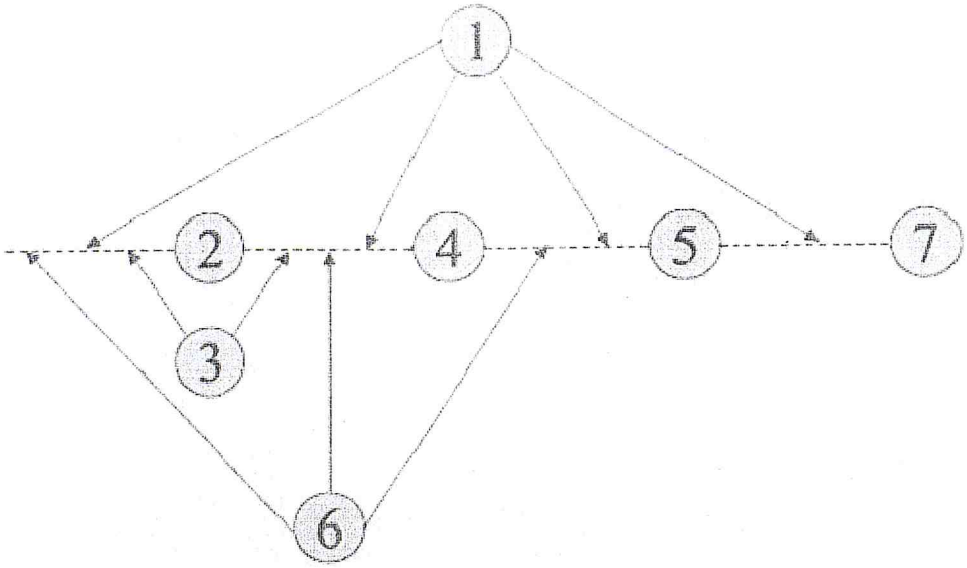



Figure 2.7: Structure des séquences dominantes

Remarquons que la séquence : $(1 \prec 2 \prec 6 \prec 3 \prec 4 \prec 5 \prec 7)$ ne fait pas partie de l'ensemble dominant de séquences puisque le travail 3, n'appartenant qu'à la pyramide $P_1$, ne peut pas être placé après le travail 6 possédant une date de fin plus grande ($d_6 > d_3$).

## 2.3 Condition de dominance pour la minimisation de $\sum U_i$

Dans cette partie, nous nous intéressons à généraliser la condition de dominance d'Erschler et al. au problème de minimisation du nombre de travaux en retard, noté $(1 \mid r_i \mid \sum Ui)$, où $U_i = 1$ si le travail $i$ est en retard (i.e. $t_i + p_i > d_i$). On rappelle que ce problème est NP-difficile [29]. On trouve dans la littérature plusieurs méthodes de résolution exactes basées sur une méthode de séparation et évaluation. Les derniers travaux de recherche traitant de ce problème sont ceux décrits dans [6] et [16]. Les méthodes proposées sont efficaces et peuvent être appliquées à des problèmes comptant plus d'une centaine de travaux, des problèmes de telle taille étant abordés et résolus de façon exacte pour la première fois durant cette décennie. M'Hallah et Bulfin publièrent également les résultats de leurs travaux [30].

Leur méthode très efficace basée sur un Branch and Bound a permis de trouver les solutions optimales pour toutes les instances difficiles n'ayant pas pu être résolues par les approches antérieures.

Toutefois, certains cas particuliers du problème 1 $|r_i| \sum U_i$ sont faciles. Dans le cas particulier où les dates de disponibilité ou les dates d'échéance des travaux sont identiques ($r_i = r$ ou $d_i = d$), le problème noté 1 $|| \sum Ui$ est résolu en $O(nlogn)$ par l'algorithme de Moore [31]. Pour le même problème, avec durées opératoires unitaires, C.L. Monma [32] a proposé un algorithme en $O(n)$. Dans le cas où les dates de disponibilité et les dates d'échéance des travaux sont similairement ordonnés (i.e. $r_i < r_j \Rightarrow d_i \leq d_j$), Kise et al. dans [26] ont proposé un algorithme en $O(n^2)$ pour résoudre le problème. Sous les mêmes conditions E.L.Lawler [27] a décrit, un peu plus tard, un algorithme plus efficace en O($nlogn$). E.L.Lawler dans [28] a également proposé un algorithme en $O(n^5)$ pour le cas préemptif 1 $|r_i, pmtn| \sum U_i$, complexité ramenée en 1999 à $O(n^4)$ par P. Baptise [4].Lorsque les durées opératoires sont identiques, Carlier a démontré dans [11] que le problème 1$|\ p_i = p,\ r_i|\ \sum U_i$ peut être résolu en $O(n^3 logn)$. Pour le problème général où des poids sont associés aux travaux, 1$|\ p_i = p,\ r_i|\ \sum w_i\ Ui$, Baptiste [5] a proposé un algorithme en $O(n^7)$. Récemment, Chrobak et al. ont montré dans [14] que la méthode de carlier [11] n'est pas optimale, les auteurs ont apporté des modifications à la méthode de baptiste [5] et ont proposé un algorithme de résolution en $O(n^5)$ pour le problème 1$|\ p_i = p,\ r_i|\ \sum U_i$ sans poids sur les travaux.

### 2.3.1 Théorème des pyramides et minimisation de $\sum U_i$

Trouver une solution optimale au problème $\sum U_i$ consiste à déterminer une séquence admissible pour une sélection de travaux $E^* \subseteq V$ la plus grande possible. Les travaux de $E^*$ sont donc à l'heure alors que les autres travaux sont tous en retard. Les travaux en retard peuvent être séquencés après ceux de $E^*$, dans n'importe quel ordre et à n'importe quelle date. Lors de la recherche d'une solution admissible à $k$ travaux, les $n - k$ travaux en retard peuvent donc être omis. Étant dominant vis-à-vis de l'admissibilité, le Théorème des pyramides peut alors être appliqué pour caractériser un sous-ensemble de séquences dominantes vis-à-vis du

critère autant de fois qu'il existe de sélections différentes de $k$ travaux. L'union de tous ces sous-ensembles constitue de façon évidente un ensemble de séquences dominantes vis-à-vis du problème $1\,|r_i|\sum U_i$. Appelons $S_{dom}$ cet ensemble.

Pour un problème à $n$ travaux, il existe un nombre gigantesque de sélections différentes de travaux (i.e.$\sum A_n^k$). Il n'est toutefois pas nécessaire d'appliquer le Théorème des pyramides à chaque sélection pour déterminer l'ensemble dominant $S_{dom}$. En effet, la notion de séquence maître-pyramide permet de caractériser un ensemble de séquences dominantes pour un nombre très important de sélections.

**Conclusion:**

Nous avons, dans ce chapitre, introduit le problème étudié et rappelé la condition de dominance utilisée pour sa résolution. Nous savons que la condition donnée par le théorème des pyramides est dominante vis-à-vis de l'admissibilité mais ne l'est pas vis-à-vis de la minimisation du nombre de travaux en retard. Toutefois, il est possible de l'exploiter pour la résolution de notre problème en déterminant des bornes. Le chapitre suivant est consacré à une présentation des modèles PLNE permettant de définir ces bornes.

# Chapitre 3

# Rappel sur la modélisation mathématique du pb 1 $|r_i| \sum U_i$

Pour mieux comprendre les modélisations mathématiques proposées dans le chapitre suivant, nous rappelons dans ce chapitre les formulations PLNE proposées dans [34] pour la résolution du problème 1 $|r_i| \sum U_i$. Pour cela dans une première section, nous parlons du problème de détermination de la séquence la plus dominante. Puis, nous montrons, dans une seconde section, comment le problème peut être exploité pour la résolution de la minimisation du $\sum U_i$.

## 3.1 Problème d'admissibilité

### 3.1.1 Problème mono-pyramidal

Nous supposons ici que la structure d'intervalles associée au problème à résoudre ne contient qu'un seul sommet $t$ et donc, une seule pyramide $P$. Nous montrons dans ce qui suit comment déterminer la séquence de travaux la plus dominante, qui peut se modéliser par un programme linéaire en nombres entiers.

### 3.1.2 Une condition nécessaire et suffisante d'admissibilité

**Définition 3** *[19]Une séquence de travaux $\sigma$ est admissible si et seulement si:*

$$p_i + \sum_{i<k<j} p_k + p_j \leq d_j - r_i, \quad \forall (i,j) \in \sigma \text{ tel que } i \prec j \tag{3.1}$$

Cette condition nécessaire et suffisante se simplifie considérablement, dans le cas du problème mono-pyramide [9], sachant que les séquences dominantes sont de la forme $\alpha \prec t \prec \beta$, où :

– $\alpha$ est l'ensemble des travaux séquencés à gauche du sommet $t$ ordonnés selon l'ordre croissant des $r_i$.

–$\beta$ est l'ensemble des travaux séquencés à droite du sommet $t$ ordonnés selon l'ordre croissant des $d_i$.

On note:

$$\begin{cases} *R^{\max} = \max\limits_{a\in\alpha}(r_t, r_a + p_a + \sum\limits_{k\in\alpha, a<k}(p_k)) : \text{la date de début au plus tôt du sommet } t. \\ *D^{\min} = \min\limits_{b\in\beta}(d_t, d_a - p_a - \sum\limits_{k\in\beta, k<b}(p_k)) : \text{la date de fin au plus tard du sommet } t. \end{cases} \tag{3.2}$$

Dans [34], le théorème suivant est démontré:

**Théorème 4** *[34]Soit $P$ une pyramide de sommet $t$, toute séquence compatible à $\sigma$ de la forme $\alpha \prec t \prec \beta$ est admissible ssi $D^{\min} - R^{\max} - p_t \geq 0$.*

Le théorème 4 admet la condition d'admissibilité. Il permet une comparaison numérique des séquences entre elles, et de déduire une condition numérique de dominance.

### 3.1.3 une condition numérique de dominance

**Corollaire 5** *[34] Soit $P$ une pyramide de sommet $t$, considérons $\sigma_1$ et $\sigma_2$ deux séquences, repectivement de la forme $\alpha_1 \prec t \prec \beta_1$ et $\alpha_2 \prec t \prec \beta_2$. si $D_1^{\min} - R_1^{\max} \geq D_2^{\min} - R_2^{\max}$, alors $\sigma_1$ domine $\sigma_2$ vis-a-vis de l'admissibilité.*

Le corollaire 5 permet une formulation mathématique du problème de recherche d'une séquence admissible sous forme d'un programme linéaire en nombres entiers (PLNE1). En effet, la séquence de travaux qui maximise $D^{\min} - R^{\max}$ domine toutes les autres.

$$\max z = D - R$$

$$\begin{cases} R \geq r_t & & (3.3) \\ R \geq r_i + \sum\limits_{(j \epsilon V \mid \{t\} \setminus r_j \geq ri)} p_j x_j^+ & , \forall i \in V \setminus \{t\} & (3.4) \\ D \leq d_t & & (3.5) \\ D \leq d_i - \sum\limits_{(j \epsilon V \mid \{t\} \setminus d_j \leq di)} p_j x_{,j}^- & , \forall i \in V \setminus \{t\} & (3.6) \\ x_i^- + x_i^+ = 1 & , \forall i \in V \setminus \{t\} & (3.7) \\ x_i^-,\ x_i^+ \epsilon \{0,1\} & , \forall i \in V \setminus \{t\} & \\ D,\ R \epsilon \mathbb{N} & & \end{cases}$$

Dans la formulation ci-dessus, la variable $x_i^+$ (resp $x_i^-$) est égale à 1 si le travail $i$ est séquencé à gauche du sommet (resp à droite du sommet). Les contraintes relatives à $R$, permettent de définir la date de début au plus tôt du sommet de la pyramide, tout en assurant que les travaux placés à gauche du sommet ($i.e., \alpha$) débutent leur exécution après leurs dates de disponibilité. De même, les contraintes relatives à $D$ définissent, quant à elles, la date de fin au plus tard du sommet $t$ tout en assurant que les travaux placés à droite de ce sommet ($i.e., \beta$) finissent leur exécution avant leurs dates échue. Enfin, la dernière contrainte impose à ce que tout travail $i$ de la pyramide $P$ soit séquencé, soit à gauche ou bien à droite du sommet, mais pas dans les deux positions en même temps.

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

### 3.1.4 Un modèle de PLNE pour le problème multi-pyramidal interdépendant

On considère ici le problème particulier $V$ caractérisé par $m$ sommets *i.e* $m$ pyramides, avec l'hypothèse que tout travail non-sommet appartient uniquement à une et une seule pyramide *i.e*, si $i \in P_k \Longrightarrow i \notin P_{k'}$, $k \neq k'$, $\forall k \in \{1, ..., m\}$ et $i \in V$ (les pyramides sont dites interdépendantes).

