UNIVERSITÉ FRANÇOIS-RABELAIS DE TOURS

ÉCOLE DOCTORALE : SANTÉ, SCIENCE, ET TECHNOLOGIES
Laboratoire d'Informatique

THÈSE présentée par :

Nguyen HUYNH TUONG

soutenue le : 17 juin 2009

pour obtenir le grade de : Docteur de l'Université François-Rabelais de Tours

Discipline/Spécialité : INFORMATIQUE

COMPLEXITÉ ET ALGORITHMES POUR L'ORDONNANCEMENT MULTICRITERE DE TRAVAUX INDÉPENDANTS : PROBLÈMES JUSTE-À-TEMPS ET TRAVAUX INTERFÉRANTS

THÈSE dirigée par :

SOUKHAL Ameur	Maître de conférences, Université François Rabelais de Tours
BILLAUT Jean-Charles	Professeur, Université François Rabelais de Tours

RAPPORTEURS :

BAPTISTE Philippe	Chargé de Recherche CNRS, HDR (Paris)
CHRÉTIENNE Philippe	Professeur, Université Pierre et Marie Curie (Paris VI)

JURY :

AGNETIS Alessandro	Professeur, Université de Sienne, Italie
BAPTISTE Philippe	Chargé de Recherche CNRS, HDR (Paris)
BILLAUT Jean-Charles	Professeur, Université François Rabelais de Tours
CARLIER Jacques	Professeur, Université de Technologie de Compiègne
CHRÉTIENNE Philippe	Professeur, Université Pierre et Marie Curie (Paris VI)
NERON Emmanuel	Professeur, Université François Rabelais de Tours
SOUKHAL Ameur	Maître de conférences, Université François Rabelais de Tours

À ma famille : mes parents, ma femme et ma fille

Remerciements

Les travaux réalisés au cours de cette thèse ont été effectués au sein du Laboratoire d'Informatique de l'Université de François Rabelais de Tours (EA 2101), dans l'équipe Ordonnancement et Conduite.

Je souhaite tout en premier lieu remercier Ameur Soukhal, Maître de Conférences à l'Ecole Polytech'Tours, qui a encadré cette thèse. Pendant ces trois années, il a su orienter aux bons moments mes travaux de recherches en me faisant découvrir l'ordonnancement au travers de son regard novateur et critique. Ses conseils et ses commentaires précieux m'ont permis de surmonter les difficulté et de progresser.

Je tiens à exprimer mes remerciements à Jean-Charles Billaut, Professeur de l'Ecole Polytech'Tours et également mon directeur de thèse, pour ses encouragements, ses conseils et sa confiance.

J'adresse tous mes sincères remerciements à Philippe Chrétienne, Professeur de l'Université Pierre et Marie Curie (Paris VI), et à Philippe Baptiste, Professeur de l'École Polytechnique (LIX), qui m'ont fait l'honneur d'accepter d'être rapporteurs de mes travaux.

Mes chaleureux remerciements s'adressent à Jacques Carlier, Professeur de l'Université de Technologie de Compiègne d'avoir accepté d'être examinateur et président du jury.

Je remercie également les autres membres du jury qui ont accepté de juger ce travail : Alessandro Agnetis, Professeur de l'Université de Siena (Italy) et Emmanuel Néron, Professeur de l'Université François Rabelais de Tours.

Je remercie le Ministère de l'Éducation et la Recherche pour le financement qui m'a été accordé pour le bon déroulement de ma thèse.

Plus largement, je voudrais remercier les différentes personnes du Laboratoire d'Informatique et du Département Informatique de Polytech'Tours auprès desquelles je suis souvent venue chercher conseil et avec lesquelles j'ai partagé de très bons moments, tant pour le travail que pour des instants de détente. Grâce à un partage et une bonne convivialité entre les membres de l'équipes, j'ai eu l'opportunité d'effectuer ma thèse dans d'excellente condition. Parmi ceux qui ont contribué à mon travail, je remercie tout spécialement Jean-Louis Bouquard et Vincent T'Kindt pour les conseils scientifiques et leur supports. Dans ces remerciements je n'oublie jamais les autres membres de l'équipe qui ont fait de mon séjour, une période très agréable et enrichissante. Je pense en particulier à Christian Proust (Directeur de l'Ecole Polytech'Tours), Patrick Martineau, Christophe Lenté, Claudine Tacquard, Carl Esswein, Geoffey Vilcot, Cédric Pessan, Mathieu Pérotin, Cédric Mocquillon, Mathieu Rouleau, Yanick Kergosen, Gaël Sauvanet, et Rabah Belaid. Mes remerciements sont adressés également aux étudiants du cycle d'ingénieur qui sont intervenus dans mes projets de recherche : Brien Lit-

teaut, Paul Vignard, Corentin Del'homme, Dan Shao, Zangou Dao, Laurent Miscopein et Daudé Guillaume.

Enfin, mes remerciements vont à ma femme My-Dung ainsi qu'à nôtre petite fille Gia-An, qui sont à mes côtés depuis le début et qui m'ont toujours écoutées et accompagnées dans cette aventure avec beaucoup de patience et surtout d'amour.

Résumé

Nous abordons dans cette thèse des problèmes d'ordonnancement de travaux indépendants sur une machine ou sur des machines parallèles. Plus précisément, nous abordons deux catégories de problèmes :

1. les problèmes d'ordonnancement de type juste-à-temps : il s'agit de déterminer un ordonnancement de sorte que les travaux se terminent le plus près possible de leur date de fin souhaitée. On considère le cas où la date de fin souhaitée commune est connue et le cas où elle est à déterminer. De nouveaux algorithmes exacts sont proposés - gloutons et programmes dynamiques -. Des schémas d'approximation sont élaborés.
2. les problèmes d'ordonnancements de travaux interférants : il s'agit de déterminer un ordonnancement qui permet d'optimiser un critère pour la globalité des travaux à effectuer, sachant que la solution trouvée doit permettre également l'optimisation d'un autre critère défini uniquement sur un sous-ensemble des travaux. Il s'agit ici d'un nouveau problème d'ordonnancement multicritère, différent de la notion classique, et qui se rapproche des problèmes de type "*multi-agent scheduling*" ou "*interfering job sets*". Les approches considérées pour trouver une solution non dominée sont l'approche ε-contrainte, la combinaison linéaire de critères et le *goal programming*. De nouveaux résultats de complexité sont montrés et des algorithmes polynomiaux/pseudo-polynomiaux sont développés pour le calcul de cette solution non dominée.

Mots-clés : ordonnancement, une machine, machines parallèles, juste-à-temps, travaux interférants, complexité, programmation dynamique, schéma d'approximation

Abstract

In this thesis we consider scheduling problems of independent jobs on a single machine or on parallel machines. More precisely, we tackle two kinds of problems :

1. just-in-time scheduling problems : it aims to determine a schedule so that a job completes as close as possible to its due date. We consider the case where the common due date is known and the case where the common due date has to be fixed. New exact algorithms based on greedy algorithms and dynamic programming are proposed. Approximation schemes are given.
2. scheduling problems with interfering jobs : the aim is to determine a schedule that optimizes a criterion for the whole set of jobs and so that the solution optimizes another objective only for a subset of jobs. It is here a new multi-criteria scheduling problem, different from the classical notion, which is related to "multi-agent" or to "interfering job sets" scheduling problems. The approaches considered for finding a non-dominated solution are the ε-constraint approach, the linear combination of criteria and the goal programming approach. New complexity results are proposed and polynomial/pseudo-polynomial algorithms are developed for the calculation of the non-dominated solution.

Keywords : scheduling, single machine, parallel machines, just-in-time, interfering jobs, complexity, dynamic programming, approximation scheme

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Motivée et stimulée par des questions qui se posent dans la planification et la gestion de la production, la théorie de l'ordonnancement est devenue un domaine important d'optimisation combinatoire, situé à l'interface entre les mathématiques appliquées, l'informatique et la recherche opérationnelle. Dans son sens le plus large, l'ordonnancement consiste à programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leur date d'exécution. Pour plus de détails sur la théorie d'ordonnancement nous pouvons citer les livres de Carlier et Chrétienne [31], de Leung [155], de Pinedo [177], de Brucker [29], de T'Kindt et Billaut [209], de Blazewicz et al. [27] et le livre d'exercices du groupe de travail GOThA [79].

Depuis les années 50 du dernier siècle, les problèmes d'ordonnancement ont évolué et leurs études n'ont pas cessé d'augmenter, peut-être du fait de leur applicabilité à une multitude croissante de problèmes réels. De plus, parmi les problèmes d'optimisation combinatoire, les problèmes d'ordonnancement sont probablement ceux qui sont le plus liés au monde industriel, puisqu'il s'agit de concilier, de façon optimale, des ressources limitées avec des activités dans le temps. Ainsi, des domaines comme l'éducation, l'agriculture, le transport ou la santé ont bénéficié des recherches issues de l'ordonnancement. Grâce aux progrès énormes des ordinateurs, les problèmes d'ordonnancement approchent de plus en plus les problèmes réels, qui attirent les chercheurs par leur intérêt et leur difficulté [184].

On constate que *le client* prend une importance considérable dans les organisations et dans les systèmes de production en particulier. L'organisation des chaînes logistiques repose sur une coordination des acteurs allant des fournisseurs des fournisseurs jusqu'au client final. Les problèmes de synchronisation et de livraison nécessitent une coordination parfaitement réglée, qui peut être obtenue grâce à des études sur le juste-à-

temps (JàT) (*just-in-time* en anglais). D'autre part, la prise en compte des particularités des clients – certains sont plus exigeants sur les délais, certains sur les quantités, certains souhaitent grouper leurs livraisons, d'autres non, etc. – nécessite l'introduction de paramètres et de mesures dédiés à certains travaux et pas à d'autres. Par nature, les travaux sont en compétition pour l'utilisation des ressources, les travaux interfèrent pour leur utilisation et leurs objectifs ne sont pas nécessairement les mêmes. Ce genre de problème nécessite des études sur les travaux interférants.

Dans le cadre de l'étude théorique de cette thèse, nous nous focalisons sur des problèmes d'ordonnancement multicritères, dont l'étude a bien évolué dans les années récentes, et notamment sur deux approches très développées récemment. La première concerne l'approche multicrière de type 'Juste-à-temps' qui attire les chercheurs depuis les vingt dernières années par son intérêt pratique et sa difficulté. La seconde présente une type de problème d'ordonnancement multi-critère plus récent appelé "ordonnancement des travaux interférants".

Introduction à la partie I

Le critère lié à la minimisation simultanée des coûts d'avance et de retard fait l'objet de cette première partie de thèse. Ce critère est issu de la production juste-à-temps que nous présentons ci-dessous. La philosophie de la production JàT s'est développée depuis une soixantaine d'années comme une politique de production, qui permet de limiter les stocks et de fabriquer des produits qui correspondent exactement à la demande [32]. Sourd [199] indique que "*l'idée de base est d'éviter toute production anticipée de produits intermédiaires ou finaux, ce qui signifie que le producteur se contente de produire la quantité strictement nécessaire, au moment voulu, pour satisfaire la demande de chaque client. Ainsi, c'est la date à laquelle le client demande à être livré qui fixe la date de livraison du produit mais, le JàT va même plus loin : toutes les dates de toutes les opérations de la fabrication du produit livré sont également fixées de manière à ce qu'il n'y ait pas de temps mort dans la chaîne de fabrication. De cette manière, le produit est emballé juste-à-temps pour être livré, il est assemblé juste-à-temps pour être emballé et, ainsi de suite jusqu'à déterminer la date de production des matières premières*".

En effet, le concept du JàT a été préconçu au Japon au sein de l'usine Toyota par Taiichi Ohno et avait comme motivation principale l'élimination des gaspillages à tous les

niveaux [161]. L'introduction du système du Kanban a permis, au moyen d'étiquettes (Kanban signifie étiquette en japonais) de signaler les demandes de chaque poste de la chaîne de production, de mettre en oeuvre simplement une politique de production JàT : limitation de la production du poste amont aux besoins exacts du poste aval. Les articles de Huang et Kusiak [105] et Akturk et Erhun [8] offrent une synthèse sur la recherche récente dans ce domaine.

Dans cette partie, nous considérons des problèmes d'ordonnancement de travaux indépendants, sur une machine unique ou sur des machines parallèles. Nous nous intéressons plus particulièrement au cas où les travaux ont une date de fin souhaitée commune. Dans le chapitre 2, nous nous intéressons à la minimisation du retard pondéré total des travaux et proposons de nouveaux algorithmes de programmation dynamique. Dans le chapitre 3, nous tenons compte des pénalités d'avance, nous identifions de nouveaux cas polynomiaux dans le cas où les travaux ont des durées identiques et nous proposons de nouveaux schémas d'approximation dans le cas général. Dans le chapitre 4, nous considérons que la date de fin est commune aux travaux (ou à un sous-ensemble des travaux) et que cette date est à déterminer. De la même façon, nous identifions des cas polynomiaux et nous proposons quelques schémas d'approximation. Le chapitre 5 conclut cette partie.

Introduction à la partie II

La deuxième partie de cette thèse porte sur l'étude d'une nouvelle classe de problèmes d'ordonnancement multicritères. Dans la plupart des études en ordonnancement multicritère, tous les travaux contribuent de la même façon à la mesure de la qualité de la solution, et celle-ci peut être donnée par plusieurs critères. Par exemple, on considérera le maximum de toutes les dates de fin d'exécution des travaux comme un critère et le plus grand retard de tous les travaux comme un autre critère. Nous considérons dans cette partie qu'un critère est fourni par l'ensemble des travaux, mais que les autres critères peuvent porter sur un sous-ensemble des travaux. On voudra par exemple minimiser le maximum de toutes les dates de fin d'exécution des travaux, tout en respectant une borne pour le plus grand retard de certains travaux seulement. Ces problèmes se posent dans ces termes pour certaines entreprises où la production journalière est évaluée par un critère qui porte sur tous les travaux, et où des

sous-ensembles de travaux doivent respecter des contraintes – parfois imposées par les clients ou relatives aux retards déjà accumulés – selon d'autres critères.

Le chapitre 6 présente ces problèmes dans le contexte de la littérature et montre son originalité. Le chapitre 7 est consacré à l'étude des problèmes à une machine, des cas polynomiaux, NP-difficiles au sens ordinaire et NP-difficiles au sens fort sont identifiés. Le chapitre 8 porte sur l'étude des problèmes à machines parallèles. Le chapitre 9 conclut cette partie.

Une conclusion générale fait une synthèse des problèmes abordés, des résultats obtenus et des nombreuses perspectives de recherche qui découlent de ces études.

Première partie

Ordonnancement juste-à-temps à date due commune

Chapitre 1

Introduction à l'ordonnancement juste-à-temps

Dans ce chapitre, nous présentons d'une façon succincte le Juste-à-Temps (JàT) et ses objectifs ainsi que les différents niveaux de décisions qui apparaissent dans un système de production. Nous nous intéresserons par la suite aux problèmes d'ordonnancement JàT qui ont pour but de planifier l'exécution d'un travail dans un atelier de production. Après avoir présenté les différentes classes de problèmes rangés selon la fonction de coût ou selon la définition des dates de fin souhaitées des travaux, nous préciserons les problèmes d'ordonnancement JàT étudiés dans cette première partie de thèse. Quelques classes d'approximabilité et quelques définitions d'algorithmes à garantie de performance seront par la suite formellement exposées.

1.1 Systèmes "Juste-à-temps"

Les marchés sont de plus en plus exigeants sans pour autant renoncer à la diversité des produits ou autoriser la moindre augmentation des coûts. La pression économique de plus en plus forte et le niveau d'exigence croissant des clients obligent les entreprises

à réduire au maximum les stocks et les délais tout en essayant d'optimiser au mieux les coûts de production. La réduction des délais est un enjeu essentiel du processus de gestion d'une chaîne logistique. Il est donc nécessaire de se munir d'une véritable fonction capable de gérer la production en tenant compte des contraintes internes (sur l'exécution de l'opération actuelle et suivantes) et externes (comme les clients).