Ainsi, toute séquence dominante respectant le Théorème des pyramides est de la forme $\alpha_1 < t_1 < \beta_1 < ... < \alpha_m < t_m < \beta_m$, où $t_k$ est le sommet de la pyramide $P_k$, et les travaux de $\alpha_k$ et $\beta_k$ appartiennent uniquement à la pyramide $P_k$.

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

Étant donné une séquence dominante $\sigma$, la condition d'interdépendance entre les sous séquences de $\sigma$ peut être assurée en imposant à chaque travail $j \in P_k$ des contraintes sur les dates de début $r_j$ et de fin $d_j$ de $j$. Ces dates ont été définies par le système (3.2) donné plus haut.

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

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

$$\begin{cases}
R_k \geq r_{tk} & ,\forall k \epsilon [1\ m] & (3.8)\\
R_k \geq r_i + \sum\limits_{(j \epsilon P_k | r_j \geq ri)} p_j x_j^+ & ,\forall k \in [1\ m],\ i \in P_k & (3.9)\\
R_k \geq R_{k-1} + p_{t_{k-1}} + \sum\limits_{(j \epsilon P_{k-1})} p_j x_{,j}^- + \sum\limits_{(j \epsilon P_k)} p_j x_j^+ & ,\forall k \epsilon [2..m] & (3.10)\\
D_k \leq d_{tk} & ,\forall k \epsilon [1m] & (3.11)\\
D_k \leq d_i + \sum\limits_{(j \epsilon P_{k-1} | d_j \leq di)} p_j x_{,j}^- & ,\forall k \in [1\ m],\ i \in P_k & (3.12)\\
D_k \leq D_{k+1} - p_{t_{K+1}} - \sum\limits_{(j \epsilon P_{k+1})} p_i x_j^+ - \sum\limits_{(j \epsilon P_k)} p_j x_{,j}^- & ,\forall k \epsilon [1..m-1] & (3.13)\\
x_i^+ + x_i^- = 1 & ,\forall i \in V \setminus \{1...m\} & (3.14)\\
x_i^+,\ x_i^- \epsilon \{0,1\} & ,\forall\ i \in V \setminus \{1...m\} & \\
D_k, R_k \epsilon \mathbb{Z} & ,\forall k \epsilon [1..m] &
\end{cases}$$

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

Par définition :

$$R_k = \max(r_{t_k},\ eft_{k-1} + \textstyle\sum_{\{j \in \alpha_k\}} p_j, \max\limits_{i \in \alpha_k}(r_i + p_i + \textstyle\sum_{\{j \in \alpha_k \setminus i < j\}} p_j)) \qquad (3.15)$$

où $eft_{k-1}$ définit la date de fin au plus tôt de la sous-séquence $\beta_{k-1}$. Ainsi les contraintes sur (3.8), (3.9) et (3.10) formalisées selon l'équation (3.15) permettent de déterminer la valeur de $R_k$.

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

Par définition :

$$D_k = \min(d_{t_k},\ lst_{k+1} - \textstyle\sum_{\{j \in \beta_k\}} p_j, \min\limits_{i \in \beta_k}(d_i - p_i - \textstyle\sum_{\{j \in \beta_k \setminus j < i\}} p_j)) \qquad (3.16)$$

où $lst_{k+1}$ définit la date de début au plus tôt de la sous-séquence $\alpha_{k+1}$.Ainsi les contraintes (3.11), (3.12) et (3.13) formalisées selon l'équation (3.16) permettent de déterminer la valeur de $D_k$ .

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

### 3.1.5 Un modèle de PLNE pour le problème multi-pyramidal quelconque

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

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

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

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

$$\begin{cases}
R_k \geq r_{tk} & ,\forall k\epsilon[1\ m] & (3.17)\\
R_k \geq r_i + \sum\limits_{(j\epsilon P_k | r_j \geq ri)} p_j x^+_{kj} & ,\forall k \in [1\ m],\ i \in P_k & (3.18)\\
R_k \geq R_{k-1} + p_{t_{k-1}} + \sum\limits_{(j\epsilon P_{k-1})} p_j x^-_{(k-1),j} + \sum\limits_{(j\epsilon P_k)} p_j x^+_{k,j} & ,\forall k\epsilon[2..m] & (3.19)\\
D_k \leq d_{tk} & ,\forall k\epsilon[1m] & (3.20)\\
D_k \leq d_i + \sum\limits_{(j\epsilon P_k | d_j \leq di)} p_j x^-_{k,j} & ,\forall k \in [1\ m],\ i \in P_k & (3.21)\\
D_k \leq D_{k+1} - p_{t_{K+1}} - \sum\limits_{(j\epsilon P_{k+1})} p_j x^+_{(k+1),j} - \sum\limits_{(j\epsilon P_k)} p_j x^-_{k,j} & ,\forall k\epsilon[1..m-1] & (3.22)\\
\sum_{k=u(i)}^{v(i)} x^+_{ki} + x^-_{ki} = 1 & ,\forall i \in V \setminus\{1...m\} & (3.23)\\
x^+_{ki},\ x^-_{ki}\epsilon\{0,1\} & ,\forall\ i \in V \setminus\{1...m\} & \\
D_k, R_k\epsilon\mathbb{N} & ,\forall k\epsilon[1..m] &
\end{cases}$$

Dans le PLNE3, modèle similaire au PLNE2, la variable binaire $x^+_{kj}$ (resp.$x^-_{k,j}$ ) prend la valeur 1 si le travail non-sommet $j$ appartenant à la pyramide $P_k$ : $u(j) \leq k \leq v(j)$, est séquencé dans $\alpha_k$ (resp. $\beta_k$). Les contraintes sur $R_k$ et $D_k$ , i.e. les contraintes (3.17)..(3.22), sont identiques à celles définies dans le PLNE2. Enfin, la contrainte (3.23) spécifique à ce modèle permet d'assurer que tout travail prend une et une seule position dans une pyramide et une seule.

## 3.2 Problème de minimisation du nombre de travaux en retard

Dans cette section, on s'intéresse au problème de minimisation du nombre de travaux en retard. Nous présentons des modèles de programmation qui permettent la détermination d'une borne inférieure et d'une borne supérieure du problème $1|r_i|\sum U_i$. Ces modèles sont des adaptations du PLNE3 donné dans la section précédente.

### 3.2.1 Utilisation de la séquence maître-pyramide pour la modélisation

**Corollaire 6** *[34] Étant donné un problème* $1|r_i| \sum U_i$, *s'il existe une séquence optimale au problème avec tous les travaux sommets ordonnancés à temps, alors la séquence maître-pyramide caractérise toutes les séquences dominantes vis-à-vis du critère* $\sum U_i$.

Dans ce qui suit, nous présentons des formulations mathématiques qui permettent d'obtenir des bornes au problème $1|ri| \sum U_i$.

#### A)Modèle PLNE pour l'obtention d'une borne supérieure

Sous l'hypothèse qu'il existe une solution optimale au problème $1|ri| \sum U_i$ avec tous les sommets ordonnancés à temps, alors la séquence maître-pyramide caractérise toutes les séquences dominantes de la forme: $\alpha_1 \prec t_1 \prec \beta_1 ..... \alpha_m \prec t_m \prec \beta_m$ vis-à-vis du critère $\sum U_i$.

Plus explicitement, en considérant le problème de minimisation du nombre de travaux en retard, un travail j non-sommet peut être séquencé dans $\alpha_K$ ou $\beta_K$ de toute pyramide $P_k$ telle que $u(j) \leq k \leq v(j)$, ou bien pas séquencé du tout, i.e. en retard. Ce dernier cas constitue une possibilité non envisageable dans le cas du problème d'admissibilité. Pour intégrer cette possibilité, il suffit de relâcher la contrainte sur les $x_{ik}^+$ et $x_{ik}^-$ par la formule suivante:

$$\sum_{k=u(i)}^{v(i)} x_{ki}^+ + x_{ki}^- \leq 1$$

De plus, il faut imposer que la séquence trouvée soit admissible. Cela est assuré par le respect de la contrainte suivante :

$$(D_k - R_k - p_{t_k}) \geq 1, \forall k \epsilon [1..m]$$

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

$$\min Z = \sum_{\{j \epsilon V \setminus \{t_1 \ldots t_m\}\}} (1 - \sum_{k=u(i)}^{v(j)} (x_{ki}^+ + x_{ki}^-)) \tag{3.24}$$

Pour formaliser le problème de minimisation du nombre de travaux en retard, il est supposé qu'il existe une solution optimale qui ordonnancerait tous les sommets à l'heure. Évidemment, une telle hypothèse n'est pas forcement réalisable. Trouver une solution optimale dans une telle forme, i.e en imposant l'ordre partiel défini par le Théorème des pyramides, permet alors de déterminer une borne au problème $1|r_i| \sum U_i$. Ainsi PLNE3 (modèle proposé pour le problème d'admissibilité) est adapté pour résoudre le problème de détermination d'une borne supérieure au problème $\sum U_i$. Cette formulation est donnée par le modèle (PLNE4) suivant:

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

$$\begin{cases}
R_k \geq r_{tk} & ,\forall k \epsilon [1..m] & (3.25) \\
R_k \geq r_i + \sum_{(j \epsilon P_k | r_j \geq ri)} p_j x_{kj}^+ & ,\forall k \in [1..m],\ i \in P_k & (3.26) \\
R_k \geq R_{k-1} + p_{t_{k-1}}(1 - y_{tk}) + \sum_{(j \epsilon P_{k-1})} p_j x_{(k-1),j}^- + \sum_{(j \epsilon P_k)} p_j x_{kj}^+ & ,\forall k \epsilon [2..m] & (3.27) \\
D_k \leq d_{tk} & ,\forall k \epsilon [1.m] & (3.28) \\
D_k \leq d_i + \sum_{(j \epsilon P_k | d_j \leq di)} p_j x_{k,j}^- & ,\forall k \in [1..m], \forall i \in P_k & (3.29) \\
D_k \leq D_{k+1} - p_{tK+1}(1 - y_{tk+1}) - \sum_{(j \epsilon P_{k+1})} p_j x_{(k+1),j}^+ - \sum_{(j \epsilon P_k)} p_j x_{k,j}^- & ,\forall k \epsilon [1..m-1] & (3.30) \\
\sum_{k=u(i)}^{v(i)} x_{ki}^+ + x_{ki}^- \leq 1 & ,\forall i \in V \setminus \{1...m\} & (3.31) \\
D_k - R_k \geq p_{tk}(1 - y_{tk}) & ,\forall k \epsilon [1..m] & (3.32) \\
x_{ki}^+,\ x_{ki}^- \epsilon \{0,1\} & ,\forall\, i \in V \setminus \{1...m\} & \\
D_k, R_k \epsilon \mathbb{N} & ,\forall k \epsilon [1..m] &
\end{cases}$$

Les contraintes (3.25) et (3.26) et (3.28) et (3.29.) sont similaires aux premières contraintes dans la mesure où les variables $R_k$ et $D_k$ demeurent inchangées. Permettre à un travail non-sommet d'être en retard est assuré par la contrainte (3.31).

Nous imposons dans la formulation que la séquence optimale soit une séquence dominante compatible avec la séquence maître-pyramide, avec si possible, tous les sommets ordonnancés à l'heure. Cependant, il existe des cas où trouver une solution avec tous les sommets à l'heure est impossible. En effet, si deux sommets $t_k$ et $t_{k+1}$ consécutifs sont tels que $d_{k+1} - r_{tk} < p_{tk+1} + p_{tk}$, alors il est impossible de trouver une solution réalisable qui séquencerait les deux sommets $k$ et $(k+1)$ à l'heure. Aussi, selon la valeur de la durée opératoire des sommets du problème, il est plus avantageux de préférer à un sommet $t_k$, un autre travail non-sommet dans le but de maximiser le nombre de travaux en avance. Pour cela, nous associons pour chaque sommet $k$ une variable $y_{t_k}$ égale à 1 si le sommet est ignoré (sa durée opératoire est mise à zéro), à 0 s'il est à l'heure. Les variables de ce type sont prises en compte dans les contraintes (3.27), (3.30) et (3.32) d'admissibilité de chaque pyramide: Si $y_{t_k}$ est égal à 0 alors la contrainte est similaire à celle du modèle PLNE3, si elle prend la valeur 1, cela implique que les travaux de la pyramide $P_k$ sont à l'heure puisque $D_k - R_k > 0$. Le critère d'optimisation $\sum U_i$ est exprimé par les variables $y_{tk}$, $x^+_{ki}$ et $x^-_{ki}$et permet d'optimiser le nombre de travaux en retards, tous travaux confondus (*sommets* et *non-sommets*), et si $\sum_{k=u(j)}^{v(j)} x^+_{ki} + x^-_{ki} = 0$ le travail non-sommet $j$ n'est attribué à aucune pyramide ou il est en retard.

### B)Modèle PLNE pour l'obtention de bornes inférieures

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

On rappelle que :

**Définition 7** *Un problème à une machine à n travaux est dit à pyramides parfaites, si:*

$$\forall (i,j) \in P_k \times P_k, (r_i \geq r_j) \Longleftrightarrow (d_i \leq d_j)\ \forall k = 1...m.$$

De plus on a le corollaire suivant :

**Corollaire 8** *Étant donné un problème $V$ à n travaux et à pyramides parfaites, la séquence maître-pyramide $\sigma_\Delta(V)$ caractérise toutes les séquences dominantes vis-à-vis du critère $\sum U_i$.*

De là, et sachant d'après le Corollaire 8 que le Théorème des pyramides est dominant vis-à-vis de la minimisation du nombre de travaux en retard pour un problème $1|ri|\sum U_i$ à structure pyramidale parfaite, le problème peut ainsi être résolu optimalement par PLNE5 suivant:

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

$$\begin{cases}
R_k \geq r_{ik} + \mathbf{y}_{t_k}(\mathbf{r}_{n_k} - \mathbf{r}_{t_k}) & ,\forall k \epsilon [1..m] \quad (3.33)\\
R_k \geq r_i + (1 - \mathbf{x}_{ik}^+)(\mathbf{r}_{n_k} - \mathbf{r}_i) + \sum_{(j \epsilon P_k | r_j \geq ri)} p_j x_{kj}^+ & ,\forall k \in [1..m],\ i \in P_k \quad (3.34)\\
R_k \geq R_{k-1} + p_{t_{k-1}}(1 - y_{tk}) + \sum_{(j \epsilon P_{k-1})} p_j x_{(k-1),j}^- + \sum_{(j \epsilon P_k)} p_j x_{kj}^+ & ,\forall k \epsilon [2..m] \quad (3.27)\\
D_k \leq d_{tk} + \mathbf{y}_{t_k}(\mathbf{d}_{n_k} - \mathbf{d}_{t_k}) & ,\forall k \epsilon [1.m] \quad (3.35)\\
D_k \leq d_i + (\mathbf{1} - \mathbf{x}_{ik}^-)(\mathbf{d}_{n_k} - \mathbf{d}_i) + \sum_{(j \epsilon P_k | d_j \leq di)} p_j x_{k,j}^- & ,\forall k \in [1..m], \forall i \in P_k \quad (3.36)\\
D_k \leq D_{k+1} - p_{t_{k+1}}(1 - y_{tk+1}) - \sum_{(j \epsilon P_{k+1})} p_j x_{(k+1)j}^+ - \sum_{(j \epsilon P_k)} p_j x_{k,j}^- & ,\forall k \epsilon [1..m-1] \quad (3.30)\\
& ,\forall i \in V \setminus \{1...m\} \quad (3.31)\\
\sum_{k=u(i)}^{v(i)} x_{ki}^+ + x_{ki}^- \leq 1 & ,\forall k \epsilon [1..m] \quad (3.32)\\
D_k - R_k \geq p_{tk}(1 - y_{tk}) & ,\forall\ i \in V \setminus \{1...m\}\\
x_i^+,\ x_i^- \epsilon \{0,1\} & ,\forall k \epsilon [1..m]\\
D_k, R_k \epsilon \mathbb{N}
\end{cases}$$

Avec:

$\bullet \mathrm{r}_{n_k} = \min(r_{t_k}, \min_{i \in P_k} r_i)$

$\bullet \mathrm{d}_{n_k} = \max(d_{t_k}, \max_{i \in P_k} d_i)$

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

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

Dans le modèle PLNE5, nous recherchons une séquence à $n' \leq n$ travaux à l'heure, et $m$ sommets. Si un sommet $t_k$ ou un travail $j$ est en retard, alors les contraintes sur $R_k$ i.e. (3.33) et (3.34) et celles sur $D_k$ *i.e*, (3.35) et (3.36) du travail correspondant doivent être désactivées. En effet, nous savons que l'inégalité $R_k \geq r_{nk}$ (resp.$D_k \leq d_{nk}$) est évidemment toujours vérifiée. De même, dans le cas où $i$ ne parvient pas à $t_{k-1}$ quand $k = u(i)$ où $i \notin \alpha_k (i \notin \beta_k$ resp.), le terme $(1 - x_{k-1,i})(r_{n_k} - r_i)$ $((1 - x_{ki})(d_{n_k} - d_i)$ resp.) desactives les contraintes (3.34) (la contraintes (3.36) resp.). Notons que la désactivation des contraintes permet de ne considérer que les travaux qui sont à l'heure.

### Relaxation des dates de disponibilité

Si on considère un problème général avec une structure pyramidale quelconque, (i.e. sa structure pyramidale n'est pas parfaite), il est alors possible de diminuer les dates de disponibilité des travaux pour rendre le problème à structure pyramidale parfaite.

L'algorithme de la figure 3.1 décrit la modification des dates de début au plus tôt. Pour l'exécuter, il est supposé que les travaux sont triés selon l'ordre croissant des dates échues et que $j_{n_k}$ est le travail de $P_k$ qui possède la plus grande date échue.

**Proposition 9** *[35] La solution obtenue, en résolvant le problème* $1|ri|\sum U_i$ *par le modèle PLNE5 avec comme dates échues celles obtenues par l'algorithme donné sur la Figure 3.1, est une borne inférieure du problème initial. Cette borne sera appelée* $LB_r$.

Algorithme 1: modification des dates de disponibilité

Pour toute pyramide $P_k(k = 1...m)$ faire

Trier les travaux suivant selon l'ordre croissant des $d_j$

Pour tout travail $j \in P_k$ *faire*

Si $r_j > r_{j-1}$ alors $r_j \leftarrow r_{j-1}$

FinSi

Fin pour tout

Fin pour tout

Figure 3.1: Algorithme de relaxation des dates de disponibilité.

### Relaxation des dates échue

De façon similaire à la relaxation des dates de début au plu tôt, il est également possible d'augmenter les dates échues dans la perspective de transformer le problème à structure pyramidale parfaite. L'algorithme de la figure 3.2 décrit le principe. Son exécution suppose que les travaux sont triés selon l'ordre croissant des dates au plus tôt. Le travail de $P_k$ possèdant la plus grande date de disponibilité.

**Proposition 10** *[35]. La solution obtenue, en résolvant le problème* $1|ri|\sum U_i$ *par le modèle PLNE5 avec comme dates échues celles obtenues par l'algorithme donné sur la Figure 3.2, est une borne inférieure du problème initial. Cette borne sera appelée* $LB_d$.

Algorithme 1: modification des dates d'échéance

Pour toute pyramide $P_k (k = 1...m)$ faire

Trier les travaux suivant selon l'ordre croissant des $r_j$

Pour tout travail $j \in P_k$ *faire*

Si $d_j < d_{j-1}$ alors $d_j \leftarrow d_{j-1}$

FinSi

Fin pour tout

Fin pour tout

Figure 3.2: Algorithme de relaxation des dates d'échéance.

**Conclusion:**

Nous avons, dans ce chapitre, présenté les programmes linéaires en nombres entiers permettant de déterminer des bornes au problème de minimisation du nombre de travaux en retards. Un modèle PLNE4 qui permet de déterminer la solution optimale du problème lorsque les travaux sommets sont tous en avance, et le modèle PLNE5 qui permet de déterminer une borne supérieure au problème, et aussi une borne inférieure lorsque les données du problème sont relâchées. Dans le chapitre suivant, nous allons procéder à un changement de variables qui va permettre une nouvelle formulation des modèles. Aussi, en vue d'étudier la possibilité d'améliorer encore plus les bornes, une notion de poids est introduite.

# Chapitre 4

# Nouvelle approche de la modélisation mathématique

Dans ce chapitre nous présentons une nouvelle formulation du $PLNE$, afin de réduire le nombre de variables binaires sur chaque pyramide.

Ensuite, en considérant le critère de minimisation du nombre de travaux en retard, nous montrons comment adapter les modèles PLNE introduits dans le chapitre précédent pour la détermination d'une borne supérieure et de bornes inférieures au problème $1|r_i|\sum U_i$.

Enfin, nous proposons d'étudier la possibilité d'améliorer la borne supérieure et cela en associant des poids $(w_i)$ à certains travaux.

## 4.1 Modèle PLNE pour le cas général(formule améliorée)

Dans cette section, nous définissons la nouvelle formulation du PLNE3′ qui consiste à remplacer les variables $x^+_{kj}$ et $x^-_{kj}$ par une seule variable $x_{kj}$ sur chaque pyramide et ajouter une pyramide fictive $P_0$($c.f$ figure4.1 ).

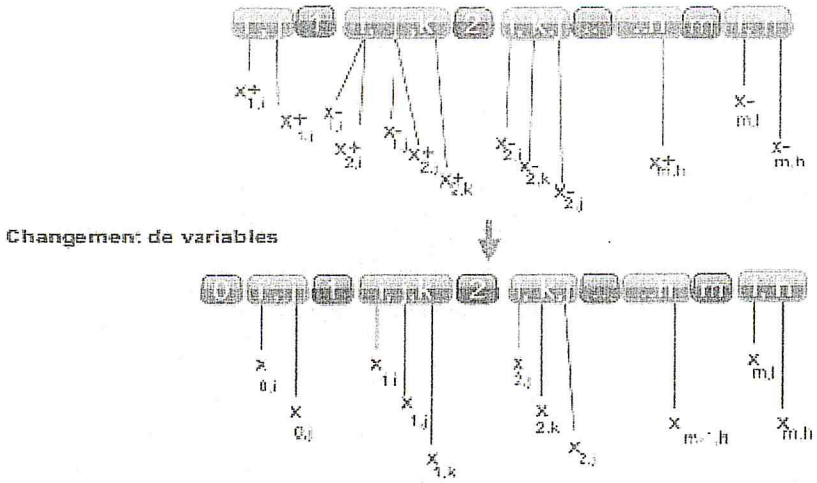



Figure 4.1: Exemple illustratif du changement de variable

Il est formulé comme suit :

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

$$\begin{cases} R_k \geq r_{tk} & ,\forall k \epsilon [1\ m] & (4.1) \\ R_k \geq r_i + \sum\limits_{(j \epsilon P_{k-1} | r_j \geq ri)} p_j x_{k-1,j} & ,\forall i, k = u(i) & (4.2) \\ R_k \geq R_{k-1} + p_{t_{k-1}} + \sum\limits_{(j \epsilon P_{k-1})} p_j x_{k-1,j} + \sum\limits_{(j \epsilon P_k)} p_j x_{k-1j} & ,\forall i, k \epsilon [2..m] & (4.3) \\ D_k \leq d_{tk} & ,\forall k \epsilon [1m] & (4.4) \\ D_k \leq d_i + \sum\limits_{(j \epsilon P_{k-1} | d_j \leq di)} p_j x_{k-1,j} & ,\forall i, k = v(i) & (4.5) \\ D_k \leq D_{k+1} - p_{t_{k+1}} - \sum\limits_{(j \epsilon P_{k+1})} p_j x_{k,j} - \sum\limits_{(j \epsilon P_k)} p_j x_{k,j} & ,\forall i, k \epsilon [1..m-1] & (4.6) \\ \sum_{k=u(i)-1}^{v(i)} x_{ki} = 1 & ,\forall i & (4.7) \\ x_{ki} \epsilon \{0,1\} & ,\forall i, \forall k \epsilon [u(i)-1; v(i)] & \\ D_k, R_k \epsilon \mathbb{Z} & ,\forall k \epsilon [1..m] & \end{cases}$$

Dans le PLNE3, la variable binaire $x_{ki}$ est égale à 1 si la tâche $i$ est séquencée à droite du sommet $k$ ou bien à gauche du sommet $(k+1)$, en supposant qu'un sommet fictif $t_0$ existe permettant ainsi de désigner les tâches séquencées à gauche du sommet $t_1$. Les contraintes sur les dates de début au plus tôt et dates de fin au plus tard d'un sommet $k$ sont données par (4.1)..(4.6). La contrainte (4.7) prend en compte le sommet fictif $t_0$.

La valeur optimale de $z^* = \min_{k=1..m} (D_k - R_k - p_{tk})$ définit la marge libre maximale qu'offre la séquence la plus dominante $\alpha_1^* \prec t_1 \prec \beta_1^* .. \alpha_m^* \prec t_m \prec \beta_m^*$, en considérant que l'ensemble $V$ des travaux est séquencé à l'heure. si la marge libre maximale $z^*$ qu'offre cette séquence est négative, alors il n'existe pas de solution admissible au problème.

## 4.2 Minimisation du nombre de travaux en retard

Dans cette section, nous présentons une nouvelle formulation pour la détermination de bornes supérieures et inférieures pour le problème $1|r_i|\sum U_i$. Ces modèls sont adaptés au PLNE3/ donnée dans la section précédente.