Dès les années 1920, lorsque la demande est devenue forte, l'industrie est passée de la production artisanale, à la production de masse. Tant que la question de la vente des produits ne se posait pas (par exemple les véhicules), le critère essentiel était la productivité. Puis, avec les crises des années 1970 il a fallu se tourner vers de nouveaux modes de production développés par les japonais (notamment par Taiichi Ohno dans les usines d'assemblage de voitures Toyota [161, 32]) pour leur marché national. Ainsi, on est passé de la production en flux poussés, à la production en flux tirés. Il s'agit donc d'un *système Juste-à-temps* (Jàt) qui peut être défini comme suit : "*C'est un système de gestion de la production en flux tendu visant la fabrication et le stockage des bonnes quantités au bon moment, à chaque étape du processus*". C'est la commande qui va déclencher la production et non plus la production qui déclenche la recherche d'un acheteur. Ainsi, la façon de gérer l'avancée des produits est renversée par rapport aux modèles pères tel que le modèle du lot économique (Harris) où la planification est basée sur une production annuelle régulière. Dans le modèle JàT, on parle de *flux tendus* pour une amélioration continue de la qualité et de la productivité et en essayant d'éliminer le "*Muda*" (*gaspillage* . Selon le président honoraire du C.A. Shoichiro Toyoda, Toyota "*le Gaspillage consiste en tout ce qui dépasse la quantité minimale requise en matériel, équipement, espace et temps pour ajouter de la valeur au produit*". Comme exemples de gaspillage, on peut citer :

- Surproduction et coûts de stockage
- Produits défectueux
- Méthodes inefficaces qui engendrent des attentes et des tâches inutiles
- Transports et manutentions inutiles

En effet, le gaspillage signifie non valeur ajoutée à détecter et éliminer pour accroître la productivité. La plus part des activités issues de *"Lean management"* sont orientées dans cette direction [158]. Comme le souligne Sensei Yoshida dans Lean Training Newsletter [225] "*être lean c'est arrêter de surproduire*". En effet, la surproduction est le pire du gaspillages car elle engendre un surstock, une surqualité et un surdimensionnement des équipements, par conséquent, une réduction des profits des entreprises. Le concept

lean est donc plus large qu'une méthode de production et s'appuie principalement sur quatre niveaux d'analyse : une redéfinition de la valeur produite par une entreprise (en particulier, prendre en considération le point de vue du client pour la définition de la valeur ajoutée d'un travail), le développement d'un schéma productif caractéristique (tirer la production en fonction de la demande et non pas la pousser en fonction des capacités locales de production), le développement d'attitudes managériales originales (tout acteur du *système de production* doit penser continuellement à une amélioration du système productif) et la formulation d'une stratégie à long terme (viser l'excellence en permanence dans un contexte de concurrence). La pensé lean dépasse alors le cadre de l'organisation de la production puisqu'elle s'étend aux services administratifs, au développement de produit et au développement informatique (pour plus de détails, voir [220]

Au coeur de la pensée lean se repose le concept JàT. Le modèle JàT cherche alors à fournir au client la commande prévue, le jour prévu, au coût minimum. Cette approche permet d'assurer la satisfaction du client tout en garantissant la rentabilité des ressources, donc de réduire les coûts et d'optimiser les services en travaillant au plus près de la demande client en réduisant les temps d'attente et les stocks. Ainsi, on peut résumer les objectifs du modèle JàT comme suit :

- Produire au rythme de la demande
- Lots de petite taille pour réduire le cycle de production ainsi renverser la façon de gérer l'avancée des produits, en passant des flux poussés aux flux tendus
- Éliminer le gaspillage
- Amélioration continue
- Respect des délais

Une telle approche peut être déployée aux niveaux stratégique, tactique et opérationnel. En effet, les décisions à prendre au niveau stratégique définissent le long terme. Quant à celles du niveau tactique, i.e. en moyen terme, elles représentent principalement la planification de la production. Les décisions opérationnelles ont pour but de respecter les décisions prises aux niveaux stratégique et tactique tout en assurant le bon déroulement de l'atelier de production en termes de gestion de moyens de transformation (les machines, par exemple) et de transport, les stocks afin de satisfaire à temps les commandes de clients. Ainsi, le système JàT a des conséquences directes sur le plan de production à mettre en place pour le très court terme au sein de l'atelier. Les

modèles d'ordonnancement classiques basés sur des critères réguliers classiques (C_{max}, T_{max}, L_{max}, etc.) ne semblent donc pas toujours satisfaisants par rapport à une politique de production en flux tendus . La qualité d'un ordonnancement peut être ainsi mesurée en fonction de l'*avance/retard* du produit par rapport à sa date de livraison, autrement dit de sa date de fin souhaitée : un coût de stockage est généré pour chaque produit réalisé avant sa date de fin souhaitée ; s'il s'agit en plus d'un produit périssable, il se peut qu'il perde complètement sa valeur. De même, un coût de pénalité est généré si le produit est livré en retard, sans parler des risques de pertes de clients dûs à leur mécontentement. Chercher un ordonnancement JàT, c'est calculer les dates de début et de fin des traitements des opérations sur la(les) machine(s) tout en minimisant le coût total de l'avance et retard des travaux.

Selon les fonctions de coûts d'avance/retard à minimiser et selon les définitions des dates de fins souhaités à respecter, on se retrouve face à des problèmes d'ordonnancement JàT différents que nous allons souligner dans les sections suivantes.

1.2 Mesurer l'avance/retard d'un travail

Mesurer l'avance ou le retard d'un travail (ou d'une quantité de produits à livrer) peut se faire de plusieurs façons.

L'avance est généralement définie en fonction des dates de fin souhaitées. Pour un travail J_i, on note E_i son avance avec $E_i = \max(0, d_i - C_i)$ avec C_i la date de fin d'exécution réelle de J_i (voir Figure 1.1). C'est la définition retenue dans notre étude. Il est à noter que dans certains cas, l'avance d'un travail est calculée en fonction de sa date d'arrivée (date de début au plus tôt), notée r_i ; ou bien, par rapport à sa date de début d'exécution réelle notée s_i. Dans le premier cas, on parle de la *promptitude* où l'avance est définie par $E_i^r = \max(0, r_i - s_i)$, illustré par la Figure 1.2 (cf. [193, 144, 97, 215]). Le second cas a été traité principalement dans [90, 62]. Les auteurs considèrent le cas d'atelier en ligne (flow shop) où tous les travaux doivent être exécutés sur toutes les machines dans le même ordre. Dans leur étude, le coût d'avance d'un travail J_i est fonction de la date de début de traitement de la première opération de J_i sur la première machine ($s_{(1,i)}$), donnée par $E_i^s = \max(0, d_i^s - s_{(1,i)})$ avec d_i^s une date de début de traitement souhaitée de la première opération de J_i (voir Figure 1.3).

Le retard d'un travail J_i, noté T_i, est toujours défini en fonction de la date de fin

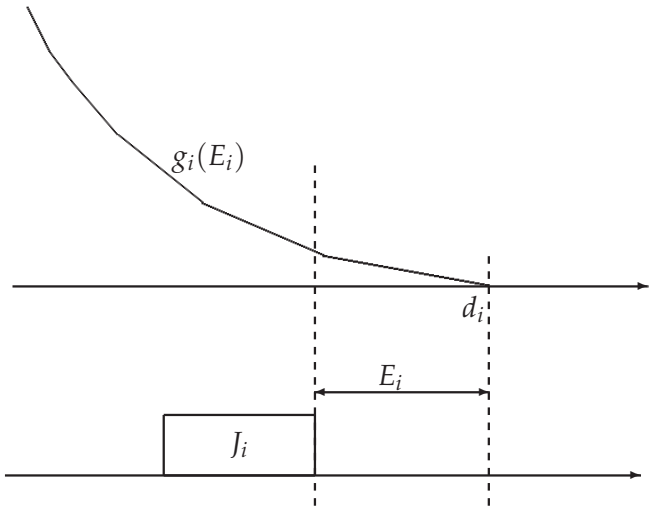

FIG. 1.1 – Calcul de l'avance en fonction de date de fin souhaitée

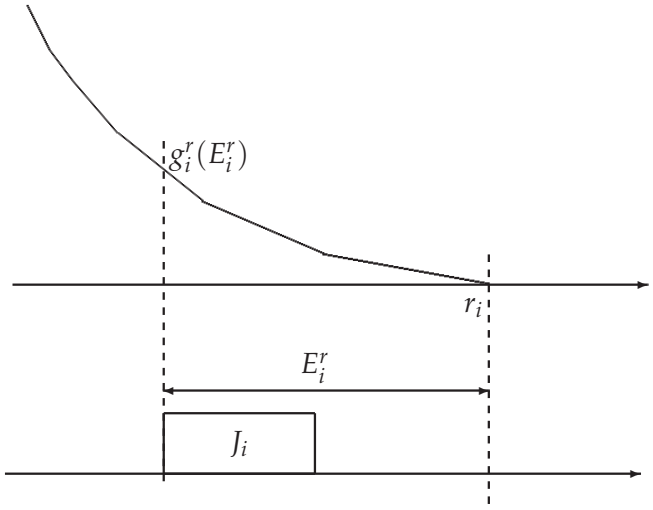

FIG. 1.2 – Calcul de l'avance en fonction de la promptitude

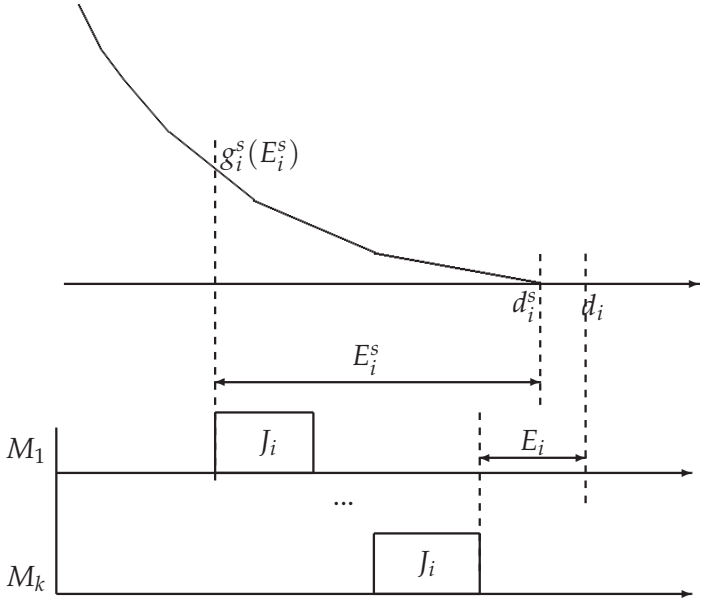

FIG. 1.3 – Calcul de l'avance en fonction de date de début souhaitée

souhaitée $T_i = \max(0, C_i - d_i)$ (voir Figure 1.4).

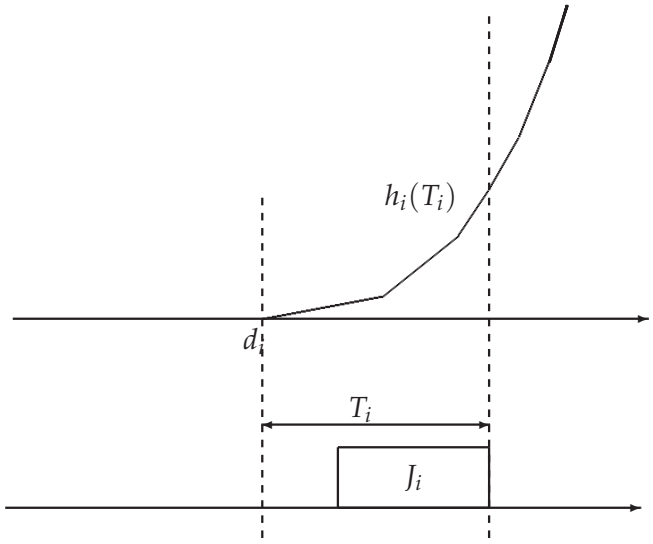

FIG. 1.4 – Calcul du retard

1.3 Mesurer le coût total d'avances et retards des travaux

Selon les motivations et les natures des problèmes, les chercheurs ont défini plusieurs métriques calculant le coût total engendré par l'avance/retard des travaux. En général, la fonction objectif se présente sous la forme d'une combinaison linéaire de

l'avance et retard pondérés. Elle est définie par : $f_i(C_i) = F_i(E_i, T_i) = \sum g_i(E_i) + h_i(T_i)$ où g_i et h_i peuvent être des fonctions en escalier, linéaires, linéaires par morceaux ou simplement convexes. Dans [200, 91], les auteurs considèrent le cas où chaque travail J_i a sa propre fonction de coût, notée $f_i(C_i)$ supposée convexe et linéaire par morceau. C'est une généralisation du cas classique où g_i et h_i sont deux fonctions linéaires. On peut citer d'autres travaux qui définissent la fonction objectif comme étant la somme des valeurs maximales de l'avance et du retard : $E_{\max} + T_{\max}$ et $\max_{1 \leq i \leq n} w_i(E_i + T_i)$ [156].

Dans cette thèse, on s'intéressera en particulier au cas où la fonction de coût d'avance/retard à minimiser est linéaire et continue (voir Figure 1.5).

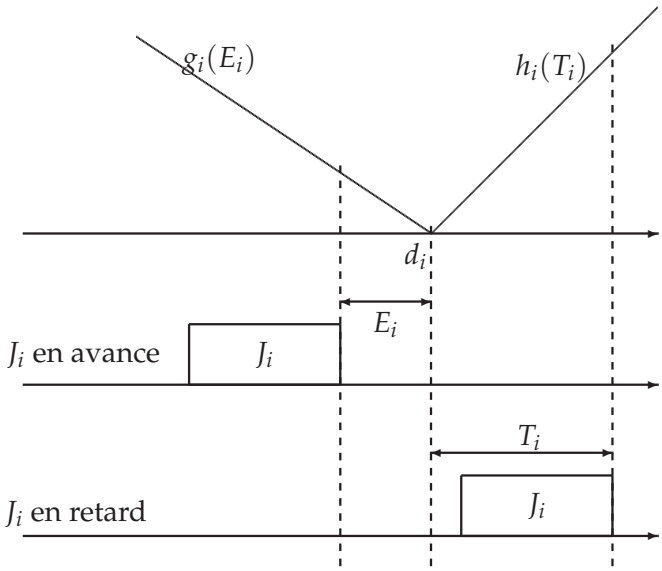

FIG. 1.5 – Fonction de coût d'avance/retard linéaire et continue

Dans, les paragraphes suivants, nous allons classer les différents problèmes d'ordonnancement JàT selon les valeurs des poids associés à l'avance/retard d'un travail.

1.3.1 Fonction de coût linéaire continue

On s'intéresse à minimiser la somme pondérée de l'avance/retard. Supposons que les fonctions de coûts g et h sont linéaires. Ainsi, le coût de l'avance et de retard total est donné par la fonction de coût $F_i(E_i, T_i) = \sum(\alpha_i E_i + \beta_i T_i)$ (voir Figure 1.5).

Selon les valeurs des poids α_i et β_i, quatre problèmes peuvent être identifiés.

1. **Cas unitaire** : si $\alpha_i = \beta_i = w$, $\forall i$; la fonction objectif est donnée par : $F_i(E_i, T_i) = \sum(E_i + T_i)$,

2. **Cas identique** : si $\alpha_i = \alpha$ et $\beta_i = \beta$, $\forall i$; la fonction objectif est donnée par : $F_i(E_i, T_i) = \sum(\alpha E_i + \beta T_i)$,
3. **Cas symétrique** : si $\alpha_i = \beta_i = w_i$, $\forall i$; la fonction objectif est donnée par : $F_i(E_i, T_i) = \sum(w_i(E_i + T_i))$
4. **Cas quelconque** : si α_i et β_i sont quelconques ; la fonction objectif est donnée par : $F_i(E_i, T_i) = \sum(\alpha_i E_i + \beta_i T_i)$

Contrairement aux fonctions objectif usuelles en ordonnancement, les critères considérés ne sont pas réguliers, c'est-à-dire qu'il arrive qu'on puisse améliorer une solution en introduisant un temps mort entre deux travaux consécutifs (voir Figure 1.6 avec deux travaux J_1 et J_2). Définir une séquence de travaux n'est donc pas suffisant pour déterminer une solution optimale, car nous avons également besoin des dates de début d'exécution de chaque travail. Selon la littérature, ce problème est connu sous le nom de "*optimal timing problem*" [41, 200, 92, 90, 189]. Dans le cas d'une seule machine, chercher les meilleures dates de début de traitement des travaux peut être résolu en temps polynomial. Pour les problèmes $\mathcal{NP}$-difficiles, les approches exactes classiques peuvent être utilisées, citons par exemple la programmation linéaire [70], la procédure par séparation et évaluation [59, 223, 134, 100, 157, 182, 64, 202]. Pour les méthodes approchées telles que les règles de séquencement, les algorithmes basés sur la recherche par faisceau filtré (beam search) [173, 62], les recherches par voisinage [228, 91, 201] sont aussi proposées. Pour la rechercher du placement optimal des travaux pour une séquence donnée, nous recommandons au lecteur intéressé les références suivantes [41, 200, 92, 24].

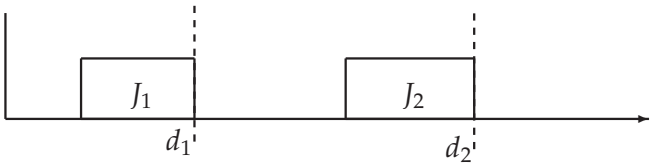

FIG. 1.6 – Juste-à-temps : critère non-régulièr

1.4 Dates de fin souhaitée

Comme les dates de fin souhaitées associées aux travaux sont de plusieurs types, les problèmes d'ordonnancement peuvent être classés en différentes catégories. En géné-

ral, les dates de fin souhaitées sont connues, mais lorsqu'il s'agit de la minimisation de l'avance/retard, ces dates de fin peuvent être variables et à déterminer (cf. par exemple [191, 169, 217]). Dans ce dernier cas, nous parlons de *dates de fin inconnues*. Selon la notation classique en trois champs, nous ajoutons dans le champ β les termes "d_i *unknown*". Dans le cas contraire, ces dates de fin sont des données du problème. Ainsi, si chaque travail a sa propre date de fin, nous parlons de *dates de fin souhaitée arbitraires*.

1.4.1 Date de fin commune donnée

Pour minimiser l'avance/retard, le plus classique est le modèle pour lequel tous les travaux ont une date de fin souhaitée commune. Un tel modèle correspond, par exemple, à un système d'assemblage où les composants d'un produit devraient être prêts en même temps, ou bien lorsqu'il s'agit d'une commande de client qui doit être livrée à une date prévue à l'avance. Plus généralement, ce modèle correspond à tout système où plusieurs travaux devraient être achevés en même temps. On parle d'un problème JàT avec date de fin souhaitée commune, et on note "$d_i = d$" – présenté par [129], la première référence introduisant le problème JàT avec date de fin souhaitée commune. Cependant, on distingue deux types de problèmes :

1. Date de fin commune *non-restrictive* (dite aussi *large*) : si les travaux peuvent être débutés avant la date zéro, i.e. augmenter la valeur d ne permet pas de réduire la valeur de la fonction objectif. Dans le champ β, on ajoute le terme *unrestrictive*.
2. Date de fin commune *restrictive* : dans ce cas on cherche à minimiser l'avance/retard par rapport à une date donnée fixée, i.e. les travaux débutent au plus tôt à la date zéro.

Pour le cas non-restrictive, une structure intéressante définissant une solution optimale a été élaborée dans [85, 61] : il existe une solution optimale où les travaux sont ordonnancés sans temps mort intermédiaire avec l'existence d'un travail k dit juste-à-temps, i.e. $C_k = d$. Généralement, la solution admet une structure "en V" (*V-shape*). Par exemple, le problème $1|d_i = d, unrestrictive|\sum(E_i + T_i)$ admet une solution optimale où les travaux en avance sont ordonnancés selon la règle LPT (*Longest Processing Time First*) et les travaux en retard respectent l'ordre SPT (*Shortest Processing Time First*) [129]. Un algorithme en $O(n^2)$ permet de déterminer l'ensemble des travaux en avance et ceux en retard. Cet algorithme peut être généralisé au cas de m machines parallèles

identiques pour le problème JàT identique (i.e. $\sum(\alpha E_i + \beta T_i)$) [101]. Dans le cas général, le problème a été montré $\mathcal{N}P$-difficile au sens ordinaire [87].

1.4.2 Famille de dates de fin commune donnée

Dans certaines situations, l'atelier de production doit honorer un certain nombre de commandes k. Ainsi, plusieurs produits faisant partie de la même commande, doivent être disponibles en même temps à une date connue fixée par le client ou par une société de transport. Dans ce cas de figure, on se retrouve face à un problème de JàT avec un ensemble de k dates de fin [55, 109, 111]. On note alors dans le champ β : "$d_i \in D$, $|D| = k$". Si k est connu on ajoute alors le terme, "k fixed".

1.4.3 Dates de fin commune contrôlable

Dans certaines situations, la date de livraison est une date à négocier auprès du client et donc elle n'est pas connue à l'avance, c'est donc une variable de décision. On parle d'un problème d'ordonnancement avec affectation de dates de fin. Dans le cas où tous les travaux doivent avoir la même date de fin, optimiser le cycle de production des travaux par exemple, le problème est dit *CON* ("*constant flow allowance*"). Dans la littérature de l'ordonnancement, nous pouvons recenser plusieurs manières dont les dates de fin souhaitées ont été définies. On peut donc donner une classification des problèmes d'ordonnancement avec dates de fins contrôlables comme suit.

1.4.3.1 Modèle SLK

Dans le modèle affectation des dates de fin de type *SLK* ("*SLacK*"), la date de fin souhaitée d'un travail J_i est une fonction de sa date d'arrivée r_i, sa durée opératoire p_i et d'un temps d'attente fixe et commun à tous les travaux noté q. Ainsi, nous avons $d_i = r_i + p_i + q$. Quand tous les travaux ont la même date d'arrivée, nous avons donc $d_i = p_i + q$. Le but est alors de minimiser l'avance/retard des travaux tout en minimisant le temps d'attente commun q.

1.4.3.2 Modèle TWK

Dans le modèle *TWK* ("*Total WorK*"), les dates de fin souhaitées sont des multiples des durées opératoires, données par $d_i = k \times p_i$ avec k un coefficient commun à tous

les travaux. On remarque que pour deux travaux de même durée opératoire, leur date de fin souhaitée est identique. Si les dates d'arrivées ne sont pas nulles, on aura alors $d_i = r_i + k \times p_i$. L'objectif est donc de minimiser l'avance/retard des travaux tout en déterminant la bonne valeur du coefficient k.

1.4.3.3 Modèle PPW

Le modèle *PPW* ("*Processing Plus Wait*") est un modèle combinant les deux règles calculées précédentes : SLK et TWK. Les dates de fin souhaitées des travaux disponibles sont déterminées par $d_i = k \times p_i + q$; sinon par $d_i = r_i + k \times p_i + q$ lorsque les dates d'arrivées ne sont pas identiques. L'affectation des dates de fin souhaitées optimales consiste à déterminer les valeurs optimales de k et q.

1.4.3.4 Modèle NOP

Dans le modèle NOP ("*Number of OPerations*"), les dates de fin souhaitées sont calculées par rapport au nombre d'opérations n_i à exécuter pour la réalisation du travail J_i : $d_i = r_i + k \times n_i$. L'objectif, dans ce cas, est de trouver la valeur optimale de k qui minimise la somme de l'avance/retard des travaux.

1.4.3.5 Autres modèles

D'autres modèles de calcul de dates de fin souhaitées contrôlables ont été considérés dans la littérature de l'ordonnancement. On peut citer par exemple le modèle RDM ("*RanDom Allowance*") où la date de fin souhaitée de chaque travail est générée aléatoirement soit selon une loi de distribution de probabilité ou bien par les méthodes dites de JIQ ("*Jobs In Queue*") et de JIS ("*Jobs In System*"). Selon les deux dernières méthodes, les dates de fin souhaitées sont calculées à partir de la longueur de la file d'attente et du nombre des travaux dans le système. Nous recommandons au lecteur intéressé les références suivantes [183, 213, 214, 73, 194, 151, 185, 176, 210], où l'approche de résolution utilisée est basée sur la simulation.

1.4.4 Dates de fin souhaitées généralisées

Dans certains systèmes de production, le transport des produits finis est négocié auprès des sociétés de transport. Le chef d'atelier dispose donc de plages horaires, autrement dit d'un ensemble de dates de fin souhaitées. Le but est de pouvoir réaliser

Problème	Signification	Date de fin souhaitée
CON	Constant flow allowance	$d_i = d$
SLK	Slack	$d_i = p_i + q$
TWK	Total work	$d_i = k \times p_i$
PPW	Processing plus wait	$d_i = k \times p_i + q$
NOP	Number of operations	$d_i = k \times n_i$
q and k sont les coefficients		
n_i est le nombre des opérations de travail J_i		

TAB. 1.1 – Classification des problèmes d'ordonnancement à affecter date de fin [126]

un ensemble de produits constituant une commande d'un client aux moments clés qui coïncident avec les départs des transporteurs. On parle alors de problèmes d'ordonnancement avec dates de fin souhaitées généralisées ([84, 88]) : l'ensemble des dates de fin souhaitées sont données et le problème consiste à affecter les travaux aux dates de fin.

Une autre variante du cas de date de fin généralisée est abordée dans [102]. Les auteurs parlent de dates de fin souhaitées positionnelles. Quand un travail J_i doit être exécuté pour la $j$ème date de fin souhaitée, on parle de dates de fin souhaitée positionnelles. Par exemple, si on donne trois dates de fin souhaitées d_1, d_2 et d_3 où $d_1 < d_2 < d_3$, pour trois travaux, l'objectif serait de réaliser au moins un travail pour la date d_1, au moins deux travaux pour la date d_2 et tous les autres travaux pour d_3. Cette méthode d'affectation de date de fin permet de savoir le nombre de travaux achevés à des moments clés définis pas les dates de fin souhaitées.

La plupart des problèmes d'ordonnancement sont montrés $\mathcal{NP}$-difficiles. Un point de vue pour résoudre efficacement ces problèmes est d'envisager des méthodes approchées. L'idée sous-jacente est qu'une bonne solution, éventuellement non-optimale mais obtenue en temps raisonnable vaut parfois mieux qu'une solution optimale nécessitant un temps de calcul très important. L'idée de gagner en temps de calcul (ou de rendre le calcul possible) au prix de la qualité de la solution peut alors séduire, voire s'imposer. Les méthodes approchées ne sont envisagées que pour des problèmes d'optimisation pour lesquels on considère qu'une solution satisfaisant les contraintes peut être exploitée sachant qu'elle le sera d'autant mieux si nous pouvons juger sa qualité indépendamment des instances du problèmes considérés, c-à-d. garantir la performance de la méthode proposée. Là encore, cette issue n'est envisageable que si le problème s'y

prête.

Dans la suite de ce chapitre, nous présentons quelques définitions et propriétés connues permettant de classer les problèmes d'ordonnancement du point de vue de leur complexité et des méthodes de résolution.

1.5 Complexité et approximation polynomiale

1.5.1 Approximation et garanties de performances

Pour présenter le principe de l'approximation polynomiale, nous nous baserons sur les travaux de Demange et Paschos [53] où les auteurs font un tour d'horizon de quelques méthodes. Les méthodes approchées diffèrent les une des autres par les compromis qualité/complexité qu'elles offrent. Nous en distinguons deux classes : les heuristiques et les algorithmes polynomiaux à garanties de performances .

Les premières, très utilisées en pratique, n'offrent aucune garantie a priori. Il s'agit de stratégies, de types très variés, dont le principe repose sur des arguments intuitifs, voire sur des paradigmes de phénomènes naturels mais qui ne sont justifiées que par les résultats qu'elles offrent. Aussi, le recours à des plans d'expérience est souvent retenu pour tester ces heuristiques. Dans certains cas, ces méthodes donnent lieu à des analyses complétant les observations ; elles sont systématiques (et même imposées dans les définitions) dans le cadre de l'approximation polynomiale. En effet ce domaine se définit par des règles strictes que doivent satisfaire les algorithmes étudiés. Le principe est de se limiter à l'emploi de méthodes polynomiales en exigeant de plus des garanties absolues sur la qualité des solutions obtenues. Le choix quant aux garanties de complexité et de qualité des solutions obtenues distingue l'approximation polynomiale des autres méthodes approchées. Il est particulièrement intéressant d'étudier le comportement des problèmes difficiles par rapport au compromis complexité/qualité de l'algorithme. L'*approximation polynomiale* restreint son champ d'étude à certains problèmes pouvant s'exprimer en terme d'optimisation. Pour chaque problème d'optimisation, on peut lui faire correspondre un problème de décision exprimés sous forme d'une question à réponse "oui" ou "non" ; la résolution du problème consiste alors à apporter une réponse à la question. À chaque instance, on associe une question portant sur l'existence ou non d'une solution meilleure qu'un seuil fixé. C'est par ce biais qu'on peut faire le lien entre les *problèmes d'optimisation* et les problèmes de décision de la classe

$\mathcal{NP}$. Ainsi, l'ensemble des problèmes d'optimisation $\mathcal{NP}$ est noté $\mathcal{NPO}$. Un problème de décision ne peut pas être plus dur que le problème d'optimisation $\mathcal{NPO}$ correspondant. Lorsque sa version décision est $\mathcal{NP}$-complet , le problème $\mathcal{NPO}$, au moins aussi difficile qu'un problème $\mathcal{NP}$-complet, est dit $\mathcal{NP}$-difficile . Le principal intérêt d'exprimer un problème en termes d'optimisation est d'avoir à sa disposition, non seulement une notion de solution acceptable (ou réalisable) du problème, c'est-à-dire qu'elle exprime par le biais de contraintes ce qu'on désire imposer à n'importe quelle solution envisageable, mais aussi une solution de qualité d'une solution réalisable par référence à la valeur de l'objectif. Selon Paschos [170], *"l'approximation polynomiale, c'est l'art d'obtenir des solutions réalisables, de valeur proche (dans un sens prédéfini) de la valeur optimale et tout cela en temps polynomial"* [170].

En nous plaçant dans le cadre des problèmes difficiles à résoudre, face aux grandes instances l'éventualité de détériorer la valeur optimale pour accélérer le temps de calcul semble être un choix absolument justifié. Cependant, il serait intéressant de pouvoir associer à chaque heuristique une tolérance quant à l'erreur qu'elles peuvent impliquer. C'est ainsi que fut introduite la notion d'erreur relative, c'est-à-dire l'écart maximal par rapport à la valeur optimale de la fonction objectif qu'un algorithme peut produire, et cela quelle que soit l'instance du problème étudié. Plusieurs notions d'approximations ont été présentées dans la littérature (voir [71, 172, 130, 13, 212, 171]), mais nous nous en tiendrons à celle qui est la plus répandue. C'est ainsi que nous obtenons la définition dans le cadre général de l'optimisation :

Définition 1.5.1 *Soit* Π *un problème de* $\mathcal{NPO}$ *et un algorithme* A *pour résoudre ce problème. Étant donnée une instance* I *du problème et une solution réalisable* $f(A(I))$ *de* I*, on définit :*

1. *"l'erreur relative de* A *respectant* I*", le rapport*

$$Err(I, A) = \frac{|f^*(I) - f(A(I))|}{\max\{f^*(I), f(A(I))\}}$$

2. *"le rapport de performance de* A *respectant* I*", le rapport*

$$R(I, A) = \min\{\frac{f(A(I))}{f^*(I)}, \frac{f^*(I)}{f(A(I))}\}$$

avec $f^*(I)$ *la valeur de la solution optimale d'une instance du problème* Π*.*

En nous plaçant dans le cadre des problèmes de minimisation pour les problèmes

d'ordonnancement avec $f^*(I)$ la valeur optimale du problème, $f(A(I))$ la valeur retournée par l'algorithme A et ε l'erreur relative, on obtient les équations

$$f(A(I)) - f^*(I) = \varepsilon f^*(I)$$

soit

$$\frac{f(A(I))}{f^*(I)} = 1 + \varepsilon$$

À partir de ces équations nous pouvons désormais définir la garantie de performance d'une heuristique, c'est-à-dire la valeur de l'écart relatif que l'algorithme A garantit quelle que soit l'instance étudiée.

Définition 1.5.2 *Soit* Π *un problème de minimisation de* $\mathcal{NPO}$*. Quelle que soit l'instance* I *du problème, un algorithme polynomial à garantie de performance* A *qui retourne une solution réalisable de* I*, vérifie l'inéquation :*

$$f(A(I)) - f^*(I) \leq \varepsilon f^*(I)$$

soit

$$\frac{f(A(I))}{f^*(I)} \leq 1 + \varepsilon$$

avec ε *fixé,* $f(A(I))$ *le coût de la solution retournée par* A *et* $f^*(I)$ *le coût de la solution optimale.*

1.5.2 Schémas d'approximation polynomiaux

Dans le cadre de cette thèse, nous nous intéressons particulièrement à l'élaboration de schémas d'approximation pour les problèmes d'ordonnancement étudiés. Précisons tout d'abord ce qu'est un schéma d'approximation. Cette présentation sera basée en grande partie sur les travaux de Woeginger [218], Demange et Paschos [53]. Nous choisissons de présenter la classe des schémas d'approximation de façon formelle. Nous nous plaçons dans la classe des problèmes d'optimisation $\mathcal{NP}$, soit la classe $\mathcal{NPO}$. Afin de présenter cette classe de façon concise, nous allons donner quelques définitions rencontrées dans la littérature [53, 71].

- $\mathcal{NPO}$

Définition 1.5.3 *Soit* Π *un problème* $\mathcal{NPO}$ *et* A *un algorithme qui, pour toute instance* I *de* Π*, retourne une solution réalisable* $A(I)$*. Étant donné un rationnel arbitraire* ε*, on dit que* A *est un algorithme d'*ε*-approximation pour* Π *si, pour chaque instance* I*, la solution réalisable* $A(I)$ *vérifie l'inégalité* $\frac{f(A(I))}{f^*(I)} \leq 1 + \varepsilon$*.*

• $\mathcal{APX}$

Définition 1.5.4 *Un problème* Π *de* $\mathcal{NPO}$ *appartient à la classe* $\mathcal{APX}$ *si un algorithme d'*ε*-approximation à temps polynomial* A *pour* Π *existe.*

Pour synthétiser les définitions précédentes, nous ajoutons qu'un algorithme d'ε-approximation peut être assimilé à une heuristique avec une garantie de performance de $1 + \varepsilon$.