### 4.2.1 Modèle PLNE pour l'obtention d'une borne supérieure

Dans cette sous section, nous présentons le model PLNE4 sous la nouvelle approche avec, pour chaque sommet, une seule variable qui définit la position d'une tâche relativement au sommet.

$$\min z = \sum_{\{i\epsilon V\setminus\{t_1,\dots t_m\}\}}(1 - \sum_{k=u(i)-1}^{v(i)} x_{ki}) + \sum_{k=1}^{m} y_k)$$

$$\begin{cases}
R_k \geq r_{tk} & ,\forall k\epsilon[1..m] \quad (4.1)\\
R_k \geq r_i + \sum_{(j\epsilon P_k | r_j \geq ri)} p_j x_{k-1,j} & ,\forall i, k = u(i) \quad (4.2)\\
R_k \geq R_{k-1} + p_{t_{k-1}}(1 - y_{k-1}) + \sum_{(j\epsilon P_{k-1})} p_j x_{k-1,j} + \sum_{(j\epsilon P_k)} p_j x_{k-1j} & ,\forall i, k\epsilon[2..m] \quad (4.10)\\
D_k \leq d_{tk} & ,\forall k\epsilon[1..m] \quad (4.4)\\
D_k \leq d_i + \sum_{(j\epsilon P_k | d_j \leq di)} p_j x_{k,j} & ,\forall i, k = v(i) \quad (4.5)\\
D_k \leq D_{k+1} - p_{t_{k+1}}(1 - y_{k+1}) - \sum_{(j\epsilon P_{k+1})} p_j x_{k,j} - \sum_{(j\epsilon P_k)} p_j x_{k,j} & ,\forall i, k\epsilon[1..m-1] \quad (4.11)\\
\sum_{k=u(i)-1}^{v(i)} x_{ki} \leq 1 & ,\forall i \quad (4.12)\\
D_k - R_k \geq p_{t_k}(1 - y_k) & ,\forall k\epsilon[1..m] \quad (4.13)\\
y_k\epsilon\{0,1\} & ,\forall k\epsilon[1..m]\\
x_{ki}\epsilon\{0,1\} & ,\forall i, \forall k\epsilon[u(i)-1; v(i)]\\
D_k, R_k\epsilon\mathbb{Z} & ,\forall k\epsilon[1..m]
\end{cases}$$

### 4.2.2 Modèle PLNE pour l'obtention d'une borne inférieure

Dans cette sous section, nous présentons le model PLNE5 sous la nouvelle approche avec une seule variable. Ce modèle permet d'avoir une borne inférieure lorsque les valeurs d'une instance sont relâchées.

$$\min z = \sum_{\{i\epsilon V\setminus\{t_1,\dots t_m\}\}}(1 - \sum_{k=u(i)-1}^{v(i)} x_{ki}) + \sum_{k=1}^{m} y_k)$$

$$\begin{cases}
R_k \geq r_{t_k} + \mathbf{y_k}(\mathbf{R_0} - \mathbf{r_{tk}}) & ,\forall k\epsilon[1..m] \quad (4.14)\\
R_k \geq r_i + (\mathbf{1} - \mathbf{x_{k-1i}})(\mathbf{R_0} - \mathbf{r_i}) + \sum_{(j\epsilon P_k | r_j \geq ri)} p_j x_{k-1,j} & ,\forall i, k = u(i) \quad (4.15)\\
R_k \geq R_{k-1} + p_{t_{k-1}}(1 - y_{k-1}) + \sum_{(j\epsilon P_{k-1})} p_j x_{k-1,j} + \sum_{(j\epsilon P_k)} p_j x_{k-1j} & ,\forall i, k\epsilon[2..m] \quad (4.10)\\
D_k \leq d_{tk} + \mathbf{y_k}(\mathbf{D_{m+1}} - \mathbf{d_{tk}}) & ,\forall k\epsilon[1..m] \quad (4.16)\\
D_k \leq d_i + (\mathbf{1} - \mathbf{x_{ki}})(\mathbf{D_{m+1}} - \mathbf{d_i}) + \sum_{(j\epsilon P_k | d_j \leq di)} p_j x_{k,j} & ,\forall i, k = v(i) \quad (4.17)\\
D_k \leq D_{k+1} - p_{t_{k+1}}(1 - y_{k+1}) - \sum_{(j\epsilon P_{k+1})} p_j x_{k,j} - \sum_{(j\epsilon P_k)} p_j x_{k,j} & ,\forall i, k\epsilon[1..m-1] \quad (4.11)\\
\sum_{k=u(i)-1}^{v(i)} x_{ki} \leq 1 & ,\forall i \quad (4.12)\\
D_k - R_k \geq p_{t_k}(1 - y_k) & ,\forall k\epsilon[1..m] \quad (4.13)\\
y_k\epsilon\{0,1\} & ,\forall k\epsilon[1..m]\\
x_{ki}\epsilon\{0,1\} & ,\forall i, \forall k\epsilon[u(i)-1; v(i)]\\
D_k, R_k\epsilon\mathbb{Z} & ,\forall k\epsilon[1..m]
\end{cases}$$

Avec:

- $\mathrm{R_0} = \min_{j\epsilon V} r_j$;
- $\mathrm{D_{m+1}} = \max_{j\epsilon V} d_j$.

La différence de ce modèle et celui du chapitre précédent se situe en premier par les termes en gras, ici on choisit des valeurs fixes au lieu des valeurs variables (cf. chapitre3) pour chaque pyramide, soit, minimum des $r_i$ (resp maximum $\mathrm{d_i}$) pour tous les travaux, et en second par le nombre des variables.

**Exemple:** *Nous donons un exemple illustratif afin de bien compredre la modélisation des PLNE (4 et 5).*

| 6 | | Nombre de travaux | |
|---|---|---|---|
| tâches | $r_i$ | $p_i$ | $d_i$ |
| 1 | 8 | 1 | 10 |
| 2 | 19 | 1 | 21 |
| 3 | 5 | 1 | 22 |
| 4 | 6 | 1 | 12 |
| 5 | 7 | 1 | 14 |
| 6 | 18 | 1 | 25 |

Figure 4.2: Structure d'une instance de donnée à traiter.

Considérons un ensemble $V$ de 6 travaux (cf. Fig 4.2). Sur cet exemple, on identifie les sommets: $s1 = 1$ et $s2 = 2$ qui carectérisent respectivement les pyramides: $P_{s1} = \{3, 4, 5\}$ et $P_{s2} = \{3, 6\}$.On a le modèle PLNE 4 suivant:

$$obj \quad : \quad \min(1 - x_{03} - x_{13} - x_{23}) + (1 - x_{04} - x_{14}) + (1 - x_{05} - x_{15}) + (1 - x_{16} - x_{26}) + y_1 + y_2$$

$$PLNE \ 4 \begin{cases} R_1 \geq r_{t1} \\ R_1 \geq r_3 + p_3 x_{03} + p_4 x_{04} + p_5 x_{05} \\ R_1 \geq r_4 + p_4 x_{04} + p_5 x_{05} \\ R_1 \geq r_5 + p_5 x_{05} \\ R_2 \geq r_{t2} \\ R_2 \geq r_6 + p_6 x_{16} \\ R_2 \geq R_1 + p_{t1}(1 - y_1) + (p_3 x_{13} + p_4 x_{14} + p_5 x_{15}) + p_6 x_{16} \\ D_2 \leq d_{t2} \\ D_2 \leq d_6 - p_6 x_{26} - p_3 x_{23} \\ D_2 \leq d_3 - p_3 x_{23} \\ D_1 \leq d_{t1} \\ D_1 \leq d_4 - p_4 x_{14} \\ D_1 \leq d_5 - p_4 x_{14} - p_5 x_{15} \\ D_1 \leq D_2 - p_{t2}(1 - y_2) - p_6 x_{16} - (p_3 x_{13} + p_5 x_{15} + p_4 x_{14}) \\ x_{03} + x_{13} + x_{23} \leq 1 \\ x_{04} + x_{14} \leq 1 \\ x_{05} + x_{15} \leq 1 \\ x_{16} + x_{26} \leq 1 \\ D_1 - R_1 \geq p_{t1}(1 - y_1) \\ D_2 - R_2 \geq p_{t2}(1 - y_2) \\ x_{ki}, y_k \in \{0, 1\}, \qquad D_k, R_k \in \mathbb{Z} \end{cases}$$

Et le model PLNE 5 suivant:

$$obj \quad : \quad \min(1 - x_{03} - x_{13} - x_{23}) + (1 - x_{04} - x_{14}) + (1 - x_{05} - x_{15}) + (1 - x_{16} - x_{26}) + y_1 + y_2$$

$$PLNE\ 5 \begin{cases} R_1 \geq r_{t1} + y_1(R_0 - r_{t1}) \\ R_1 \geq r_3 + (1 - x_{03})(R_0 - r_3) + p_3 x_{03} + p_4 x_{04} + p_5 x_{05} \\ R_1 \geq r_4 + (1 - x_{04})(R_0 - r_4) + p_4 x_{04} + p_5 x_{05} \\ R_1 \geq r_5 + (1 - x_{05})(R_0 - r_5) + p_5 x_{05} \\ R_2 \geq r_{t2} + y_2(R_0 - r_{t2}) \\ R_2 \geq r_6 + (1 - x_{06})(R_0 - r_6) + p_6 x_{16} \\ R_2 \geq R_1 + p_{t1}(1 - y_1) + (p_3 x_{13} + p_4 x_{14} + p_5 x_{15}) + p_6 x_{61} \\ D_2 \leq d_{t2} + y_2(D_{m+1} - d_{t2}) \\ D_2 \leq d_6 + (1 - x_{06})(D_{m+1} - d_6) - p_6 x_{26} - p_3 x_{23} \\ D_2 \leq d_3 + (1 - x_{03})(D_{m+1} - d_3) - p_3 x_{23} \\ D_1 \leq d_{t1} + y_1(D_{m+1} - d_{t1}) \\ D_1 \leq d_4 + (1 - x_4)(D_{m+1} - d_4) - p_4 x_{14} \\ D_1 \leq d_5 + (1 - x_{05})(D_{m+1} - d_5) - p_4 x_{14} - p_5 x_{15} \\ D_1 \leq D_2 - p_{t2}(1 - y_2) - p_6 x_{61} - (p_3 x_{13} + p_5 x_{15} + p_4 x_{14}) \\ x_{03} + x_{13} + x_{23} \leq 1 \\ x_{04} + x_{14} \leq 1 \\ x_{05} + x_{15} \leq 1 \\ x_{16} + x_{26} \leq 1 \\ D_1 - R_1 \geq p_{t1}(1 - y_1) \\ D_2 - R_2 \geq p_{t2}(1 - y_2) \\ x_{ki}, y_k \in \{0, 1\}, \quad D_k, R_k \in \mathbb{Z} \end{cases}$$

## 4.3 Modèle avec poids associés aux variables sommets et non-sommets.

Au vu de tester l'efficacité des modèles proposés, des expérimentations sont réalisées et données dans le chapitre suivant. Voulant améliorer les résultats obtenu, et sachant que les sommets ont des positions fixes dans la séquences maître-pyramide, des poids ont été associés aux travaux sommets au niveau du critère afin de forser des position en retards aux sommets. Ainsi des poids sont associés aux variables sommets et non-sommets donnés comme suit :

La fonction objective du modèle PLNE5 devient:

$$\min z = \sum_{\{i \epsilon V \setminus \{t_1 .... t_m\}\}} w_{x_i}(1 - \sum^{v(i)} x_{ki}) + w_{y_i}(\sum_{k=1}^{m} y_k)$$

telque, $w_{x_i} = m + 10$ et $w_{y_i} = m$; *où* m: nombre de sommets.

Les expérimentations réalisés dans ce cas sont données et commentées dans le chapitre suivant.

**Conclusion :**

Nous avons dans ce chapitre proposé de nouveaux modèles de programmations pour la détermination de bornes aux solutions optimales, et nous nous sommes proposés d'étudier la possibilité d'améliorer la qualité des solutions en associant des poids aux travaux.

Dans le but d'expérimenter la qualité des modéles proposés dans ce chapitre, des implémentations sur les modeles PLNE proposés réalisées et décrites dans le chapitre qui suit.

# Chapitre 5

# Implémentation et Résolution

Les formulations mathèmatiques de progammation linéaire en nombres entiers proposés pour la résolution du problème $1|ri|\sum U_i$ (PLNE 4 et PLNE5) présentées dans le chapitre 4 ont été implémentées par le solveur ILOG CPLEX. Pour cela nous avons dévloppé une application sur $C^{++}$ afin de traiter les données et les rendre exploitable par le solveur.