Un problème appartient à la classe $\mathcal{APX}$ si on sait qu'il existe un ε compris entre 0 et 1 satisfaisant cette condition.

Définition 1.5.5 *Soit* Π *un problème* $\mathcal{NPO}$*, un algorithme* A *est un schéma d'approximation pour* Π *si, pour chaque instance* I *de* Π *et pour une famille de rationnel* ε*,* $A(x, \varepsilon)$ *renvoie une solution réalisable dont l'erreur relative est au plus* ε*.*

Un schéma d'approximation pour un problème est donc un algorithme prenant comme paramètre ε. Il est ainsi possible de préciser la garantie de performance de l'algorithme en question.

• $\mathcal{PTAS}$

Définition 1.5.6 *Un problème* Π *de* $\mathcal{NPO}$ *appartient à la classe* ***PTAS*** *s'il admet un schéma d'approximation à temps polynomial, c'est-à-dire un schéma d'approximation dont la complexité est bornée par* $q_\varepsilon(|I|)$*, où* q_ε *est un polynôme.*

Un **PTAS** (*Polynomial Time Approximation Scheme*) est un schéma d'approximation, donc un algorithme, dont la complexité est polynomiale par rapport à la taille de l'instance, même si le polynôme majorant la complexité est en fonction de ε.

• $\mathcal{FPTAS}$

Définition 1.5.7 *Un problème* Π *de* $\mathcal{NPO}$ *appartient à la classe* ***FPTAS*** *s'il admet un schéma d'approximation à temps complètement polynomial, c'est-à-dire un algorithme dont la complexité est bornée par* $q(|I|, 1/\varepsilon)$*, où* q *est un polynôme.*

Un **FPTAS** (*Fully Polynomial Time Approximation Scheme*) est un schéma d'approximation à temps complètement polynomial c'est-à-dire un algorithme dont la complexité est polynomiale en fonction de la taille de l'instance et de l'inverse du paramètre ε. En général, il est possible de déduire un FPTAS à partir d'un algorithme pseudo-polynomial, tel qu'un algorithme de programmation dynamique, quelle que soit l'erreur relative ε choisie.

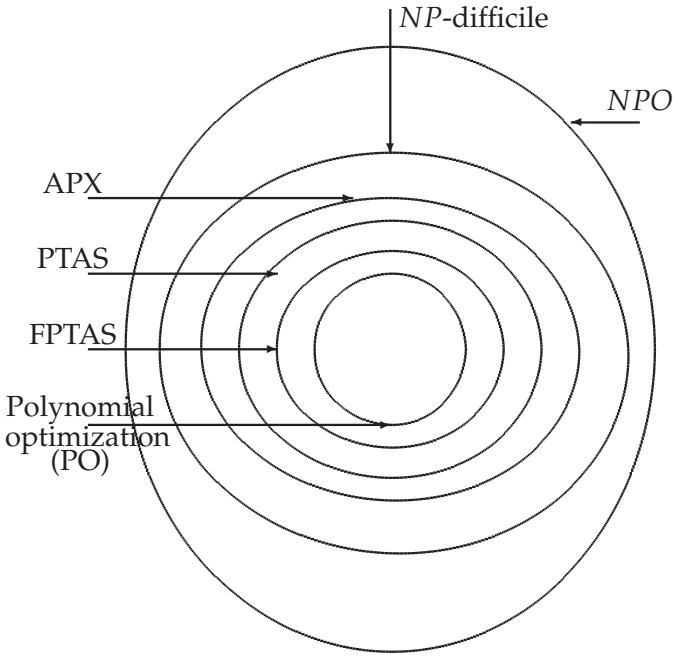

FIG. 1.7 – Classes d'approximabilité (sous l'hypothèse $\mathcal{P} \neq \mathcal{NP}$) [171]

En effet, il est à noter que les problèmes $\mathcal{NP}$-difficiles au sens fort n'admettent pas de schéma d'approximation FPTAS et que certains problèmes $\mathcal{NP}$-difficiles au sens ordinaire n'en admettent pas non plus. Les différentes classes de problèmes présentées s'encapsulent tel que montré sur la figure 1.7. En 1978, Garey et Johnson [71] avaient identifié certaines conditions sur les problèmes d'optimisation qui garantissent que le problème ne possède pas de schéma d'approximation, sous l'hypothèse $\mathcal{P} \neq \mathcal{NP}$. On peut également faire référence à de nombreux autres travaux [172, 15, 14]. Le principe des schémas d'approximation est apparu au milieu des années 70 avec les travaux de Horowitz, Ibarra, Kim, et Sahni [104, 124, 190] et ils sont, pour la plupart, basés sur des formulations de programmation dynamique qui sont, rappelons le, des méthodes

exactes de résolution. On dénote deux grandes familles d'approches pour transformer une formulation de programmation dynamique en schéma d'approximation. La première consiste à arrondir les données d'entrée d'une instance. Le but est de simplifier les données de l'instance (les entrées), ainsi de rendre le problème obtenu facile à résoudre. Nous appelons cette méthode la technique des "entrées arrondies" (*rounding-the-input-data*) [190]. La deuxième méthode consiste à simplifier l'espace de recherche de la programmation dynamique en réduisant l'espace de recherche sur lequel la méthode pseudo-polynomiale doit être appelée (*trimming-the-state-space*) [124].

1.6 Problèmes d'ordonnancement Juste-à-temps abordés

Pour conclure de chapitre introductif, nous avons souligné que l'approche de type Juste-à-Temps peut être déployée aux différents niveaux décisionnels d'un système de production. Dans cette première partie de thèse, nous nous sommes intéressés au niveau opérationnel où nous essayons d'apporter quelques améliorations à l'ordonnancement à court terme dans un atelier de production. Nous nous sommes particulièrement intéressés aux problèmes d'ordonnancement présentés dans le tableau 1.2, où notre but est de proposer des méthodes exactes ou approchées avec garantie de performances pour déterminer une meilleure politique d'ordonnancement des travaux à réaliser.

	Critère	$d_i = d$			$d_i \in D$, $\|D\| = k$, k fixé
		d donnée		d inconnue	
		non-restrictive	restrictive		
Chapitre 2	$\sum w_i T_i$	-	1 ;Pm ;Qm	-	Pm ;Qm
Chapitre 3	$\sum(\alpha_i E_i + \beta_i T_i)$	-	1 ;Pm ;Qm	-	Pm ;Qm
Chapitre 4	$\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$	-	1 ;Pm ;Qm	-	1,Pm
	1 - Une seule machine Pm - m machines parallèles identiques avec m fixé Qm - m machines parallèles uniformes avec m fixé				

TAB. 1.2 – Tableau de problèmes abordés

Chapitre 2

Retard pondéré avec une date de fin donnée

Dans ce chapitre, nous nous intéressons à l'ordonnancement des travaux avec des dates de fin souhaitées communes sur machines parallèles. L'objectif est de réduire au maximum la somme des retards pondérés. Ce problème est connu pour être $\mathcal{NP}$-difficile, Lawler et Moore [150] proposent une méthode basée sur la programmation dynamique (ProgDyn) pour résoudre le cas d'une seule machine avec une date de fin souhaitée commune. D'après les auteurs, cet algorithme de ProgDyn peut être généralisé au cas de plusieurs machines. Nous montrons que cette généralisation n'est pas si immédiate. Un nouvel algorithme de ProgDyn est proposé dans ce chapitre pour résoudre le cas de m machines parallèles identiques et uniformes. Nous montrons comment cet algorithme peut être généralisé au cas de familles de dates de fin souhaitées. Une discussion sur les schémas d'approximation est abordée à la fin de ce chapitre.

2.1 Introduction

Les problèmes d'ordonnancement avec comme objectif la minimisation des retards pondérés ont attiré l'attention des chercheurs depuis plus d'une quarantaine d'années.

D'un point de vue pratique, cet objectif vise à réduire les risques de coûts supplémentaires causés par les retards de livraison, le non respect du carnet de commandes et des contrats, ou tout simplement le mécontentement ou la perte de clients. Un autre exemple pratique du problème considéré ici est présenté dans [56] où une famille de produits doit se présenter devant l'atelier de séchage à des instants déterminés après la procédure de peinture et de vernissage. Ici le client n'est autre que le chef de l'atelier de séchage. Comme certains produits sont plus prioritaires que d'autres, on cherche donc à minimiser les coûts pondérés des produits arrivant en retard à l'entrée de l'atelier de séchage.

La minimisation de la somme pondérée des retards est un cas particulier des problèmes d'ordonnancement dits Juste-à-Temps (JàT) [20, 77, 126]. Si le coût de stockage et/ou de la détérioration des produits sont négligés, alors la production en avance par rapport à une date de fin souhaitée n'est pas pénalisée.

Dans certains cas, négliger les pénalités d'avance peut être justifié, comme le souligne Rossier [186] dans le cas d'ateliers de sidérurgie. Les trois raisons qui ont poussé l'auteur à considérer le coût d'avance nul pour la production de barres métalliques sont : "premièrement, au niveau opérationnel, l'ensemble des commandes envisagé pour une planification est limité par un horizon restreint et, lorsque les délais sont tenus, il est préférable de minimiser les temps de réglage plutôt que les durées de stockage. Deuxièmement, il est coûteux en pratique de stopper la production sur un groupe de coulage (opérations de mise hors service) afin que des barres soient produites juste à temps. Troisièmement, étant donné les taux de renouvellement et les volumes des stocks d'en-cours du niveau opérationnel, il est raisonnable de supposer que les coûts occasionnés par l'intérêt des capitaux immobilisés sont négligeables par rapport aux autres impératifs de la production". Néanmoins, s'il s'agissait des produits périssables, les coûts d'avance ne peuvent pas être ignorés. L'ordonnancement avec coût d'avance et retard sera abordé dans les chapitres suivants.

Le problème considéré dans ce chapitre est défini comme suit : n travaux sont à exécuter sans pré-emption sur m machines parallèles identiques ($m \geq 1$). Aucune pénalité n'intervient si un travail J_i est achevé avant la date de fin souhaitée ($C_i \leq d_i$). Sinon, une pénalité notée w_i ($i = 1, .., n$) intervient pour chaque unité de retard. Comme défini dans [133], un travail peut être dans un des trois états suivants : travail en avance si $C_i < d$, travail totalement en retard si $S_i > d$, ou "straddling job" si $S_i < d \leq C_i$ où S_i est

la date de début d'exécution de J_i. Ainsi, le coût de retard total d'un travail J_i est donné par $w_i \times (C_i - d_i) = w_i \times T_i$. L'objectif est donc la minimisation de la somme pondérée des retards, notée $\sum_{i=1}^{n} w_i T_i$. Selon la notation en trois champs, le problème étudié est noté $Pm|d_i \in D, |D| = l|\sum w_i T_i$. Ce problème est $\mathcal{NP}$-difficile car le cas d'une seule machine $1|d_i = d|\sum w_i T_i$ est $\mathcal{NP}$-difficile [226].

Dans la section suivante, nous présentons quelques résultats connus de la littérature.

2.2 État de l'art

Selon [133], résoudre le $1|d_i|\sum w_i T_i$ et ses extensions n'a pas que des intérêts pratiques ([1] et [192]). Ce problème joue traditionnellement le rôle d'un "terrain d'essai" pour tester les techniques de la recherche opérationnelle destinées à la résolution des problèmes d'ordonnancement telles que celles citées dans [43, 178]. Par exemple, plusieurs programmes dynamiques et algorithmes d'approximation ont été proposés pour certaines extensions du problème $1|d_i|\sum w_i T_i$.

En abordant le cas simple avec une seule date de fin souhaitée, Lawler et Moore [150] développent un algorithme pseudo-polynomial en $O(n^2 d)$ basé sur la programmation dynamique. Une vingtaine d'années plus tard, Yuan [226] montre que le $1|d_i = d|\sum w_i T_i$ est $\mathcal{NP}$-difficile.

Le cas général $1|d_i|\sum w_i T_i$ est montré $\mathcal{NP}$-difficile au sens fort [146, 154].

Le problème général $1||\sum w_i T_i$ et ses variantes a beaucoup attiré l'attention des chercheurs, des algorithmes énumératifs qui utilisent la programmation dynamique et une procédure par séparation et évaluation ont été proposés par Kan et al. en 1975 [179], Fisher en 1976 [68], Lawler en 1979 [147], Potts et Wassenhove en 1985 [178]. Ces algorithmes et d'autres méthodes approchées sont discutés et évalués par Abdul-Razaq et al. en 1990 [1].

Selon [1], les méthodes exactes permettent de résoudre efficacement des problèmes de petite taille (le nombre de travaux ne dépasse pas 50). Les algorithmes exacts de type énumératif exigent des ressources informatiques considérables tant en terme de temps de calcul qu'en espace de stockage. Par conséquent, plusieurs heuristiques ont été proposées, basées sur des règles d'échange (2-opt, par exemple), des méthodes de voisinages (descente locale [43], recuit simulé [159], recherche tabou [44]) ou encore des

méthodes dites de construction progressive d'une solution (à chaque itération un travail est fixé à une position). Plusieurs heuristiques constructives ont été développées par Fisher [68], Morton et al [163], Cheng et al [39] ou encore Potts et Van Wassenhove [180]). Elles sont très rapides, mais la qualité de la solution n'est pas totalement satisfaisante par rapport aux résultats obtenus principalement par des méthodes de voisinages ou basées sur les règles d'échange. Cette dernière classe d'heuristiques peut être décrite comme suit : à partir d'une solution initiale, on essaye à chaque itération d'améliorer la solution actuelle par des changements locaux. La procédure d'échange est appliquée jusqu'à ce qu'aucune amélioration ne soit possible. Ce principe est utilisé dans la recherche tabou (Crauwels et al. [44], Bozejko et al. [28], Bilge et al. [25]), recuit simulé (Matsuo et al. [159], Potts et Van Wassenhove [180]), où dans les opérateurs de croisement et de mutation des algorithmes évolutionnaires [44], l'optimisation à colonie de fourmis artificielles (DenBasten et al. [23]. Soulignons tout de même l'efficacité de la descente locale proposée par Congram et al. en 2002 [43], reprise par la suite en 2004 par Grosso et al. [81] et récemment dans les travaux de Ange et Bampis en 2005 [11]. Dans [11], le temps de traitement de chaque travail dépend de sa date de début d'exécution. L'efficacité de la méthode proposée dans [11] repose sur sa capacité d'explorer un voisinage de taille exponentielle dans un temps polynomial en se basant sur le principe de la programmation dynamique.

Si les poids w_i sont identiques, le $1|d_i|\sum T_i$ est $\mathcal{NP}$-difficile au sens ordinaire [60] et peut être résolu en $O(n^4P)$ où P correspond à la somme des durées opératoires [146]. Dans le cas de poids agréables, i.e. $1|p_j < p_i \Rightarrow w_j \geq w_i, d_i|\sum w_i T_i$, le problème est résolu en temps pseudo-polynomial [146]. Dans le cas de durées opératoires identiques, il a été montré que chercher une solution optimale revient à résoudre un problème d'affectation, ce qui peut être réalisé en $O(n^3)$ [154].

Il arrive dans de nombreuses situations pratiques que la date de fin souhaitée d'un travail et son poids soient proportionnels à sa durée opératoire ([146], [12] et [39]). Pour le modèle '*equal-slack due date*' (c-à-d. $d_i = p_i + q$, voir 1.1) avec des poids unitaires, le problème est résolu en temps polynomial [146]. Dans le cas de poids quelconques, le problème est montré $\mathcal{NP}$-difficile au sens ordinaire [39]. Lorsque les coûts de pénalités sont fonction linéaire des durées opératoires ($w_i = k \times p_i$), Arkin et Roundy [12] montrent que ce problème $1|w_i = kp_i|\sum w_i T_i$ est $\mathcal{NP}$-difficile au sens ordinaire et un algorithme pseudo-polynomial est proposé. Plus généralement, il s'agit du modèle

'processing-plus-wait due date' (c-à-d. $d_i = \alpha p_i + q$ et $w_i = kp_i$, c.f 1.1) [39]. La complexité du problème varie selon la valeur de α ($\alpha \in \mathcal{R}$). En effet, si $\alpha \leq 0$, il existe une solution optimale où la règle LPT est respectée [39] ; si $\alpha \geq 1$, une solution optimale est définie par l'ordre SPT [39] ; sinon ($0 < \alpha < 1$) le problème devient $\mathcal{NP}$-difficile [39]. Pour ce dernier cas, Arkin et Roundy proposent un algorithme pseudo-polynomial pour déterminer une solution optimale [12].

Le cas de famille de dates de fin souhaitées donnée est étudié par Kolliopoulos et Steiner [135]. Les auteurs s'intéressent donc au problème $1|d_i \in D, |D| = l|\sum w_i T_i$ et proposent de le résoudre par la programmation dynamique.

Problème	Méthode & Complexité	Référence
$1\|p_i = p\|\sum w_i T_i$	Polynomial	[154]
$Pm\|p_i = p, r_i\|\sum T_i$	Polynomial	[21]
$P\|p_i = p, r_i\|\sum T_i$	Polynomial - programmation linéraire	
$P\|p_i = 1, r_i\|\sum w_i T_i$	Polynomial - problème de transportation $O(mn^3)$	
$Q\|p_i = p\|\sum w_i T_i$	Polynomial - problème d'affectation $O(n^3)$	[154]
$1\|w_i = kp_i\|\sum w_i T_i$	$\mathcal{NP}$-difficile au sens ordinaire	[12]
	Programme dynamique $O(n^2 P)$	[12]
$1\|w_i = kp_i, \alpha \in (0,1), d_i = \alpha p_i + q\|\sum w_i T_i$	$\mathcal{NP}$-difficile au sens ordinaire	[39]
	Programme dynamique $O(n^2 P)$	[12]
$1\|w_i = kp_i, \alpha \leq 0, d_i = \alpha p_i + q\|\sum w_i T_i$	Polynomial (règle LPT)	[39]
$1\|w_i = kp_i, \alpha \geq 1, d_i = \alpha p_i + q\|\sum w_i T_i$	Polynomial (règle SPT)	[39]
$1\|d_i = d\|\sum w_i T_i$	$\mathcal{NP}$-difficile au sens ordinaire	[226]
	Programme dynamique $O(n^2)d$	[150]
$1\|d_i = p_i + q\|\sum T_i$	Polynomial	[146]
$1\|d_i = p_i + q\|\sum w_i T_i$	$\mathcal{NP}$-difficile au sens ordinaire	[39]
	Programme dynamique $O(n^2 P)$	[39]
$1\|\|\sum T_i$	$\mathcal{NP}$-difficile au sens ordinaire	[60]
	Programme dynamique $O(n^7 P..)$	[146]
$1\|p_j \geq p_k \Rightarrow w_j \leq w_k\|\sum w_i T_i$	Programme dynamique $O(n^3 P)$	[146]
$1\|d_i \in D, \|D\| = l\|\sum w_i T_i$	FPTAS	[135]
$1\|\|\sum w_i T_i$	$\mathcal{NP}$-difficile au sens fort	[146], [154]
	$(n-1)$-approximation algorithme en $O(n^2)$	[39]
	PSE $(n = 50)$	[179], [68], [180], [1]

TAB. 2.1 – État de l'art sur la minimisation des retards

2.2.1 Approximabilité du retard pondéré

Étant donné que le problème en général est $\mathcal{NP}$-diffile, certains auteurs se sont intéressés à l'étude et à l'élaboration de schémas d'approximation polynomiaux. On peut citer par exemple les travaux présentés dans [148, 138, 133, 136, 137]. Le problème $1|d_i|\sum T_i$ accepte un schéma d'approximation complètement polynomial (FPTAS - *fully polynomial time approximation scheme*) [148, 138, 137].

Pour le cas général $1|d_i|\sum w_i T_i$, Cheng et al. [39] montrent qu'il existe un algorithme de $(n-1)$-approximation. Avec un nombre fixe de dates de fin souhaitées donné (famille de dates de fin), Kolliopoulos et Steiner [135] expliquent que si le poids des travaux est borné par une fonction polynomiale en n, le problème accepte cependant un FPTAS.

Kolliopoulos et Steiner [136] déterminent un FPTAS en $O(n^3/\varepsilon)$ pour le problème $1|d_i = d|\sum w_i(T_i + d)$, où $\sum w_i d$ est une constante supplémentaire. On peut montrer que la séquence optimale pour $1|d_i = d|\sum w_i(T_i + d)$ est également optimale pour le cas $1|d_i = d|\sum w_i T_i$, mais un algorithme d'approximation pour le premier ne peut pas garantir la même approximation pour le deuxième. Néanmoins, pour le problème $1|d_i = d|\sum w_i T_i$, Kellerer et Strusewich [133] déterminent un FPTAS en $O(n^6 \log W/\varepsilon^3)$ où $W = \sum_{i=1}^{n} w_i$. Cependant, ce schéma est considéré comme un FPTAS du fait que la complexité est en fonction de $\log W$.

L'étudie des schémas d'approximation pour le cas unitaire avec dates de fin quelconques ($1|d_i|\sum T_i$) est présenté dans [148]. Lawler propose un FPTAS de complexité $O(n^7/\varepsilon)$, pour tout $\varepsilon > 0$. Par la suite, un autre schéma d'approximation améliorant la complexité de ce dernier est proposé par Kovalyov [138] et ensuite par Koulamas [137]. Ainsi, la meilleure complexité d'un FPTAS connu est en $O(n^5 \log n + n^5/\varepsilon)$ [137].

Pour le problème '*equal-slack due date*', Cheng et al. [39] définissent un FPTAS pour un temps de calcul borné par $O(n^4/\varepsilon)$.

Contrairement au cas $1|d_i|\sum w_i T_i$ où il existe une littérature abondante traitant ses différentes variantes, très peu de résultats sont dédiés au problème à m machines parallèles. En effet, pour déterminer un schéma d'approximation pour le problème $Pm|d_i = d|\sum T_i$, Kovalyov et Werner [141] montrent qu'il n'existe aucun algorithme d'approximation polynomial sauf si $\mathcal{P} = \mathcal{NP}$. Cependant, ils proposent deux algorithmes d'approximation avec garantie de performance et montrent que la valeur de la solution obtenue donnée par un des deux algorithmes d'approximation, notée par $X^\sharp$, satisfait

l'inégalité suivante $(X^\sharp - X^*) \leq \varepsilon(X^* + d)$ (X^* correspond à la valeur optimale). Autrement dit, les auteurs définissent un FPTAS pour le problème $Pm|d_i = d|\sum T_i + d$.

Problème	Méthode & Complexité	Référence
$1\|d_i = d\|\sum w_i T_i$ 2-approximation algorithme en $O(n^2)$	[65]	
	FPTAS $O(n^6/\varepsilon^3)$	[133]
$Pm\|d_i = d\|\sum w_i T_i$	$\mathcal{NP} \setminus \mathcal{APX}$	[141]
	Heuristique avec $\frac{X^\sharp - X^*}{X^* + d} \leq \varepsilon$	[141]
$1\|d_i = p_i + q\|\sum w_i T_i$	FPTAS $O(n^4/\varepsilon)$	[39]
$1\|\|\sum T_i$	FPTAS $O(n^7/\varepsilon)$	[146]
	FPTAS $O(n^6 logn + n^6/\varepsilon)$	[138]
	FPTAS $O(n^6 logn + n^6/\varepsilon)$	[137]
$1\|d_i \in D, \|D\| = l\|\sum w_i T_i$	FPTAS	[135]
$1\|\|\sum w_i T_i$	$(n-1)$-approximation algorithme en $O(n^2)$	[39]

TAB. 2.2 – État de l'art sur la minimisation des retards

2.2.2 Remarques sur programme dynamique de Lawler et Moore

En 1969, Lawler et Moore [150] proposent un programme dynamique général où la fonction récursive peut être adaptée pour résoudre une grande variété de problèmes d'ordonnancement à une seule machine, machines parallèles (identiques, uniformes, non-reliées) et flowshop à deux machines. Les auteurs considèrent plusieurs fonctions objectifs, par exemple $\sum w_i T_i$, $\sum U_i$. Ils proposent une formulation générique d'un programme dynamique pour la résolution les problèmes d'ordonnancement considérés. La fonction de récurrence est ainsi définie par Lawler et Moore (voir [150] section 1) comme suit :

Soit $f(i,t)$ correspondant au coût de retard pondéré total des i premiers travaux ordonnancés dans l'intervalle $[0,t]$ et $[\sum_{j=i+1..n} p_j + t, \sum_{j=1..n} p_j]$ (t est la date d'achèvement du travail J_i). Ainsi nous avons :

$$
\begin{aligned}
f(0,t) &= 0, && (\forall t \geq 0)\\
f(i,t) &= +\infty, && (\forall i \in \{0,1,\ldots,n\}; \forall t < 0)\\
f(i,t) &= \min \left\{ \begin{array}{l} f(i,t-1), \\ \alpha_i(t) + f(i-1, t-a_i), \\ \beta_i(t) + f(i-1, t-b_i) \end{array} \right\}, && (\forall i \in \{1,\ldots,n\}; \forall t \geq 0)
\end{aligned}
$$

Où a_i et b_i sont des durées opératoires de travail J_i. Selon la date d'achèvement du travail J_i, les auteurs considèrent deux coûts différents : $\alpha_i(t)$ et $\beta_i(t)$. Par exemple, si l'objectif est la minimisation de la somme des retards, alors le coût d'avance $\alpha_i(t) = 0$; cependant le coût de retard est donné par $\beta_i(t)$. Le problème est résolu en $O(f(n,T))$, où T est une valeur suffisamment grande. Le temps de calcul global est en $O(nT)$.

Les auteurs proposent une application de la présente fonction récursive à plusieurs problèmes d'ordonnancement. Dans la section 10 du même article [150], les auteurs proposent de résoudre le problème d'ordonnancement $1|d_i = d|\sum w_i T_i$. Nous allons d'abord proposer une correction de la fonction de récurrence pour le cas d'une seule machine et montrons par la suite pourquoi ce programme dynamique tel qu'il est défini ne peut pas être généralisé au cas de m machines comme le suggèrent les auteurs.

Soient $(\mathcal{A})$, $(\mathcal{B})$ et $(\mathcal{C})$ respectivement l'ensemble des travaux en avance, des travaux en retard et le "*straddling job*". Sans perte de généralité, les travaux sont numérotés selon l'ordre WLPT (*Weighted Longest Processing Times*, c-à-d. $w_i/p_i \leq w_{i+1}/p_{i+1}$).

À l'itération k (J_k est considéré comme le "*straddling job*"), les auteurs notent par $f^{(k)}(n,t)$, la meilleure solution du sous-ensemble des travaux $\mathcal{J} \setminus \{J_k\} = \{J_1, J_2, \ldots, J_{k-1}, J_{k+1}, \ldots, J_n\}$ avec $\alpha_i(t) = 0$, $\beta_i(t) = w_i t$, $a_i = p_i$ et $b_i = 0$. Ainsi, la fonction récursive prend la forme suivante :

$$
\begin{aligned}
f^{(k)}(0,t) &= 0, && (\forall t \geq 0)\\
f^{(k)}(i,t) &= +\infty, && (\forall i \in \{0,1,\ldots,n\} \setminus \{k\}; \forall t < 0)\\
f^{(k)}(i,t) &= \min \left\{ \begin{array}{l} f^{(k)}(i,t-1), \\ f^{(k)}(i-1,t-p_i), \\ f^{(k)}(i-1,t) + w_i t \end{array} \right\}, && (\forall i \in \{1,\ldots,n\} \setminus \{k\}; \forall t \geq 0)
\end{aligned}
$$

Le deuxième terme de la fonction récursive correspond au travail où J_i est dans $(\mathcal{A})$ et se termine à l'instant t. Le dernier terme considère le cas où J_i est dans $(\mathcal{B})$.

Supposons que J_i est dans $(\mathcal{B})$. Comme les travaux sont rangés dans l'ordre WLPT, J_i doit être le premier travail en retard (voir Fig.2.1) (les travaux en retard sont ordonnancés selon WSPT). Avec $P_i = \sum_{j=1}^{i} p_i$ nous avons $C_i = P_n - P_{i-1} + t$ et le coût généré par J_i est égal à $w_i(C_i - d)$.

Ainsi, une meilleure expression de la relation de récurrence est :

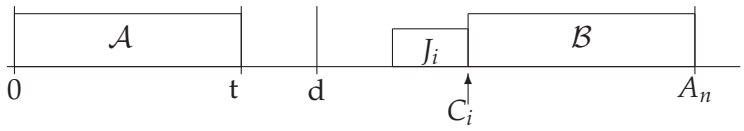

FIG. 2.1 – Cas d'une seule machine - position du nouveau travail en retard

$$
\begin{aligned}
f^{(k)}(0,t) &= 0, && (\forall t \geq 0)\\
f^{(k)}(i,t) &= +\infty, && (\forall i \in \{0,1,\ldots,n\} \setminus \{k\}; \forall t < 0)\\
f^{(k)}(i,t) &= \min \left\{ \begin{array}{l} f^{(k)}(i-1, t-p_i), \\ f^{(k)}(i-1,t) + w_i(P_n - P_{i-1} + t - d) \end{array} \right\}, && (\forall i \in \{1,\ldots,n\} \setminus \{k\}; \forall t \geq 0)
\end{aligned}
$$

Par conséquent, le coût de retard dû au "*straddling job*" est ajouté à $f^{(k)}(n,t)$ et les valeurs de k et t pour lesquelles $f^{(k)}(n,t) + w_k(t + p_k - d)$ est minimum indiquent l'identité du travail de la classe $(\mathcal{C})$ et sa date de fin d'exécution, tandis que $f^{(k)}(n,T)$ permet de déterminer les travaux des classes $(\mathcal{A})$ et $(\mathcal{B})$. En posant $T = \sum_i p_i$, notons que, dans le cas d'une seule machine, la date de fin d'exécution du dernier travail est toujours égale à P_n, ce qui correspond à la date d'achèvement des travaux (makespan) quelle que soit la séquence donnée. Ainsi, déterminer la date de fin d'exécution du "*straddling job*" ne pose aucun problème.

Les auteurs proposent ensuite de généraliser ce résultat au cas de m machines parallèles (voir la Section 11, [150]). Toutefois, la méthode proposée souffre de deux inconvénients majeurs : (1) contrairement au cas d'une seule machine, les auteurs ne donnent aucune information concernant la gestion des "*straddling jobs*" et (2) la date de fin d'exécution du dernier travail dépend de la machine et peut être supérieure à la valeur du makespan optimal donné par la résolution du $Pm||C_{\max}$. Reprenons ces deux remarques avec plus de détails :

1. Sur chaque machine, nous devons considérer un "*straddling job*". $\dfrac{n!}{(n.m)!}$ différentes possibilités doivent être envisagées, ce qui augmente significativement la complexité globale de l'algorithme annoncé de $O(nmT^m)$.

2. Par conséquent, les temps morts générés sur chaque machine entre les deux blocs de travaux $(\mathcal{A})$ et $(\mathcal{B})$ doivent être pris en considération. Ces temps morts augmentent ainsi la difficulté du problème et rendent la mise en oeuvre du programme dynamique proposé impossible.

En conclusion, selon la fonction de récurrence proposée par Lawler et Moore [150], il est impossible de calculer une solution optimale en $O(mnT^m)$. Notons également que pour le cas $m = 1$ (une seule machine), la complexité du programme dynamique est $O(nT)$. Or, dans la cas d'une seule machine les auteurs indiquent qu'une solution optimale est calculée en $O(n^2d)$.

2.3 Problème d'ordonnancement à une seule machine

Cette section est dédiée au problème $1|d_i = d|\sum w_iT_i$. Si $d = 0$ le problème est équivalent au $1||\sum w_iC_i$ et peut être résolu en $O(n \log n)$. Si $d > 0$, nous proposons ici un nouvel algorithme pseudo-polynomial en $O(n^2d^2)$. La complexité dans le cas d'une seule machine est plus importante que celle proposée dans [150], mais l'intérêt de cette nouvelle approche est qu'elle s'étend facilement au cas de m machines parallèles [117].

Rappelons qu'il existe une solution optimale sans aucun temps mort entre les travaux [150] (entre l'instant 0 et P). Les premiers travaux sont ordonnancés dans un ordre arbitraire et les travaux en retard sont exécutés dans l'ordre non-décroissant des p_i/w_i. Ainsi, il existe une solution optimale dans laquelle les travaux en avance sont également ordonnancés selon l'ordre non-décroissant des p_i/w_i.

2.3.1 Nouveau programme dynamique

Notre approche se résume comme suit : à partir d'un straddling job (n possibilités), nous déterminons l'ensemble des travaux qui vont être ordonnancés en avance pour lesquels la fonction coût est minimisée.

Partons d'une solution initiale où tous les travaux sont supposés en retard, il s'agit d'un problème de sac-à-dos où on cherche à déterminer l'ensemble des travaux à ordonnancer avant la date de début du "*straddling job*" et qui maximisent l'apport correspondant au coût réduit de la valeur de la solution initiale.