## 5.1 Présentation du ILOG CPLEX

L'environnement de développement intégré (IDE) est utile pour la construction, le débogage et l'ajustement des modèles d'optimisation pour les problèmes de planification et d'ordonnancement. Il utilise les deux moteurs d'optimisation ILOG CPLEX et ILOG CP Optimizer.

**ILOG Optimization Decision Manager (ODM):**

C'est une application spécialisée pour la création interactive de tableaux de bord et de graphiques d'aide à la dicision. Il fait appel à ILOG OPL pour la résolution.

**ILOG CPLEX Concert Technology**

Nom : ILOG CPLEX = C simplex (Compagnie américaine rachetée par ILOG à partir de la version 6 .3) est un moteur de programmation mathématique, offrant des algorithmes de

résolution de problème d'optimisation mathématiques:

- programmes linéaires.
- programmes linéaires en nombres entiers.
- programmes quadratiques à contraintes linéaires.
- génération de colonnes.
- programmes quadratiques à contraintes quadratiques.

Ce produit offre un bon niveau de fonctionnalité et de flexibilité nécessaires pour le développement de modèle d'optimisation pour les languages: C, JAVA, . NET. et $C^{++}$, ce dernier étant utilisé pour nos expérimentations.

## 5.2 Implémentation

Les programmmes linéaires PLNE4 et PLNE5 ont été implémentés directement sur le solveur ILOG OPL IDE.

Pour cela nous avons mis en place une application sur $C^{++}$ pour traiter nos données et les rendre exploitable par l'IDE.

Les étapes du programme sont les suivantes:

- **Première étape** : consiste à récupérer à partir d'une instance, les données du problème depuis un fichier source, structuré comme suit: le nombre $n$ de tâches est représenté sur la première ligne, ce qui nous donne la taille du problème. Puis on trouve autant de lignes que de tâches où sont définies une date de disponibilité à gauche, une durée opératoire au centre et une date échéance à la droite, comme décrit sur la figure 4.2 pour un exemple de 6 travaux.

Pour cet exemple, en considérant les indices des tâches dans l'ordre de leur parution dans le fichier, les sommets sont les tâches $t_1 = 1$, $t_2 = 2$, caractérisant les deux pyramides $P_1$= { 3 ,4,5} et $P_2$= {3 ,6}.

- **Deuxième étape**: consiste à définir à partir de l'instance de donnée la structure d'intervalle correspondante en spécifiant les tâches sommets ($m$ étant le nombre de sommets) ainsi que les autres tâches constituant les $m$ pyramides. Pour cela deux petits algorithmes simples sont utilisés. Le premier parcours les tâches une par une et effectue des tests par rapport à toutes les autres tâches afin d'identifier les sommets puis les marques. Ensuite le deuxième algorithme parcours toutes les tâches non-marquées, les compare aux sommets (tâches marquées) et les affecte à ses sommets si les conditions d'appartenance de la tâche à la pyramide relative au sommet sont valides.

- **La dernière étape**: est l'écriture des données dans un fichier nommé "résultat" et servant de base pour sa résolution par le solveur CPLEX. Il est représenté pour notre exemple par la figure 5.1 dont la structure est la suivante:

$m = 2;$
$R_0 = 5;$
$D_{m+1} = 25;$

$$
\begin{aligned}
nonsommet &= \{ \\
&< 3,1,2,1,5,22 > \\
&< 4,1,1,1,6,12 > \\
&< 5,1,1,1,7,14 > \\
&< 6,2,2,1,18,25 > \\
&\quad \};
\end{aligned}
$$

$$
\begin{aligned}
sommet &= \{ \\
&< 1,1,1,8,10 > \\
&< 2,2,1,19,21 > \\
&\quad \};
\end{aligned}
$$

$$
\begin{aligned}
pyrset &= [ \\
&\quad [111] \\
&\quad [110] \\
&\quad [110] \\
&\quad [011] \\
&\quad ];
\end{aligned}
$$

Figure 5.1: Structure du fichier résultat.

Nous allons détailler dans ce qui suit la structure de données illustrée sur la figure 5.1.

- $n = 2$ (nombre de sommet), $R_0 = \min r_i = 2$ et $D_{m+1} = \max d_i = 25$.

- Le premier ensemble *nonsommet* $= \{\ldots\}$ constitue l'ensemble des tâches non-sommet avec les caractérisations suivantes (voir la figure 5.2): le nombre de lignes représente le nombre de tâches non-sommets constituants les différents pyramides, ce nombre est égal à $n - m$ donnés par ordre croissant de leur date de disponibilité. On remarque dans ce cas que les tâches sont déja données dans l'ordre de leur indices dans l'instance de données. Sur chaque ligne, on peut lire, dans l'ordre, l'identifiant de la tâche $i$, l'indice $(u_i)$ de la première pyramide à laquelle appartient la tâche, l'indice $(v_i)$ de la dernière pyramide à laquelle appartient la tâche, sa durée opératoire $p_i$, sa date de disponibilité $r_i$ et sa date échue $d_i$. Selon la figure 5.2, si on considère la tâche 6, on lit respectivement, $u_i = v_i = 2$ (la tâche appartient uniquement à la pyramide 2), sa durée opératoire est de 1, sa date de disponibilité est de 18 alors que son date échue est de 25 .

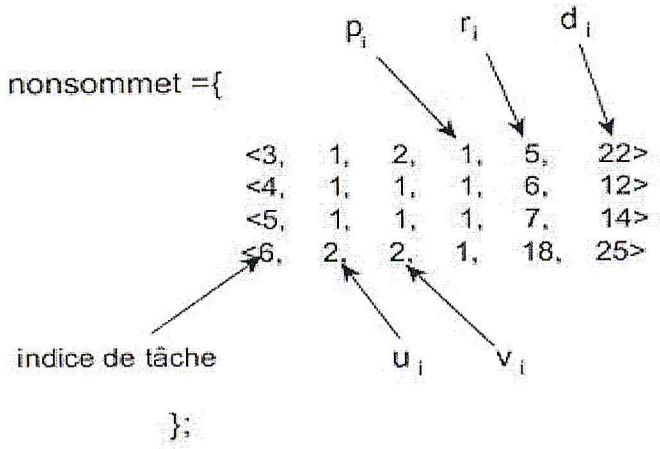



Figure 5.2: Structure des tâches non sommets dans le fichier résultat.

- Le second ensemble *sommet* $= \{\ldots\}$ de la figure 5.3 est l'ensemble des $m = 2$ sommets representant l'ensemble des pyramides, cf. figure 5.3. Pour plus de détails, sur chacune des $m$ lignes, on peut lire en premier l'indice $i$ de chaque pyramide ensuite l'indice du sommet de la pyramide $P_i$, puis la durée opératoire du sommet, sa date de disponibilité et enfin sa date échue.

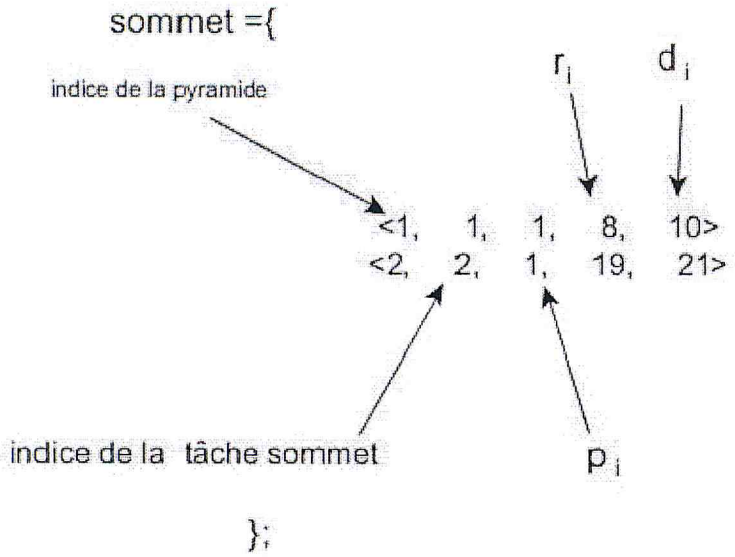



Figure 5.3: Structure des sommets dans le fichier résultat.

- Dans le dernier bloc de la figure 5.4, on a la matrice pyrset [...] de taille $(n-m)*(m+1)$ en zéros-uns, $(n-m)$ étant le nombre de tâches non-sommet du problème et $(m+1)$ au début, on avait parlé de $t_0$ comme tâches fictive. C'est vrai qu'on a $m+1$ sommets avec le sommet fictif est une pyramide fictive permettant de positionner les dernières tâches non-sommet à gauche de son sommet, cf. figure 5.4. Chaque ligne correspondant à une tâche donnée, et pour chaque colonne correspondant à un sommet en commençant par 1 jusqu'à $m+1$ ici $m=2$, on lit la valeur 1 si la tâche est placée à gauche de $k$, 0 sinon. Par exemple sur la 2ème ligne de la figure 5.4 correspondant à la tâche 4, on peut lire que la tâche est séquencée à gauche et à droite de la pyramide 1 et que la tâche 3 appartient à la pyramide 1 et 2, sa variable de position prend des valeurs 1 dans tous les cas c-à-d qu'elle peut se mettre à gauche du sommet 3 (à droite du sommet 2) alors que la tâche 4 peut se mettre seulement à gauche du sommet 1 ou 2 ce qui explique que le $x_{4,3}$=0 sur la $3^{ième}$ colonne.

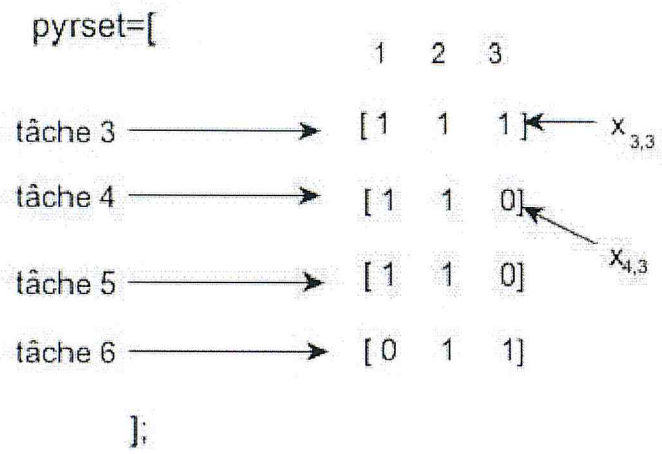



Figure 5.4: Affectation des tâches selon la séquence dominante dans le fichier résultat.

le problème est exécuté sur une machine TOSHIBA Satelelite L750-21C munie d'un processeur Intel(R) Core(TM) i5-2450M CPU@2.50GHz, et dotée d'une Ram de 4 GO, sous système d'exploitation Windows7 64 bits.

Nous aurons les résultats suivant :

$$
\begin{aligned}
& \text{// } \textit{solution (optimale with objective } Z = 0) \\
X \;=\; & [[1\ 0\ 0] \\
& [1\ 0\ 0] \\
& [1\ 0\ 0] \\
& [0\ 1\ 0]]; \\
Y \;=\; & [0\ 0]; \\
R \;=\; & [9\ 20]; \\
D \;=\; & [10\ 21];
\end{aligned}
$$

Et les statistiques suivantes:

- un temp d'éxecution: $0s$
- contraintes: 20

– Variables:16

- Binaire:11

- Entier: 4

- Autres: 1

A titre de comparaison, les résultats suivants sont issues de l'implémentation du modèle PLNE4 avec les variables de l'implémentation du model PLNE4 (avec $x^{+} et\ x^{-}$)

– un temp d'éxecution: $0s$

– contraintes: 22

– Variables: 23

- Binaire: 18

- Entier: 4

- Autres: 1

Nous remarquons, que notre modèle est meilleur relativement au nombre de contraintes et au nombre de variables binaires et cela est due au changement des vaiables $x_{ik}^{+}$ *et* $x_{ik}^{-}$ par une seule variable $x_{ik}$.

## 5.3 Résultats des expérimentations

### 5.3.1 Cas des modèles sans poids

Dans cette section, une serie d'experimentations a été realisée sur un peu plus de 495 problèmes testés, dans le but d'appécier la qualité de notre modèle. Aprés avoir testé le modèle sur de petits problèmes, pour vérifier son bon fonctionnement (section 5.2), nous l'avons ensuite testé sur les instances de Baptiste [6] ayant entre 80 et 160 travaux.

- 120 instances à 80 tâches.
- 120 instances à 100 tâches.
- 120 instances à 120 tâches.
- 120 instances à 140 tâches.

- 15 instances à 160 tâches.

Pour chaque instance testée, nous nous sommes intéressées à déterminer:

- une borne supérieure UB par la résolution du modèle PLNE4 et au temps CPU correspondant;
- une borne supérieure UB par la résolution du modèle PLNE5 et au temps CPU correspondant;
- une borne inférieure par la résolution du modèle PLNE5 en relaxant les dates au plus tôt $r_i(LB_r)$ et le temps CPU de résolution;
- une borne inférieure par la résolution du modèle PLNE5 en relaxant les dates de fin au plus tard $d_i(LB_d)$ et le temps CPU de résolution.

A titre d'exemple, le tableau 5.1 donne un aperçu de quelques résultats trouvés pour deux instances de 80, 100, 120, 140 et 160.

$LB_r$ et $LB_d$ représentent les bornes inférieures. $LB_r$ est le résultat obtenu en appliquant le PLNE5 sur l'instance en relâchant les dates de disponibilité pour rendre le problème parfait alors que $tLB_r$ représente son temps d'exécution requis. De même pour $LB_d$ est obtenu en appliquant le PLNE5 mais cette fois en relaxant les dates d'échéance, avec $tLB_d$ représentant son temps d'exécution.

$UB_{PLNE4}$ et $UB_{PLNE5}$ représentent les bornes supérieures. Pour chaque instance nous avons donc $UB_{PLNE4}$ (resp.$UB_{PLNE5}$) qui est le résultat obtenu en résolvant le PLNE4 (resp. PLNE5) avec leur temps d'exécution $tUB_{PLNE4}$ (resp. $tUB_{PLNE5}$). Notons que nous nous sommes limités à une heure de calcul et que nous avons retenu la borne $UB_{PLNE5}$ comme borne supérieures qui est de meilleure qualité même si son temps d'exécution et plus long.

L'optimum est atteint lorsque l'une des bornes supérieures $UB$ est égale à une des bornes inférieures $LB_r$ ou $LB_d$. Pour une instance donnée, par exemple l'instance 14 à 80 travaux, la même borne inférieure ($LB_r = LB_d = 12$) est obtenue pour un temps de calcul de moins

| Instances | $LB_r$ | $tLB_r$ | $LB_d$ | $tLB_d$ | $UB_{PLNE5}$ | $tUB_{PLNE5}$ | $UB_{PLNE4}$ | $tUB_{PLNE4}$ |
|---|---|---|---|---|---|---|---|---|
| P80_1 | 21 | 0,45 | 21 | 0,53 | 21 | 0,56 | 22 | 0,05 |
| P80_14 | 12 | 0,12 | 12 | 0,95 | 12 | 1,11 | 12 | 0,81 |
| P100_5 | 43 | 2946,16 | 42 | 158,73 | 42 | 14,09 | 49 | 0,14 |
| P100_32 | 7 | 1,98 | 7 | 1,51 | 7 | 0,44 | 7 | 0,11 |
| P120_1 | 18 | 0,62 | 17 | 0,33 | 17 | 0,25 | 18 | 0,01 |
| P120_56 | 8 | 2,15 | 8 | 2,29 | 8 | 0,81 | 9 | 0,09 |
| P140_12 | 52 | 1035,22 | 51 | 3600 | 53 | 590,37 | 55 | 1,08 |
| P140_31 | 4 | 0,30 | 6 | 0,63 | 6 | 1,03 | 6 | 0,34 |
| P160_1 | 31 | 1,65 | 31 | 1,48 | 31 | 1,04 | 36 | 0,08 |
| P160_15 | 35 | 724,83 | 33 | 934,35 | 35 | 722,43 | 35 | 0,48 |

Table 5.1: Comparaison des résultats(Borne) pour un échantillon.

de 1s. Les deux bornes supérieures, ici égale à la borne inférieure, est obtenue en un temps de calcul moins de 2s, nous pouvons conclure sur l'optimalité pour cette instance. Par contre pour l'instance 12 de 140 tâches, nous remarquons que le calcul des bornes inférieures s'est arrêté, $LB_r$ dépasse les $3600s$ fixé, et $LB_r$ aprés $1035,22$ $s$ de calcul déclanche l'erreur $\ll$ *out of Memory* $\gg$, qui veut dire que toute la mémoire est utilisée où le calcule qui s'est arrêté pour cette instance et l'optimalité qui n'est donc pas atteinte. Dans ce qui suit, nous rapportons dans plusieurs tableaux, une synthèse des resultats trouvés pour l'ensemble des instances testées.

Le Tableau 5.2 synthétise les résultats des PLNE ayant permis la détermination des différentes bornes supérieures et inférieures, à savoir, le pourcentage des instances résolues optimalement en moins d'une heure. En voulant analyser la qualité du temps de calcul pour la borne supérieure $UB_{PLNE\,5}$, nous avons calculé le pourcentage d'instances résolues optimalement en se limitant à chaque fois à des temps de calcul inférieurs respectivement à 1s, 60s et 3600s. Vu que le nombre d'instances résolu en moins d'une minute est important (voir les 3 dernières colonnes), nous avons arrêté la borne $UB_{PLNE\,5}$ comme borne supérieure afin de la comparer, comme nous le verrons par la suite, à la meilleure borne inférieure.

| | instances résolues | | $UB_{PLNE5}$ | | |
|---|---|---|---|---|---|
| | $UB_{PLNE4}$ | $UB_{PLNE5}$ | $tUB_{PLNE5} < 1_s$ | $tUB_{PLNE5} < 60_s$ | $tUB_{PLNE5} < 3600_s$ |
| N=80 | 100% | 97% | 37% | 89% | 97% |
| N=100 | 100% | 95% | 40% | 87% | 94% |
| N=120 | 97% | 90% | 16% | 73% | 90% |
| N=140 | 97% | 77% | 8% | 59% | 77% |
| N=160 | 100% | 27% | 7% | 20% | 27% |

Table 5.2: Pourcentage des instances résolues pour les bornes supérieures.

| | $LB_r$ | | $LB_d$ | | $\max(LB_r, LB_d)$ |
|---|---|---|---|---|---|
| | %instances résolues ($tLB_r<3600_s$) | la moyenne du teps de calcul | % instances résolues ($tLB_r<3600_s$) | la moyenne du teps de calcul | % instances résolues ($tmax(LB_r,LB_d<3600_s$) |
| N=80 | 99% | 44,94 | 97% | 47,74 | 99% |
| N=100 | 93% | 123,9 | 95% | 103,64 | 94% |
| N=120 | 91% | 200,15 | 92% | 139,18 | 87% |
| N=140 | 73% | 206,98 | 73% | 240,61 | 69% |
| N=160 | 27% | 121,16 | 27% | 39,76 | 27% |

Table 5.3: Pourcentage des instances résolues pour les bornes inférieures.

Dans le Tableau 5.3, nous retrouvons une analyse des bornes inférieures ($LB_r$ et $LB_d$). Les colonnes 2 et 3 pour $LB_r$, le pourcentage d'instances résolus en moins d'une heure ainsi que la moyenne du temps de calcul. De même pour $LB_d$. Alors que sur la dernière colonne, on peut lire le pourcentage des instances résolues optimalement en considérant à la fois les deux bornes, i.e. le maximum entre ($LB_r$ et $LB_d$)

Dans le tableau 5.4, nous retrouvons le pourcentage des solutions optimalement atteintes en comparant la borne supérieure respectivement aux bornes inférieures $LB_r$ et $LB_d$. La 4ème colonne du Tableau 5.4 illustre les écarts entre la borne supérieure et la meilleure borne inférieure trouvée $LB$; nous remarquons que l'écart n'est pas important et donc les valeurs des deux bornes sont proches de l'optimum. La 5éme colonne du Tableau 5.4 illustre le pourcentage des solutions optimales trouvées pour les instances à 80, 100, 120, 140 et 160 tâches qui sont respectivement 95%, 91%, 91%, 83% et 75%, ainsi que les temps de calcul moyen correspondants.

Enfin une comparaison entre les résultats que nous avons trouvé et les résultats de Baptiste est donnée dans le tableau 5.4. Nous constatons que les résultats des expérimentations

| | Notre approche | | | | | Baptiste | |
|---|---|---|---|---|---|---|---|
| | UB=LB$_r$ % | UB-LB$_d$ % | UB=LB % | UB-LB % | Tcpu | UB=LB % | Tcpu |
| N=80 | 74% | 76% | 0,0504 | 95% | 33,70$_s$ | 76,7% | 117,3$_s$ |
| N=100 | 76% | 70% | 0,0869 | 91% | 67,16$_s$ | 90,00% | 273,5$_s$ |
| N=120 | 64% | 62% | 0,0877 | 91% | 123,68$_s$ | 84,20% | 538,2$_s$ |
| N=140 | 64% | 63% | 0,1684 | 83% | 139,79$_s$ | 72,50% | 1037,3$_s$ |
| N=160 | 50% | 75% | -0,25 | 75% | 105.1$_s$ | – | – |

Table 5.4: Pourcentage des instances résolues optimalement.

sont trés bons, notament comparés au temps d'execution et aux tésultats de Baptiste obtenus pour les même problèmes. Il démontrent l'intérêt de notre formulation et son efficacité. La borne supérieure (PLNE5) et les bornes inférieures sont en particulier très bonnes.

### 5.3.2 Cas des modèles avec poids

Comme nous l'avons introduit dans le chapitre précédent, nous avons associé des poids au niveau du critères en vue d'analyser au mieux le problème et voir l'influence des positions des sommets dans la séquence optimale. Ainsi une série d'expérimentations a été réalisée sur une série d'instances, les résultats étant données sur le table 5.5.

D'après les résultats des expérimentations données par la figure 5.5 et illustrées plus explicitement sur des graphes, on peut remarquer que pour toutes les instances (80 et 100 tâches) la valeurs de $y_w$ (nombre de travaux sommets en retard) se rapproche plus vers la solution optimale, et ceci constitue bien notre objectif dans la mesure où on voulait forcer les tâches sommets à être en retard lorsqu'un choix peut se faire entre une tâche sommet et une tâche non-sommet (voir graphes 5.3 et 5.4). D'autres part, et d'après les graphes 5.1 et 5.2, on remarque bien en terme de temps d'exécution, que pour certaines instances, le temps requis a considérablement diminué lorsque des poids sont associés aux tâches, alors que pour d'autres instances, il a au contraire augmenté. Afin de mieux analyser les résultats les écarts moyens suivants sont calculées :

Travaux de 80 tâches:

En considérant toutes les instances de 80 tâches on trouve un écart moyen entres les

| | 80 *Tâches* | | | | | | |
|---|---|---|---|---|---|---|---|
| | | *Sans w* | | | Avec $w$ | | |
| Instances | Sommet | $UB_{PLNE\ 5}$ | Y | T CPU | $UB_{PLNE\ 5}$ | $Y_w$ | T CPU |
| P80-3 | 50 | 30 | 18 | 72,73 | 30 | 25 | 55,08 |
| P80-4 | 52 | 34 | 19 | 111,32 | 34 | 28 | 25,46 |
| P80-5 | 49 | 30 | 13 | 12,17 | 30 | 17 | 23,31 |
| P80-9 | 52 | 22 | 10 | 9,27 | 22 | 17 | 14,79 |
| P80-12 | 46 | 38 | | 3129,93 | 38 | 32 | 5,40 |
| P80-64 | 62 | 34 | 28 | 20,13 | 34 | 31 | 58,14 |
| P80-66 | 50 | 43 | 41 | 117,38 | 43 | 43 | 2,87 |
| P80-70 | 65 | 30 | 21 | 62,63 | 30 | 25 | 261,32 |
| P80-87 | 45 | 20 | 8 | 1048,08 | 20 | 14 | 344,45 |
| P80-98 | 39 | 11 | 4 | 85,73 | 11 | 10 | 817,60 |
| | 100 *Tâches* | | | | | | |
| P100-2 | 72 | 34 | 25 | 82,87 | 34 | 34 | 618,11 |
| P100-3 | 72 | 34 | 21 | 14,05 | 34 | 25 | 106,2 |
| P100-9 | 66 | 22 | 11 | 48,16 | 22 | 14 | 136,45 |
| P100-11 | 51 | 36 | 14 | 102,51 | 36 | 19 | 382,41 |
| P100-15 | 62 | 19 | 10 | 256,01 | 19 | 15 | 335,50 |
| P100-44 | 49 | 13 | 6 | 27,71 | 13 | 10 | 308,60 |
| P100-63 | 82 | 38 | 34 | 69,06 | 38 | 37 | 574,24 |
| P100-66 | 69 | 53 | 36 | 1769,66 | 53 | 52 | 18,66 |
| P100-71 | 57 | 43 | 23 | 267,15 | 44 | 39 | 84,83 |
| P100-81 | 56 | 33 | 23 | 840,43 | 26 | 23 | 200,32 |

Table 5.5: comparaisons de 80 et 100 tâches pour 10 instances

temps CPU d'exécution avec et sans poids égal à $\sum(T-T_w)/n = 306,095$ lorsqu'on considère uniquement les instances donnant de bonnes solutions ($Tw < T$), l'écart moyen donné par : $\sum(T - T_w)/n = 725,359$ est bien considérable.