Reprenons la fonction de récurrence du problème de sac-à-dos. Soit H_i la valeur de l'objet i et soit ω_i son poids. Soit x_i la variable de décision, avec $x_i = 1$ si l'objet i est choisi et 0 sinon. L'objectif est de maximiser $\sum H_ix_i$ sous la contrainte $\sum \omega_ix_i \leq \Omega$. La fonction de récurrence du problème de sac-à-dos est :

$$h(i,\omega) = \max(h(i-1,\omega), h(i-1,\omega-\omega_i) + H_i), \forall i \in \{1,\ldots,n\}, \forall \omega \in \{0,\ldots,\Omega\}$$

Si l'on considère que la valeur de chaque tâche correspond à son coût de retard, l'objectif est de mettre dans le sac-à-dos (ensemble de travaux qui vont être ordonnancés en avance) les éléments pour un apport total maximal, c'est-à-dire les travaux qui contribuent le plus au coût du retard pondéré total. La capacité du sac, i.e. le poids Ω est la date de fin souhaitée commune et le poids d'un travail est égal à sa durée opératoire. La seule différence avec le problème du sac-à-dos est la définition de H_i qui doit être calculée en fonction de ω (c'est une fonction $H_i(\omega)$).

Une solution optimale est calculée comme suit.

1. à l'itération k ($1 \leq k \leq n$), soit J_k le *straddling job*.
2. pour chaque date de fin d'exécution possible de J_k, notée $C_k = \ell$ ($C_k \in [d, d + p_k - 1]$), nous construisons une solution optimale à l'aide du programme dynamique.
3. La solution optimale est obtenue pour le meilleur couple (J_k, C_k) possible.

Par la suite, nous supposons que les travaux sont numérotés selon l'ordre SPT ($p_i/w_i \leq p_{i+1}/w_{i+1}$).

Pour le "*straddling job*" J_k qui finit à l'instant $\ell = C_k$ nous construisons une solution initiale où tous les travaux sauf J_k sont ordonnancés en retard, c-à-d. qu'ils sont ordonnancés après C_k (il n'y a pas de travaux en avance). Cette séquence initiale est notée par $S_0^{(k,\ell)}$. On note $C_{i,0}^{(k,\ell)}$ et $wT_{i,0}^{(k,\ell)}$ la date de fin d'exécution et la pénalité pondérée de retard du travail J_i dans $S_0^{(k,\ell)}$. Nous avons :

$$C_{i,0}^{(k,\ell)} = \ell + \sum_{a=1, a\neq k}^{i} p_a, \qquad wT_{i,0}^{(k,\ell)} = w_j(C_{i,0}^{(k,\ell)} - d)$$

Le coût global de $S_0^{(k,\ell)}$ est égal à $\sum_{i=1, i\neq k}^{n} wT_{i,0}^{(k,\ell)} + w_k(\ell - d)$.

Soit $g^{(k,\ell)}(i,t)$ la réduction maximale du coût de la solution initiale pour les travaux $\{J_1, \ldots, J_i\}$, sous réserve que la date de fin d'exécution du "*straddling job*" J_k est ℓ et que la date de fin de traitement des travaux en avance appartenant au sous-ensemble $\{J_1, \ldots, J_i\}$ est t avec $t \leq \ell - p_k < d$. On note A le sous-ensemble des travaux en avance et B pour le sous-ensemble des travaux en retard de $\{J_1, \ldots, J_{i-1}\}$. La figure 2.2 présente la situation à la phase i. σ représente la séquence des travaux dans $\mathcal{J} \setminus \{J_1, \ldots, J_i\}$ rangés selon l'ordre de WSPT.

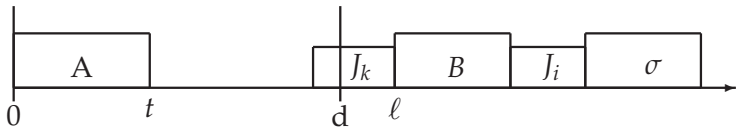

FIG. 2.2 – Cas d'une seule machine - nouvelle fonction de récurrence

Deux décisions sont alors possibles pour le travail J_i :

1. soit le travail J_i sera ordonnancé en avance, et il appartient à A (voir Fig. 2.3). Pour le problème sac-à-dos, cela correspond au cas où $x_i = 1$.

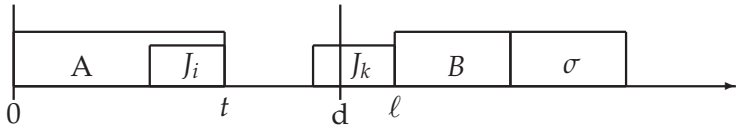

FIG. 2.3 – Cas d'une seule machine - travail J_i est en avance

2. soit le travail J_i restera en retard et appartient alors à B (voir Fig. 2.4). C'est le cas où $x_i = 0$.

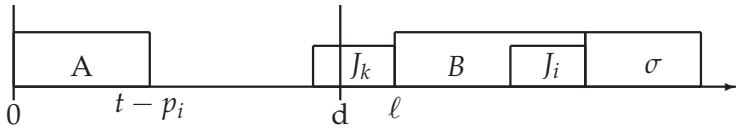

FIG. 2.4 – Cas d'une seule machine - travail J_i est en retard

L'objectif est de minimiser le coût des travaux en retard. Autrement dit, à partir d'une instance, le but du jeu est de choisir un sous-ensemble de travaux en retard le plus coûteux. Le problème correspond à un sac-à-dos.

Nous devons maintenant déterminer les coefficients correspondant à H_i. Il est clair que ces coefficients dépendent de la date de fin d'exécution des travaux en retard (ou sur la valeur de t), c'est-à-dire sur les décisions prises précédemment. On note par $G^{(k,\ell)}(i,t)$ la valeur de travail J_i s'il est décidé 'en avance'. La relation de récurrence sera sous la forme :

$$g^{(k,\ell)}(i,t) = \max(g^{(k,\ell)}(i-1,t), g^{(k,\ell)}(i-1,t-p_i) + G^{(k,\ell)}(i,t))$$

- Si le travail J_i est décidé 'en retard', le coût ne change pas et $g^{(k,\ell)}(i,t) = g^{(k,\ell)}(i-1,t)$.
- D'autre part, si le travail J_i est décidé 'en avance', le coût diminue : le travail J_i

n'est pas en retard et les travaux de σ peuvent commencer plus tôt. Rappelons que selon la solution initiale, tous les travaux sont en retard. Plus précisément, nous avons :

- à l'itération i, certains travaux ont été décidés 'en avance' et donc le travail J_i a été implicitement déplacé à gauche. On note $C_{j,i}^{(k,\ell)}$ pour sa date de fin d'exécution à l'itération i. La contribution du travail J_i à la valeur de $G^{(k,\ell)}(i,t)$ est égale à $w_i(C_{i,i-1}^{(k,\ell)} - d)$.
- les travaux de σ peuvent être déplacés à gauche. La contribution de ces travaux exécutés en avance pendant p_i unité de temps à la valeur de $G^{(k,\ell)}(i,t)$ est de $p_i \times \sum_{j\in\sigma} w_j$.

Nous avons : $C_{i,i-1}^{(k,\ell)} = C_{i,0}^{(k,\ell)} - (t - p_i)$.

Ainsi, $w_i(C_{i,i-1}^{(k,\ell)} - d) = w_i(C_{i,0}^{(k,\ell)} - (t - p_i) - d) = wT_{i,0}^{(k,\ell)} - w_i(t - p_i)$.

Finalement, le coût de réduction si le travail J_i est 'en avance' correspond à la valeur $G^{(k,\ell)}(i,t) = wT_{i,0}^{(k,\ell)} - w_i(t - p_i) + p_i \times \sum_{j\in\sigma} w_j$.

La relation de récurrence est la suivante :

$$
\begin{aligned}
& g^{(k,\ell)}(0,0) = 0 \\
& g^{(k,\ell)}(0,t) = -\infty, && (\forall t > 0) \\
& g^{(k,\ell)}(i,t) = -\infty, && (\forall i \in \{0,1,\ldots,n\} \setminus \{k\}; \forall t < 0) \\
& g^{(k,\ell)}(i,t) = \max \left\{ \begin{array}{l} g^{(k,\ell)}(i-1,t), \\ g^{(k,\ell)}(i-1,t-p_i) + G^{(k,\ell)}(i,t) \end{array} \right\}, && (\forall i \in \{1,\ldots,n\} \setminus \{k\}; \forall 0 \le t \le d) \\
& g^{(k,\ell)}(i,t) = g^{(k,\ell)}(i,\ell - p_k), && (\forall i \in \{0,1,\ldots,n\} \setminus \{k\}, \forall t > \ell - p_k) \\
& \text{avec } G^{(k,\ell)}(i,t) = w_i(\ell + \sum_{j=1,j\neq k}^{i} p_j - d) - w_i \times (t - p_i) + p_i \times \sum_{j\in\sigma} w_j
\end{aligned}
$$

L'objectif est donc de minimiser pour les valeurs de k et de ℓ telles que :

$$\sum_{i=1,i\neq k}^{n} w_i(\ell + \sum_{j=1,j\neq k}^{i} p_j - d) - g^{(k,\ell)}(n,\ell - p_k) + w_k \times (\ell - d)$$

Pour chaque J_k et C_k, la complexité de l'algorithme est en $O((n-1) \times (C_k - p_k))$ qui est bornée par $O(nd)$. Comme indiqué ci-dessus, nous avons n choix pour le straddling job J_k et $\min(d, p_k)$ possibilités de $\ell = C_k$. Ainsi, la complexité globale est bornée par $O(n^2d^2)$.

Remarque. La complexité de ce nouveau PrgDyn est plus grande que celle proposée par Lawler et Moore [150]. Cette augmentation est due à la nécessité de déterminer la date de fin d'exécution du "*straddling job*" afin de calculer la contribution des coût des travaux en retard. En effet, selon notre programme dynamique, nous devons dé-

Algorithme WTSMCDD : $1|d_i = d|\sum w_i T_i$

1 : Ordonner les travaux selon l'ordre non-décroissant de WSPT //
2 : **Pour** ($k = 1$ **à** n) et ($\ell = d$ **à** $d + p_k - 1$)**faire**
3 : | // Soit J_k le "*straddling job*" et $C_k = \ell$ //
4 : | $C_{j,0}^{(k,\ell)} = \ell + \sum_{a=1,a\neq k}^{j} p_a$; $wT_{j,0}^{(k,\ell)} = w_j(C_{j,0}^{(k,\ell)} - d)$
5 : | $g^{(k,\ell)}(0,0) = 0$; $g^{(k,l)}(0,t)$ = -∞, si $t \neq 0$
6 : | $P^{(k)}(i) = \sum_{j=1,j\neq k}^{i} p_j$; $W^{(k,\ell)}(1,0) = \sum_{j=2,j\neq k}^{n} w_j$
7 : | **Pour** $i = 1$ **à** n, $i \neq k$ **faire**
8 : | | **Pour** $t = 0$ **à** $d - 1$ **faire**
9 : | | | $G^{(k,\ell)}(i,t) = w_i(\ell + P^{(k)}(i) - d) - w_i \times (t - p_i) + p_i W^{(k,\ell)}(i,t)$
10 : | | | $g^{(k,l)}(i,t) = \max(g^{(k,l)}(i-1,t), g^{(k,\ell)}(i-1,t-p_i) + G^{(k,\ell)}(i,t)$
11 : | | | **Si** ($F^{(k,l)}(i,t) = F^{(k,l)}(i-1,t)$) **alors**
12 : | | | | $W^{(k,\ell)}(i+1,t) = W^{(k,\ell)}(i,t)$
13 : | | | **Sinon**
14 : | | | | $W^{(k,\ell)}(i+1,t) = W^{(k,\ell)}(i,t-p_i) - w_i$
15 : | | | **FinSi**
16 : | | **FinPour**
17 : | **FinPour**
18 : | $S^{(k,\ell)} = \min_{t=d-p_k}^{d-1}(\sum_{j=1,j\neq k}^{n} wT_{j,0}^{(k,\ell)} - g^{(k,\ell)}(n,\ell-p_k) + w_k \times (\ell - d))$
19 : **FinPour**
20 : La solution optimale correspond à la solution définie par (k,ℓ) dont le coût de S est minimum.

FIG. 2.5 – Algorithme WTSMCDD

terminer la date de début de straddling job. On peut penser que le straddling job soit Jàt, ensuite on détermine la meilleure décision des travaux en avance-retard, et enfin on détermine la meilleure date de fin d'exécution du "*straddling job*" par rapport à la meilleure séquence trouvée. Malheureusement, cette stratégie ne garantit pas l'optimalité de la meilleure solution obtenue. L'exemple suivant (Table 2.3) montre les lacunes de cette stratégie .

TAB. 2.3 – Durées opératoires, dates de fin souhaitées, et pénalités

	J_1	J_2	J_3	J_4	J_5
p_i	4	3	12	16	9
d_i	17	17	17	17	17
w_i	14	9	13	17	4

L'erreur apparaît dès l'itération où le "*straddling job*" est défini par $k = 2$ ($p_k = 3$, $w_k = 9$). La séquence trouvée est $(4, \mathbf{2}, 1, 3, 5)$ avec un coût de 444, mais l'optimal est en fait $(1, 3, \mathbf{2}, 4, 5)$ avec un coût de 432. Sans insertion du "*straddling job*", la séquence $(4, 1, 3, 5)$ domine $(1, 3, 4, 5)$. Par conséquent, la meilleure solution trouvée est $(4, \mathbf{1}, 2, 3, 5)$ avec le coût de 438. Par conséquent, la complexité du nouveau PrgDyn ne

peut pas être réduite à $O(n^2 d)$.

2.4 Problème d'ordonnancement à machines identiques

Nous allons maintenant examiner le problème d'ordonnancement des travaux indépendants sur m machines parallèles identiques [115]. Ce problème est noté $Pm|d_i = d|\sum w_i T_i$. Si $n \leq m$, le problème peut être résolu en temps polynomial en ordonnançant un travail par machine. Sinon, soit S^* une solution obtenue en résolvant le problème $Pm||C_{\max}$ par le programme dynamique de Rothkopf (en $O(n(P_n)^m)$) [187] où $P_n = \sum_{i=1}^{n} p_i$. Si $C_{max}(S^*) \leq d$, S^* est également optimale pour $Pm|d_i = d|\sum w_i T_i$ depuis $\sum w_i T_i = 0$.

Par la suite, nous considérons le cas où $C_{max}(S^*) > d$. La propriété suivante est nécessaire pour définir la formule de récurrence du programme dynamique.

Proposition 2.4.1 *Il existe un ordonnancement optimal où sur chaque machine nous avons :*

1. *un "*straddling job*",*
2. *pas de temps morts entre les travaux en retard (incluant le straddling job),*
3. *les travaux en avance sont ordonnancés dans un ordre arbitraire,*
4. *les travaux en retard entièrement (sans considérer le "*straddling job*") sont ordonnancés selon l'ordre WSPT.*

Preuve. Les points (2-4) sont triviaux (voir section précédente et cf. [150]). Le dernier point est aussi valable pour les machines avec un coût de retard nul : sur ces machines, le dernier travail est considéré comme le "*straddling job*" pour lequel sa date de fin d'exécution est d. Il suffit donc de déplacer le dernier travail vers la droite jusqu'à d sans augmenter le coût total. Dans ce cas, il est possible d'avoir un temps mort entre l'ensemble des travaux en avance et le "*straddling job*".

2.4.1 Programme dynamique pour le cas de machines identiques

Considérons J_{k_j} le straddling job sur M_j, $1 \leq j \leq m$. Nous notons par $\mathcal{J}_{str}$ l'ensemble des straddling jobs. Soit

$$g^{(k_1,k_2,\ldots,k_m,\ell_1,\ell_2,\ldots,\ell_m)}(i, t_1, t_2, \ldots, t_m, \mathcal{T}_2, \mathcal{T}_3, \ldots, \mathcal{T}_m)$$

le coût total minimal pour les travaux $\{J_1, \ldots, J_i\}$, sous réserve que le sous-ensemble des travaux en avance de $\{J_1, \ldots, J_i\} \setminus \mathcal{J}_{str}$ se termine à l'instant $t_j < d$ sur M_j $\forall j$, $1 \leq j \leq m$, la date de fin d'exécution de straddling job J_{k_j} sur M_j est égale à ℓ_j et la date de fin d'exécution du dernier travail sur M_j est $\mathcal{T}_j$ $\forall j, 2 \leq j \leq m$.

Sans perte de généralité, nous illustrons notre démarche au cas de deux machines ($m = 2$).

À l'état initial, nous choisissons deux "*straddling jobs*" J_{k_1} et J_{k_2} avec date de fin d'exécution ℓ_1 et ℓ_2. Ensuite, nous supposons que tous les $(n-2)$ travaux restants sont ordonnancés en retard sur M_1 après J_{k_1} selon la règle WSPT. On note cette solution globale $S_0^{(k_1,k_2,\ell_1,\ell_2)}$.

Ensuite, nous affectons chaque travail à une machine, en position en avance ou en retard, et donc nous devons déterminer les trois sous-ensembles optimaux A_1, A_2 et B_2 où :

1. A_1 correspond aux travaux ordonnancés en avance sur M_1,
2. A_2 correspond aux travaux ordonnancés en avance sur M_2,
3. B_2 correspond aux travaux ordonnancés en retard sur M_2.