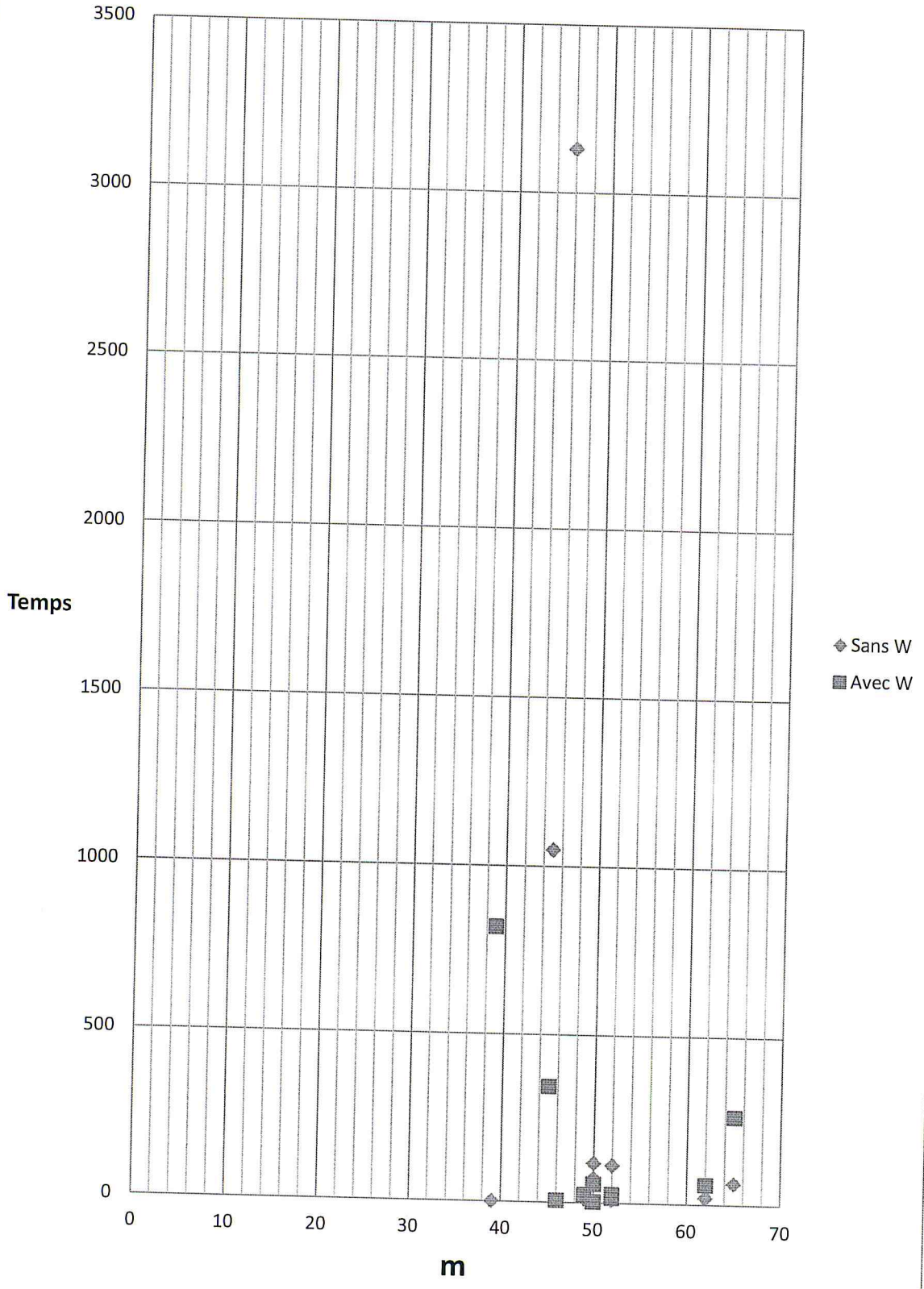
Considérant les instances de 100 tâches l'écart moyen entres les temps d'exécution en considérant les deux critères avec et sans poids on trouve :

un écart moyen $\sum(T - T_w)/n = -13,77$, cet écart moyen est négatif mais si on ne considères que les bonnes instances dont les temps ont diminué en associant des poids aux tâches ($Tw < T$), on remarque que l'écart moyen donné par $\sum(T - T_w)/n = 1751$ est très significatif.

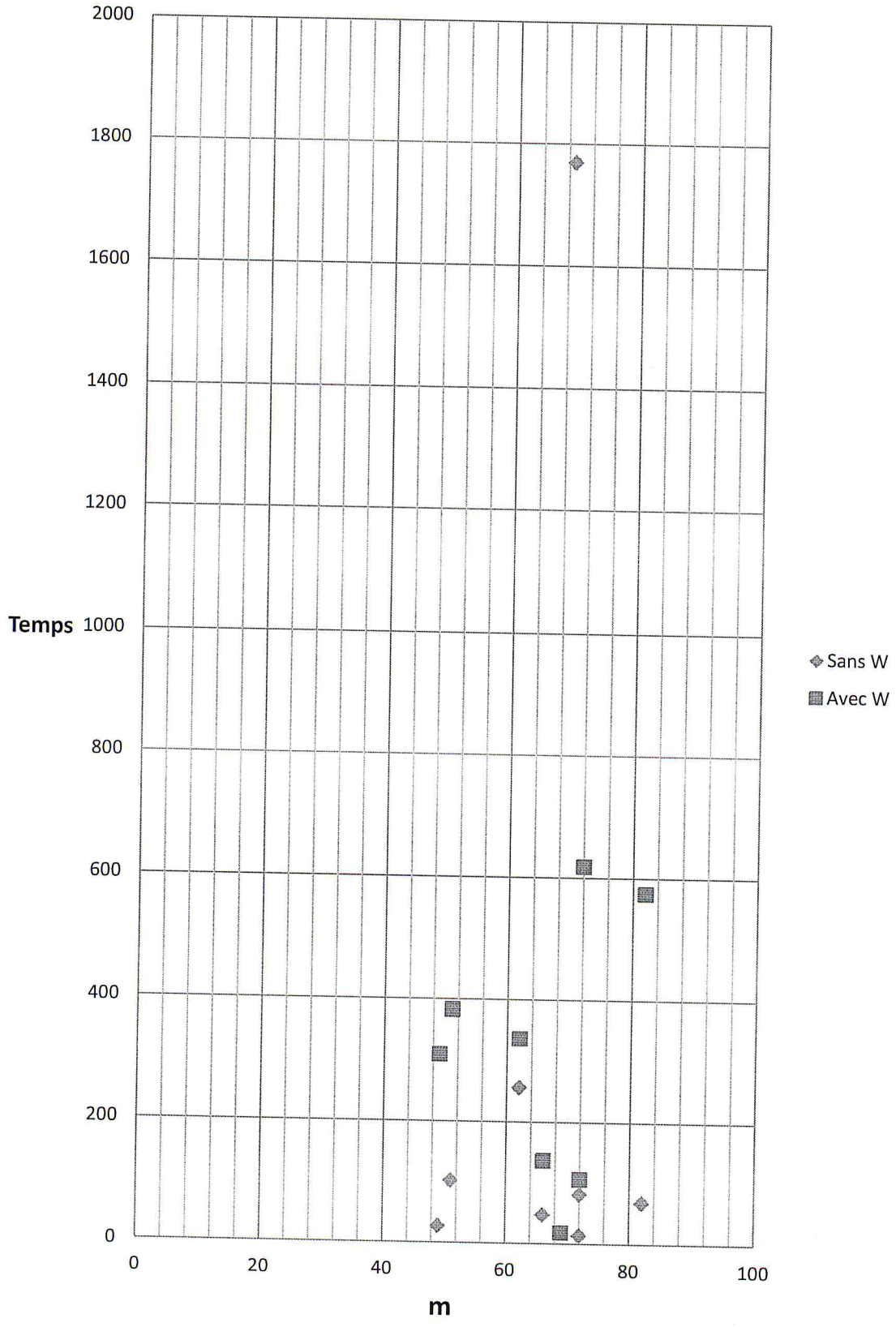
Nous remarquons, que pour certaines instances l'influence des poids à été remarquable voir (figure 5.5, Graphes 5.1, 5.2) les instance 12, 66, 87 de 80 tâches et 66, 81, 71 de 100 tâches, indique la rapidité de la résolution du modéle.

Les résultats des expérimentations illustrés sur les graphes 5.3, 5.4 montrent que pour toutes les instances de 80 et 100 tâches la valeure de $y_w$ augmente vers la solution optimale d'où le but de sérrer les sommets afin de les rendre en retard.

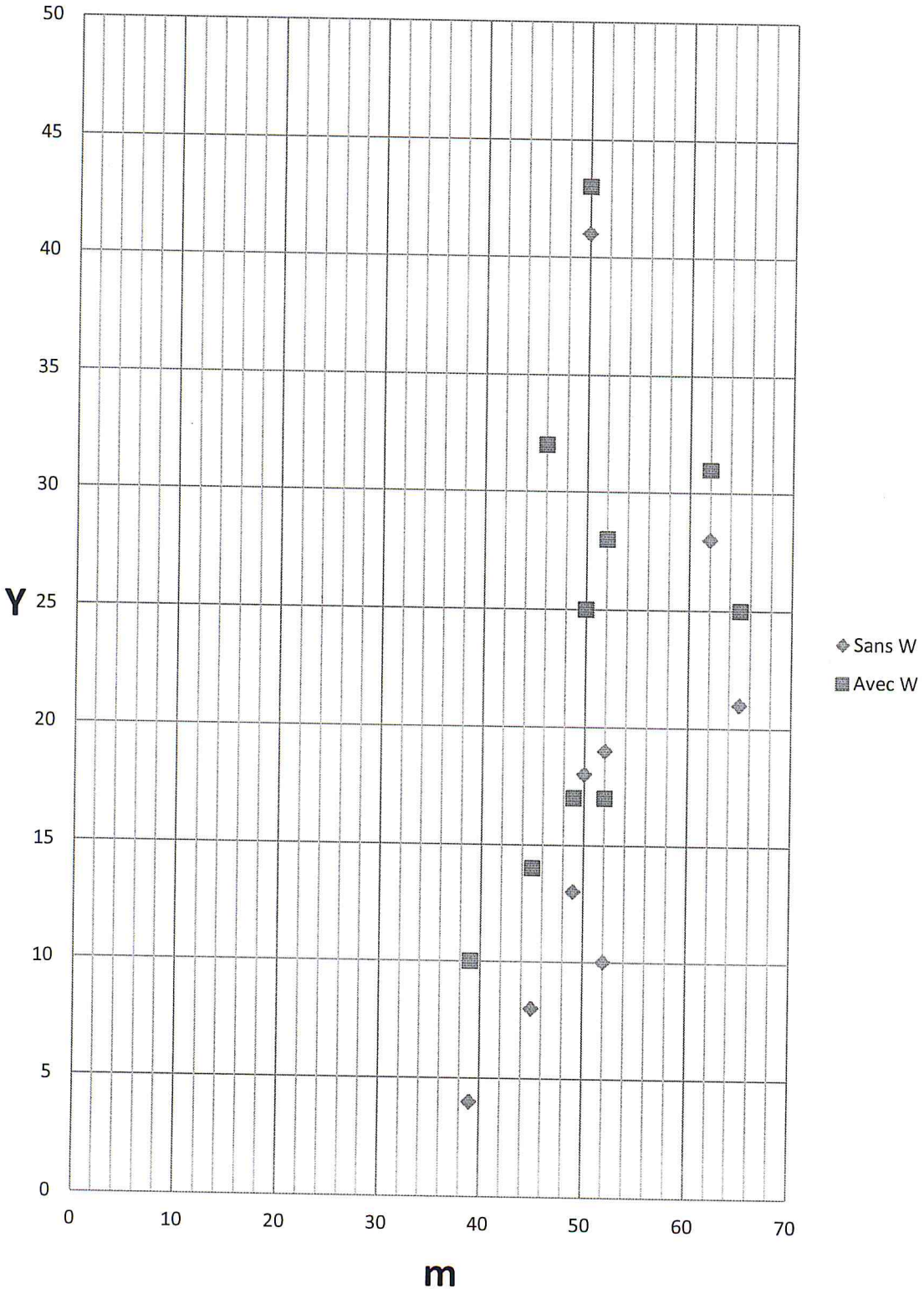
**Graphe 5.1 Impact des poids associés pour 10 instances de 80 Tâches**