Le reste des travaux correspond à B_1, c'est-à-dire aux travaux ordonnancés en retard sur M_1 (globalement, A_j est le sous-ensemble des travaux sur M_j et B_j le sous-ensemble des travaux en retard sur M_j).

La fonction $g^{(k_1,k_2,\ell_1,\ell_2)}(i, t_1, t_2, \mathcal{T}_2)$ est la valeur minimale du coût total pour les travaux $\{J_1, \ldots, J_i\} \setminus \{J_{k_1}, J_{k_2}\}$, sous réserve que le sous-ensemble des travaux de $\{J_1, \ldots, J_i\} \setminus \{J_{k_1}, J_{k_2}\}$ ordonnancé en avance sur M_1 se termine à l'instant $t_1 \leq \ell_1 - p_{k_1} < d$ et à la date $t_2 \leq \ell_2 - p_{k_2} < d$ sur M_2. La Figure 2.6 présente la situation à la phase i. σ représente la séquence des travaux dans $\mathcal{J} \setminus \{J_1, \ldots, J_i\}$ rangés selon l'ordre WSPT.

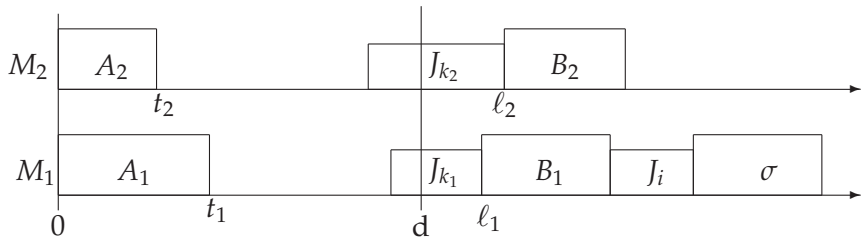

FIG. 2.6 – Fonction récurrente du cas de machines parallèles

Soit $wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)} = w_i \times (\ell_1 + \sum_{a=1, a\neq k_1, a\neq k_2}^{i} p_a - d)$ le coût engendré par le travail J_i

dans la solution initiale et $G_u^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2)$ le coût de reduction qui correspond à la décision u concernant le travail J_i à l'état $(t_1,t_2,\mathcal{T}_2)$.

Soit $wT^{(k_1,k_2,\ell_1,\ell_2)} = \sum_{i=1,i\notin\{k_1,k_2\}}^{n} wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$ le coût total. À l'itération (k_1,k_2,ℓ_1,ℓ_2) la fonction de coût doit être maximisée. Le coût final est alors désigné par $g^{(k_1,k_2,\ell_1,\ell_2)}(n,t_1,t_2,\mathcal{T}_2)$.

Nous suivons le même raisonnement que dans la section 2.3.1. Quatre décisions sont alors à considérer pour le travail J_i :

$u=1$: J_i est en avance sur M_1 ($J_i \in A_1$, voir Figure 2.7). Par conséquent,

$$g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-p_i,t_2,\mathcal{T}_2) + G_1^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2)$$

avec $G_1^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)} - w_i \times (t_1 - p_i + t_2 + \mathcal{T}_2 - \ell_2) + p_i \times \sum_{j\in\sigma} w_j$

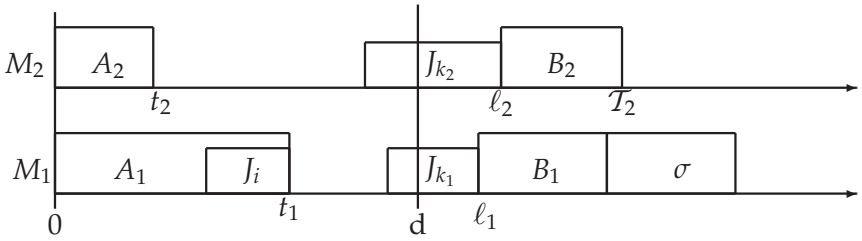

FIG. 2.7 – Cas 1 - J_i est en avance sur M_1

$u=2$: J_i est en retard sur M_1 ($J_i \in B_1$, voir Figure 2.8). Par conséquent,

$$g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2) + G_2^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2)$$

avec $G_2^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = 0$.

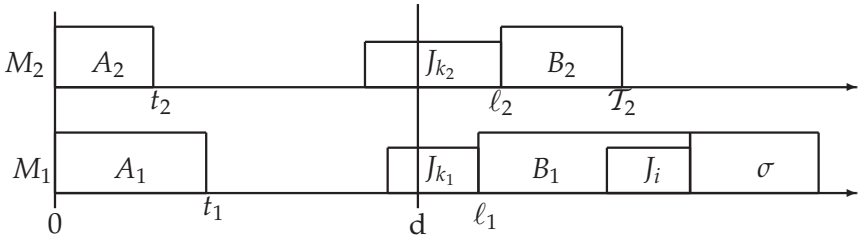

FIG. 2.8 – Cas 2 - J_i est en retard sur M_1

$u=3$: J_i est en avance sur M_2 ($J_i \in A_2$, voir Figure 2.9). Par conséquent,

$$g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-p_i,\mathcal{T}_2) + G_3^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2)$$

avec $G_3^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)} - w_i \times (t_1 + t_2 - p_i + \mathcal{T}_2 - \ell_2) + p_i \times \sum_{j\in\sigma} w_j$

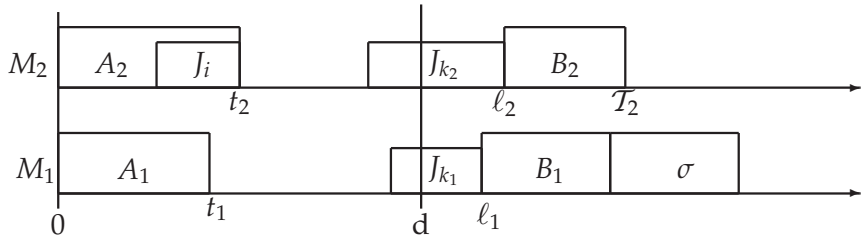

FIG. 2.9 – Cas 3 - J_i est en avance sur M_2

$u = 4$: J_i est en retard sur M_2 ($J_i \in B_2$, voir Figure 2.10). Par conséquent,

$$g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2 - p_i) + G_4^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2)$$

avec $G_4^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)} - w_i \times (t_1 + t_2 + \mathcal{T}_2 - p_i - \ell_2) + p_i \times \sum_{j\in\sigma} w_j - w_i \times (\mathcal{T}_2 - d)$

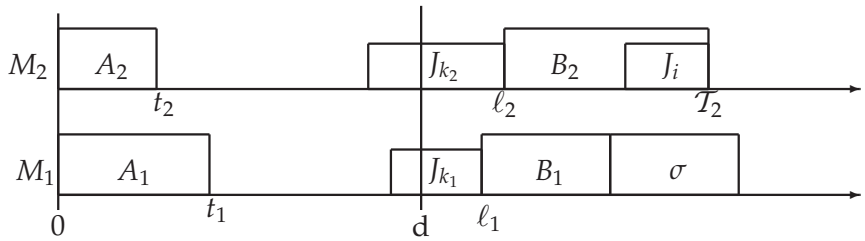

FIG. 2.10 – Cas 4 - J_i est en retard sur M_2

La relation récursive générale peut être facilement déduite à partir de ces décisions. Nous cherchons les valeurs de k_1, k_2, ℓ_1, ℓ_2, t_1, t_2 et $\mathcal{T}_2$ telles que :

$$wT^{(k_1,k_2,\ell_1,\ell_2)} - g^{(k_1,k_2,\ell_1,\ell_2)}(n,t_1,t_2,\mathcal{T}_2) + \sum_{j=1}^{2} w_{k_j} \times (\ell_j - d)$$

soit minimum.

Il y a $\frac{n\times(n-1)}{2}$ possibilités pour le choix du *"straddling job"*, d^4 possibilités pour (ℓ_1,ℓ_2,t_1,t_2) et P_n possibilités pour $\mathcal{T}_2$. Le temps de calcul de cet algorithme est donc en $O(n^3d^4P)$.

Cet algorithme peut être généralisé au cas de m machines. Par conséquent, pour tout $m \geq 1$ fixé, nous pouvons déterminer un algorithme de PrgDyn pseudo-polynomial avec une complexité de $O(n^{m+1}d^{2m}P^{m-1})$. On en déduit le théorème suivant.

Théorème 2.4.2 *Une solution optimale peut être déterminée par le programme dynamique précédent pour le problème* $Pm|d_i = d|\sum w_i T_i$ *en un temps de calcul borné par* $O(n^{m+1}d^{2m}P^{m-1})$ *avec* $P = \sum_{i=1}^{n} p_i$.

Remarque 2.4.1 *L'algorithme PrgDyn peut résoudre le problème* $Pm||\sum w_i C_i$, *il suffit de considérer* $d = 0$.

Remarque 2.4.2 *Dans le cas où* $m = 1$, *l'algorithme PrgDyn est le même que celui décrit dans la Section 2.3.1.*

2.4.2 Extension au cas de machines uniformes

Nous allons maintenant étudier l'ordonnancement des travaux indépendants sur m machines parallèles uniformes [116]. Le problème est noté par $Qm|d_i = d|\sum w_i T_i$. La vitesse de la machine M_j est donnée par γ_j. Sans perte de généralité, nous supposons que $\gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_m$ et que les vitesses γ_j sont entières. La durée opératoire de travail J_i sur machine M_j est donnée par $p_{i,j} = p_i \times \gamma_j$ (i.e. la machine M_j plus performante que la machine M_{j+1}). Dans la suite, $P = \sum_{i=1}^{n} p_{i,m}$.

Notons que la Proposition 2.4.1 de Section 2.4 reste valable. Cependant, nous avons dans ce cas de nouvelles propriétés :

Proposition 2.4.3 *Pour une solution optimale, soit la machine* M_k *où aucun travail n'est en avance. En plus, s'il n'y a pas de "*straddling job*" exécuté sur* M_k, *alors aucun travail n'est affecté à* M_j *pour tout* $j \geq k$.

Preuve. Si un travail est exécuté sur M_k, alors déplacer ce travail vers la gauche diminue le coût total, ce qui contredit le fait que la solution est optimale. Donc aucun travail n'est alors affecté à M_k. Comme les machines sont numérotées selon l'ordre décroissant de leur performance, alors il n'y a aucun travail exécuté sur n'importe quelle autre machine $M_{k'}$ avec $k' > k$.

Notons que si tous les travaux exécutés sur la machine M_k sont en avance, le dernier travail peut être déplacé vers la droite jusqu'à la date d sans augmenter le coût. Ce travail peut donc être considéré comme "*straddling job*".

Ainsi, nous pouvons supposer qu'il existe m' "*straddling jobs*" sur les m' premières machines. Nous pouvons donc déduire la fonction de récurrence du programme dynamique comme suit.

Algorithme WTP2CDD : $P2|d_i = d|\sum w_i T_i$

1 : Ordonner les travaux selon l'ordre non-décroissant de p_i/w_i
2 : **Pour** $k_1 \in \{1 \dots n\}$ **et** $k_2 \in \{1 \dots n\}$ **et** $k_1 \neq k_2$ **faire**
3 : /*Soient J_{k_1}, J_{k_2} les "*straddling jobs*" sur M_1 et M_2, respectivement.*/
4 : **Pour** $\ell_1 \in \{d, \dots, d + p_{k_1} - 1\}$ **et** $\ell_2 \in \{d, \dots, d + p_{k_2} - 1\}$ **faire**
5 : $C_{j,0}^{(k_1,k_2,\ell_1,\ell_2)} = \ell_1 + \sum_{a=1, a\notin\{k_1,k_2\}}^{j} p_a$
6 : $wT_{j,0}^{(k_1,k_2,\ell_1,\ell_2)} = w_j(C_{j,0}^{(k_1,k_2,\ell_1,\ell_2)} - d)$
7 : $wT^{(k_1,k_2,\ell_1,\ell_2)} = \sum_{j=1, j\notin\{k_1,k_2\}}^{n} wT_{j,0}^{(k_1,k_2,\ell_1,\ell_2)}$
8 : $g^{(k_1,k_2,\ell_1,\ell_2)}(0,0,0,\ell_2) = 0$
9 : $W^{(k_1,k_2,\ell_1,\ell_2)}(0,0,0,\ell_2) = \sum_{j=1}^{n}(w_j) - w_{k_1} - w_{k_2}$
10 : $g^{(k_1,k_2,\ell_1,\ell_2)}(0,t_1,t_2,\mathcal{T}_2)$ = -∞, si $t_1, t_2 \neq 0$ et $\mathcal{T}_2 \neq (\ell_2)$
11 : $W^{(k_1,k_2,\ell_1,\ell_2)}(0,t_1,t_2,\mathcal{T}_2)$ = -∞, si $t_1, t_2 \neq 0$ et $\mathcal{T}_2 \neq (\ell_2)$
12 : **Pour** $i \in \{1, \dots, n\}, i \neq k, i \neq l$ **faire**
13 **Pour** $t_1 \in \{0, \dots, S_k\}$ **et** $t_2 \in \{0, \dots, S_l\}$ **et** $t_3 \in \{0, \dots, P\}$ **faire**
14 : $m_0 = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2)$
15 : $m_1 = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-p_i,t_2,\mathcal{T}_2) + wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$
$-w_i(t_1+t_2+\mathcal{T}_2-p_i) + p_i W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-p_i,t_2,\mathcal{T}_2)$
16 : $m_2 = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-p_i,\mathcal{T}_2) + wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$
$-w_i(t_1+t_2+\mathcal{T}_2-p_i) + p_i W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-p_i,\mathcal{T}_2)$
17 : $m_3 = F^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2-p_i) + wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$
$-w_i(t_1+t_2+\mathcal{T}_2-p_i) + p_i W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2-p_i) - w_i(\mathcal{T}_2-d)$
18 : $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = \max\{m_0,m_1,m_2,m_3\}$
19 : **Si** $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = m_0$ **alors**
20 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2)$
21 : **Sinon**
22 : **Si** $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = m_1$ **alors**
23 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-p_i,t_2,\mathcal{T}_2) + w_i$
24 : **Sinon**
25 : **Si** $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = m_2$ **alors**
26 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-p_i,\mathcal{T}_2) + w_i$
27 : **Sinon**
28 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2-p_i) + w_i$
29 : **FinSi**
30 : **FinSi**
31 : **FinSi**
32 : **FinPour**
33 : **FinPour**
34 : $wT_{min}^{(k_1,k_2,\ell_1,\ell_2)} = wT^{(k_1,k_2,\ell_1,\ell_2)} + \sum_{j=1}^{2} w_{k_j} \times (\ell_j - d) - \max_{(t_1,t_2,\mathcal{T}_2)} g^{(k_1,k_2,\ell_1,\ell_2)}(n,t_1,t_2,\mathcal{T}_2)$
35 : **FinPour**
36 : **FinPour**
37 : $wT_{min} = \min_{(k_1,k_2,\ell_1,\ell_2)} wT_{min}^{(k_1,k_2,\ell_1,\ell_2)}$
38 : La solution optimale correspond à $(k_1,k_2,\ell_1,\ell_2,t_1,t_2,\mathcal{T}_2)$ dont le coût est wT_{min}.

FIG. 2.11 – Algorithme WTP2CDD

Programme dynamique au cas de machines uniformes

En suivant la même approche présentée dans les sous-sections précédentes, nous montrons qu'une solution peut être déterminée en un temps pseudo-polynomial selon les trois étapes suivantes :

1. Considérons m' "*straddling jobs*". Au maximum, nous avons C_n^1 possibilités si $m' = 1$, C_n^2 si $m' = 2$, etc. Ainsi, au plus nous avons $\sum_{p=1}^{m} C_n^p$ possibilités bornées par mn^m (avec $C_n^p = \binom{p}{n} = \frac{n!}{p!(n-p)!}$). Ensuite, nous fixons la date de fin d'exécution ℓ_j de chaque "*straddling job*" J_{k_j} dans $[d, d-1+p_{k_j}]$, $\forall j, 1 \leq j \leq m'$.
2. Nous considérons une solution initiale où tous les autres travaux sont ordonnancés sur la première machine après la date de fin d'exécution de J_{k_1} selon la règle WSPT. Cela signifie qu'aucun travail n'est en avance.
3. Avec la fonction récursive g décrite ci-dessous, nous déterminons l'ensemble des travaux à exécuter sur chaque machine, en avance ou en retard.

À l'itération i, la fonction récursive notée par $g^{(k_1,...,k_{m'},\ell_1,...,\ell_{m'})}(i, t_1, \ldots, t_{m'}, \mathcal{T}_2, \mathcal{T}_3, \ldots, \mathcal{T}_{m'})$ représente le coût minimum engendré par l'ordonnancement des travaux du sous-ensemble de $\{J_1, \ldots, J_i\} \setminus \{J_{k_1}, ..., J_{k_{m'}}\}$ en avance sur M_j dans l'intervalle $[0, t_j]$ avec $t_j < d$.