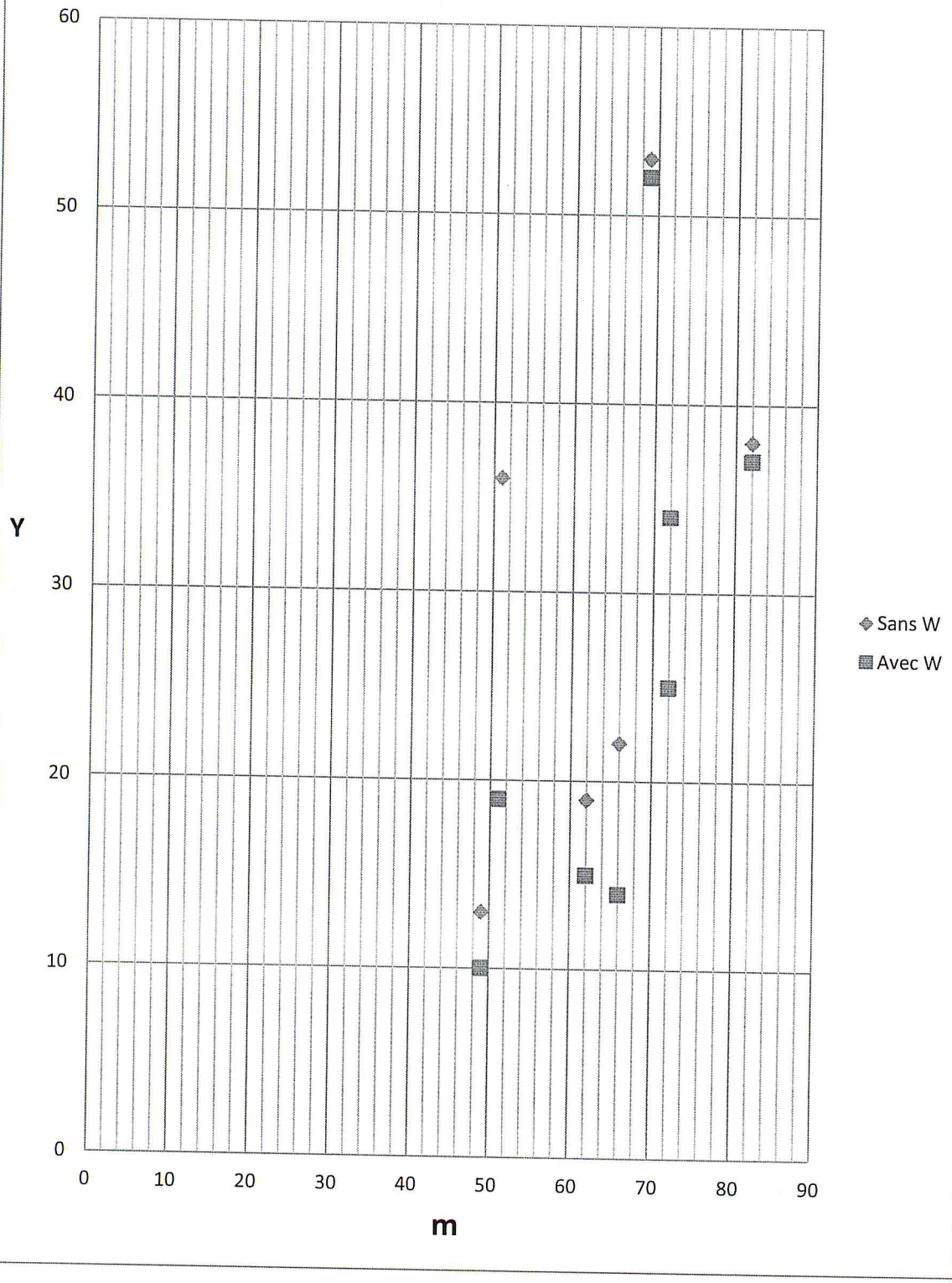
**Graphe 5.2 Impact des poids associés pour 8 instances de 100 Tâches**



**Graphe 5.3 Impact des poids associés pour les valeurs de Y de 80 tâches à 10 instances.**



**Graphe 5.4 Impact des poids associés pour les valeurs de Y de 100 tâches à 8 instances.**



# Conclusion et perspectives

La recherche de solution optimale pour le problème d'ordonnancement à une machine avec fenêtres d'exécutions est fortement combinatoire. C'est en particulier le cas lorsque le critère envisagé est la minimisation du nombre de travaux en retard.

Dans ce mémoire, et à partir de certains travaux déja réalisés sur l'exploitation de la condition de dominance pour la résolution d'un problème d'optimisation $1|ri|\sum U_i$ sous forme de PLNE que nous avons exploité, nous avons introduit le problème étudié et rappelé la condition de dominance utilisée pour sa résolution. Nous savons que cette condition donnée par le théorème des pyramides est dominante vis-à-vis de l'admissibilité mais ne l'est pas vis-à-vis de la minimisation du nombre de travaux en retard. Toutefois, il est possible de l'exploiter pour la résolution de notre problème en déterminant des bornes à travers des modèles PLNE adaptés de celui donné pour la détermination de la solution la plus dominantes.

Ainsi, nous avons présenté les programmes linéaires en nombres entiers permettant de déterminer des bornes au problème de minimisation du nombre de travaux en retards.

Un premier modèle PLNE4 permettant de déterminer la solution optimale au problème lorsque les travaux sommets sont tous en avance et constitue ainsi une borne supérieure, et un second modèle PLNE5 qui permet de déterminer une autre borne supérieure, plus efficace en terme de valeur de la solution retournée. Aussi à travers le même modèle PLNE5, une borne inférieure est déterminée lorsque les données du problème sont relâchées. A la base de ces modèles et au vue daméliorer la qualité des solutions ainsi que leur temps dexécution, nous nous sommes proposés de procéder à un changement de variables qui a conduit à une nouvelle formulation des modèles. D'un autre coté, afin de pouvoir étudier la possibilité daméliorer encore plus les bornes, une notion de poids a été introduite et associée aux variables de décisions.

Les formulations mathématiques de programmation linéaire en nombres entiers proposées pour le problème $1|ri|\sum U_i$ ont été implémentées sur le solveur ILOG CPLEX 6.3. Pour cela nous avons mis en en place une application sur C$^{++}$ afin de traiter les données et de les rendre exploitable par le solveur. Une comparaison avec les résultats de l'approche [6] a

montré que les résultats des expérimentations sont trés bons, notamment du point de vue du temps d'exécution et de la qualité des bornes inférieures et supérieures. Il démontrent l'intérêt de notre nouvelle formulation et son éfficacité.

Dans notre étude menée dans ce mémoire, nous avons proposé de nouveaux modèles de programmations pour la détermination de bornes aux solutions optimales, et nous nous sommes proposés d'étudier la possibilité d'améliorer la qualité des solutions en associant des poids aux travaux. Nous avons ainsi touché de près à la notion de dominance et aux modèles de programmation linéaire en variables mixtes et à leur résolution à travers un solveur. Les différentes connaissances acquises nous ont été bénéfiques et nous ont inspiré, en plus du travail réalisé, pour mettre des perspectives de recherche à mener dans les travaux futurs. La plus intéressante, consiste de poursuivre l'étude sur l'intérêt d'associer des poids aux tâches, vue que l'expérimentation de certaines instances a révélé un temps d'exécution très réduit, et faire une analyse plus approfondie sur la structure pyramidale d'une instance pour statuer sur la nécessité d'associer des poids ou pas au problème considéré.

# Bibliographie


[1] Airzem M, Emkiedeche Y, Ourari S & Boudhar M. Méthode exacte basée sur le con-cept de dominance pour minimiser le nombre de tâches en retard, cas d'une ressource. Thèse d'ingéniorat, USTHB, Alger, Algerie, 2006.

[2] J. F. Allen. Time and time again: The many ways to represent time. International Journal of Intelligent Systems, vol. 6, no. 4, pages 341-355, 1991.

[3] R. N. Antony, "Planning and Control Systems: A Framework for Analysis", Harvard University Press, Cambridge, Massachusetts, 1965.

[4] P. Baptiste. A $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research Letters, vol. 24, pages 175-180, 1999.

[5] P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. Journal of Scheduling, vol. 2, pages 245-252, 1999.

[6] P. Baptiste, L. Peridy & E. Pinson. A Banch and Bound to Minimize the Number of late Jobs on a Single Machine with Release Time Constraints. European Journal of Operations Research,vol. 144, no. 1, pages 1-11, 2003.

[7] J. Benassy, "La Gestion de la Production", Hermès, Paris, 1987.

[8] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt & J. Weglarz. Scheduling in computer and manufacturing processes. Springer Verlag, 1996.

[9] C.Briand and S.Ourari.Minimizing the number of late job in single machine sheduling with nested execution intervals. IEEE, Troyes, France, 2006.

[10] C. Briand, S. Ourari & B. Bouzouia. An efficient ILP formulation for the single machine scheduling problem. RAIRO-Operation Research, vol. 44, no. 1, pages 61-71, 2010.

[11] J. Carlier. Problèmes d'ordonnancement à durées égales. QUESTIO, vol. 5, no. 4, pages 219-228, 1981.

[12] J. Carlier. The one-machine sequencing problem. European Journal of Operational Research, vol. 11, pages 42-47, 1982.

[13] J.Carlier & P.Chrétienne. Problèmes d'ordonnancement: modélisation/complexité/algorithmes. Masson, 1988.

[14] M. Chrobak, C. Dürr, W. Jawor, L. Kowalik & M. Kurowski. A Note on Scheduling Equal-Length Jobs to Maximize Throughput. Journal of Scheduling, vol. 9, no. 1, pages 71-73, 2006.

[15] G.B. Dantzig & P. Wolfe. Decomposition principle for linear programs. Operations Research, vol. 8, pages 101-111, 1960.

[16] S. Dauzère-Pérès and M. Sevaux. An exact method to minimize the number of tardy jobs in single machine scheduling. Journal of Scheduling, vol. 9,no. (6), pages 405-420, 2004

[17] M. Dorigo. Optimisation, Learning and Natural Algorithms. Politechnico di milano, Université Paul Sabatier, 1992.

[18] G. Doumeights, D. Breuil et L. Pun, "La gestion de production assistée par ordinateur", Hermès, Paris, 1983.

[19] J.Erschler, F. Roubellat, J. Vernhes. Characterizing the set of feasible sequences for n jobs to be carried out on single machine. European Journal of Operational Research, vol. 144: 1-11, 1980.

[20] J. Erschler, G. Fontan, C. Merce & F. Roubellat. A New Dominance Concept in Scheduling n Jobs on a Single Machine with Ready Times and Due Dates. Operations Research, vol. 31, no. 1, pages114-127, 1983.

[21] P. Esquirol & P. Lopez. L'ordonnancement. Economica, 1999.

[22] M. R. Garey, David S. Johnson: The Complexity of Near-Optimal Graph Coloring. J. Assoc. Comp. Mach vol. 23, pages43-49 (1976).

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

[24] J .H . Holland. Adaptation in natural and artificial systems. Rapport technique, Ann Arbor, University of Michigan, 1975.

[25] H.T. La. Utilisation d'ordres partiels pour la caractérisation de.solutions robustes en ordonnnancement. Thèse de doctorat, INSA,Toulouse, France, 2004.

[26] H. Kise, T. Ibaraki, et H. Mine. A solvable case of the one-machine scheduling problem with ready and due times. Operations Research, vol. 26, pages 121 126, 1978.

[27] E. L. Lawler. Scheduling a single machine to minimize the number of late jobs. Rapport technique, Preprint. Computer Science Division, University of California, Berkeley, 1982.

[28] E. L. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Annals of Operations Research, vol 26, pages 125-133. 1990.

[29] J. K. Lenstra, A. H. G. Rinnoy Kan, et P. Brucker. Complexity of machine scheduling problems. Annuals of Discrete Mathematics, vol. 1, pages 343-362, 1977.

[30] R. M'Hallah & R. L. Bulfin. Minimizing the weighted number of lardy jobs on a single machine with release dates. European Journal of Operational Research, vol. 176, no. 1, pages 727-744, 2007.

[31] J. M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. Management Science, vol. 15, pages 102-109, 1968.

[32] C. L. Monma. Linear time algorithms for scheduling on parallel processors. Operations Research, vol. 30, pages 116-124, 1982.

[33] S. Ourari & C. Briand. Conditions de dominance pour le problème à une machine avec minimisation des travaux en retard. Dans 9ème congrès de la société française de Recherche Opérationnelle et d'aide à la décision. (ROADEF08 ), Clermont-Ferrand, France, 2008.

[34] S. Ourari, C. Briand & B. Bouzouia. Minimizing the number of tardy jobs in single machine scheduling using MIP. Dans MISTA09, Dublin, Irland, 2009.

[35] S. Ourari. De l'ordonnancement déterministe à l'ordonnancement distribué sous incertitude. Thèse de doctorat, UPS, Toulouse, France, 2011.

[36] M. Pinedo. Scheduling: Theory, algorithms and systems (2nd edition). Prentice Hall, 2008.