Selon la solution initiale, soit $wT_{i,0}$ le coût de retard du travail J_i et soit $wT = \sum wT_{i,0}$ le coût total des travaux en retard (afin de simplifier les notations, les indices k_j et ℓ_j seront omis). À l'itération i, soit $L_i = (\bigcup_{j=1}^{m'} A_j) \cup (\bigcup_{j=2}^{m'} B_j)$ et soit $P_i' = \sum_{h \in L_i} p_h$. Une autre expression de P_i' est donnée par :

$$P_i' = t_1 + \sum_{j=2}^{m'} (t_j + \mathcal{T}_j - \ell_j)/\gamma_j$$

Et soit W_i' la somme des poids des travaux ordonnancés après J_i dans S^0. Nous définissons le coût engendré par la décision u comme suit (afin de simplifier les notations, les indices k_j and ℓ_j sont omis) :

- G_2 correspond au cas où J_i est ordonnancé en retard sur M_1 :
  $G_2(i, t_1, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_{m'}) = g(i-1, t_1, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_{m'})$
- $G_{2j-1}, (1 \leq j \leq m')$, correspond au cas où J_i est ordonnancé en avance sur M_j :
  $G_{2j-1}(i, t_1, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_{m'})$

$= g(i-1, t_1, \ldots, t_j - p_i\gamma_j, t_{j+1}, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_{m'}) + wT_{i,0} - w_i \times (P'_i\gamma_1 - p_i\gamma_1) + p_i\gamma_1 \times \sum_{j\in\sigma} w_j$

– G_{2j}, $(2 \leq j \leq m')$, correspond au cas où J_i est ordonnancé en retard sur M_j :
$G_{2j}(i, t_1, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_{m'})$
$= g(i-1, t_1, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_j - p_i\gamma_j, \ldots, \mathcal{T}_{m'}) + wT_{i,0} - w_i \times (P'_i\gamma_1 - p_i\gamma_1) + p_i\gamma_1 \times \sum_{j\in\sigma} w_j - w_i \times (\mathcal{T}_j - d)$

La valeur optimale de l'algorithme est donnée par

$$wT - \max_{t_j, \mathcal{T}_j} g(n, t_1, \ldots, t_{m'}, \mathcal{T}_2, \ldots, \mathcal{T}_{m'}) + \sum_{j=1}^{m'} w_{k_j} \times (\ell_j - d).$$

Le temps de calcul global dans le cas de m' "*straddling jobs*" (aucun travail n'est affecté à une machine M_j avec $j > m'$) est borné par $O(n^{m'+1}d^{2m'}P^{m'-1})$. Nous déduisons le théorème suivant.

Théorème 2.4.4 *Une solution optimale pour le problème $Qm|d_i = d|\sum w_iT_i$ peut être déterminée en $O(mn^{m+1}d^{2m}P^{m-1})$.*

Remarque 2.4.3 *Le problème avec des machines non-reliées ne peut pas être résolu par notre approche car l'ordre WSPT n'est pas le même pour toutes les machines.*

2.5 Approximabilité

Nous présentons dans cette sous-section la preuve de Kovalyov et Werner [141] montrant que aucun algorithme d'approximation en temps polynomial n'existe pour le problème d'ordonnancement à machines parallèle avec l'objectif de minimisation des retards sur une date de fin souhaitée commune. Nous en déduisons que les problèmes $Pm|d_i = d|\sum w_iT_i$ et $Pm|d_i = d|\sum w_iT_i$ n'acceptent aucun algorithme d'approximation, ni FPTAS ni PTAS.

Notons que le problème est $\mathcal{NP}$-difficile même pour $m = 1$. Dans le cas de m machines, si m n'est pas fixé alors le problème est $\mathcal{NP}$-difficile au sens fort parce que le problème ne peut pas être plus facile que $P||C_{\max}$ [71]. En effet, la réponse à la question "Existe-t-il un ordonnancement avec valeur $\sum T_j \leq 0$?" ne peut pas être obtenue en temps polynomial sauf si $\mathcal{P} = NP$.

Algorithme WTQ2CDD : $Q2|d_i = d|\sum w_i T_i$

1 : Ordonner les travaux selon l'ordre non-décroissant de p_i/w_i
2 : **Pour** $k_1 \in \{1 \dots n\}$ **et** $k_2 \in \{1 \dots n\}$ **et** $k_1 \neq k_2$ **faire**
3 : /*Soient J_{k_1}, J_{k_2} les "*straddling job*" sur M_1 et M_2, respectivement.*/
4 : **Pour** $\ell_1 \in \{d, \dots, d + \gamma_1 p_{k_1} - 1\}$ **et** $\ell_2 \in \{d, \dots, d + \gamma_2 p_{k_2} - 1\}$ **faire**
5 : $C_{j,0}^{(k_1,k_2,\ell_1,\ell_2)} = \ell_1 + \sum_{a=1, a \notin \{k_1,k_2\}}^{j} \gamma_1 p_a$
6 : $wT_{j,0}^{(k_1,k_2,\ell_1,\ell_2)} = w_j(C_{j,0}^{(k_1,k_2,\ell_1,\ell_2)} - d)$
7 : $wT^{(k_1,k_2,\ell_1,\ell_2)} = \sum_{j=1, j \notin \{k_1,k_2\}}^{n} wT_{j,0}^{(k_1,k_2,\ell_1,\ell_2)}$
8 : $g^{(k_1,k_2,\ell_1,\ell_2)}(0,0,0,\ell_2) = 0$
9 : $W^{(k_1,k_2,\ell_1,\ell_2)}(0,0,0,\ell_2) = \sum_{j=1}^{n}(w_j) - w_{k_1} - w_{k_2}$
10 : $g^{(k_1,k_2,\ell_1,\ell_2)}(0,t_1,t_2,\mathcal{T}_2)$ = -∞, si $t_1, t_2 \neq 0$ et $\mathcal{T}_2 \neq (\ell_2)$
11 : $W^{(k_1,k_2,\ell_1,\ell_2)}(0,t_1,t_2,\mathcal{T}_2)$ = -∞, si $t_1, t_2 \neq 0$ et $\mathcal{T}_2 \neq (\ell_2)$
12 : **Pour** $i \in \{1, \dots, n\}, i \neq k, i \neq l$ **faire**
13 **Pour** $t_1 \in \{0, \dots, S_k\}$ **et** $t_2 \in \{0, \dots, S_l\}$ **et** $t_3 \in \{0, \dots, P\}$ **faire**
14 : $m_0 = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2)$
15 : $m_1 = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-\gamma_1 p_i,t_2,\mathcal{T}_2) + wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$
$-w_i(t_1+t_2+\mathcal{T}_2-\gamma_1 p_i) + p_i W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-\gamma_1 p_i,t_2,\mathcal{T}_2)$
16 : $m_2 = g^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-\gamma_2 p_i,\mathcal{T}_2) + wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$
$-w_i(t_1+t_2+\mathcal{T}_2-\gamma_1 p_i) + p_i W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-\gamma_2 p_i,\mathcal{T}_2)$
17 : $m_3 = F^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2-\gamma_2 p_i) + wT_{i,0}^{(k_1,k_2,\ell_1,\ell_2)}$
$-w_i(t_1+t_2+\mathcal{T}_2-\gamma_1 p_i) + p_i W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2-\gamma_2 p_i) - w_i(\mathcal{T}_2-d)$
18 : $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = \max\{m_0,m_1,m_2,m_3\}$
19 : **Si** $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = m_0$ **alors**
20 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2)$
21 : **Sinon**
22 : **Si** $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = m_1$ **alors**
23 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1-\gamma_1 p_i,t_2,\mathcal{T}_2) + w_i$
24 : **Sinon**
25 : **Si** $g^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = m_2$ **alors**
26 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2-\gamma_2 p_i,\mathcal{T}_2) + w_i$
27 : **Sinon**
28 : $W^{(k_1,k_2,\ell_1,\ell_2)}(i,t_1,t_2,\mathcal{T}_2) = W^{(k_1,k_2,\ell_1,\ell_2)}(i-1,t_1,t_2,\mathcal{T}_2-\gamma_2 p_i) + w_i$
29 : **FinSi**
30 : **FinSi**
31 : **FinSi**
32 : **FinPour**
33 : **FinPour**
34 : $wT_{min}^{(k_1,k_2,\ell_1,\ell_2)} = wT^{(k_1,k_2,\ell_1,\ell_2)} + \sum_{j=1}^{2} w_{k_j} \times (\ell_j - d) - \max_{(t_1,t_2,\mathcal{T}_2)} g^{(k_1,k_2,\ell_1,\ell_2)}(n,t_1,t_2,\mathcal{T}_2)$
35 : **FinPour**
36 : **FinPour**
37 : $wT_{min} = \min_{(k_1,k_2,\ell_1,\ell_2)} wT_{min}^{(k_1,k_2,\ell_1,\ell_2)}$
38 : La solution optimale correspond à $(k_1,k_2,\ell_1,\ell_2,t_1,t_2,\mathcal{T}_2)$ dont le coût est wT_{min}.

FIG. 2.12 – Algorithme WTQ2CDD

Dans [141], Kovalyov et Werner ont montré que le problème $Pm|d_i = d|\sum T_i$ n'admet pas aucun algorithme ε-approximation A en temps polynomial avec $\varepsilon < \infty$ sauf si $\mathcal{P} = \mathcal{NP}$. Car sinon, la valeur T^A (T^A est le retard pour une solution donnée par l'algorithme A) peut être déterminée en temps polynomial telle que $T^A - \varepsilon T^* \leq T^* \leq T^A$. En effet, $T^* \geq 0$. Si $T^A = 0$ alors $T^* = 0$, et la réponse pour la question précédente est "oui". Soit $T^* = 0$. Par définition d'un algorithme ε-approximation, nous avons $T^A \leq T^*(1+\varepsilon) = 0$. Par conséquent, $T^* > 0$ doit être respecté, c-à-d. la réponse à la question ci-dessus est "non". Ainsi, la réponse est donnée en temps polynomial, ce qui est contradictoire avec la preuve de la $\mathcal{NP}$-complétude du problème de la minimisation du $C_{\max}$ sur m machines parallèles identiques.

Néanmoins nous pouvons toujours proposer des schémas d'approximation pseudo-polynomial en fonction du paramètre $W = \sum_{i=1}^{n} w_i$. Certes, ces schémas n'ont de sens que si W est borné par une fonction polynomiale en n, comme c'était le cas pour le problème $1|d_i \in D, |D| = l|\sum w_i T_i$ [135].

Conclusion

Nous avons examiné dans ce chapitre, le problème de l'ordonnancement de tâches indépendantes avec des dates de fin souhaitées communes sur une seule machine, sur machines parallèles identiques et sur machines uniformes. L'objectif était la réduction au maximum du retard total pondéré. Nous avons montré pourquoi l'algorithme de Lawler et Moore ne peut pas être généralisé au cas de m machines. Nous avons par la suite proposé une nouvelle formulation d'un programme dynamique. Ensuite, nous avons étendu l'algorithme de ProgDyn pour le cas de machines parallèles identiques et uniformes. Le principe du programme dynamique présenté dans ce chapitre peut être valable pour une généralisation au cas où il y a l dates de fin souhaitées communes.

Il serait intéressant de mettre en oeuvre les ProgDyn et d'évaluer leur efficacité (taille de problème résolu/temps de calcul). Une autre direction de recherche serait d'étudier l'existence d'un schéma d'approximation dans le cas d'une seule machine avec dates de fin souhaitées arbitraires, car la question sur l'existence d'un PTAS pour le problème d'ordonnancement $1|d_i|\sum w_i T_i$ reste ouverte.

Chapitre 3

Avance et retard pondérés avec dates de fin données

Venant de la philosophie 'juste-à-temps' (JàT) pour la gestion de la production, ce type de problème d'ordonnancement a de larges applications industrielles. Ces problèmes consistent à déterminer une solution qui minimise la somme pondérée des pénalités d'avance et de retard des travaux. Il s'agit donc de calculer un ordonnancement où chaque travail se termine le plus près possible de sa date de fin souhaitée. Dans le cas d'une seule machine avec une date de fin souhaitée commune donnée, le problème a été montré $\mathcal{NP}$-difficile. Dans ce chapitre, on ordonnance n travaux sur une seule machine ou sur m machines parallèles. Pour le cas de durées opératoires identiques, l'ordonnancement des travaux sur m machines identiques ou uniformes est montré polynomial, même s'il s'agit de ℓ dates de fin souhaitées communes données ($\ell < n$). Pour le cas général, nous montrons que le problème d'ordonnancement des travaux avec durées opératoires quelconques et une date de fin souhaitée commune "non-restrictive" sur m machines parallèles identiques accepte un schéma d'approximation en temps polynomial (PTAS).

3.1 Introduction

Parmi les problèmes d'ordonnancement rencontrés dans les ateliers de production, le modèle "Juste-à-temps" (JàT) tient une place importante. Il s'agit en particulier, d'une politique de production qui permet de limiter les stocks et d'éviter le gaspillage. La quantité de produits fabriqués correspond au carnet de commandes. Le producteur se contente de produire la quantité strictement nécessaire, au moment voulu, pour satisfaire la demande de chaque client. C'est la date à laquelle le client demande à être livré qui précise les délais de la production. On parle alors de date de fin souhaitée. Il n'est alors pas souhaitable de livrer les clients en retard à cause des coûts de pénalités liés essentiellement au mécontentement de ce dernier. De plus, il arrive qu'on ne souhaite pas produire en avance pour diverses raisons (limitation d'espace de stockage, coûts de stockage, risques de dégradation du produit fini, etc.). Il est ainsi préférable de produire JàT, ce qui consiste à déterminer un ordonnancement tel qu'un travail se termine le plus près possible de sa date de fin souhaitée ([20, 36, 126]).

Dans ce chapitre, on s'intéresse à l'ordonnancement de travaux indépendants sur une ou plusieurs machines en parallèle où l'objectif est de minimiser la somme pondérée des coûts d'avance/retard. Tous les travaux devraient être accomplis avant une date de fin souhaitée commune. Cette date due correspond à la situation où des articles devraient être fournis en même temps à une chaîne d'assemblage, par exemple, pour éviter le stockage ou le retardement de livraisons.

Selon la date de fin souhaitée, on définit deux versions du problème : avec date due commune non-restrictive et avec date due commune restrictive [145, 126]. En effet, si l'indisponibilité d'une ressource avant la date zéro doit être prise en compte, le problème est dit restrictif. Dans le cas contraire, il est dit non-restrictif.

Selon la notation classique [80], les problèmes considérés dans ce chapitre sont :

1. Durées opératoires identiques ($p_i = p, \forall i, i = 1, \ldots, n$)
   - $1|p_i = p, d_i = d|\sum(\alpha_i E_i + \beta_i T_i)$ (cf. section 3.3.1)
   - $1|p_i = p, d_i \in D, |D| = \ell|\sum(\alpha_i E_i + \beta_i T_i)$ (cf.section 3.3.3)
   - $Qm|p_i = p, d_i \in D, |D| = \ell|\sum(\alpha_i E_i + \beta_i T_i)$ (cf. section 3.3.4)
2. Durées opératoires quelconques
   - $1|d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ (cf. section 3.4.1)
   - $Pm|d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ (cf.section 3.4.2)

où le nombre de machines $m \geq 1$ est fixé ; α_i (resp. β_i) est la pénalité par unité de temps d'avance E_i (resp. de retard T_i) du travail J_i.

Parmi les premiers résultats adressés pour la résolution des problèmes d'ordonnancement JàT, on peut citer Kanet [129]. L'auteur a montré que si la somme des durées opératoires des travaux est inférieure ou égale à la date de fin souhaitée, le problème avec pénalités unitaires ($\alpha_i = \beta_i = 1$), noté $1|d_i = d, non - restrictive| \sum(E_i + T_i)$, est polynomial. Ainsi, une solution optimale est obtenue comme suit. En partant de la date d, les travaux rangés selon l'ordre SPT d'indice impair sont ordonnancés en avance, et d'indice pair sont ordonnancés en retard. Ainsi, la date de fin du travail de plus petite durée opératoire est d, et il est dit "travail juste-à-temps". Quelques années plus tard, le même problème est étudié par Bagchi et al. [16]. Les auteurs proposent un autre algorithme polynomial où la solution optimale retournée peut être préférée par le chef d'atelier puisqu'elle permet de minimiser la somme des durées opératoires des travaux ordonnancés en avance.

Selon les valeurs des pénalités α_i et β_i, si $(\alpha_i, \beta_i) \in \{(0, w_i), (w_i, w_i)\}$ (avec w_i la pénalité quelconque d'un travail J_i), le problème à une machine est montré $\mathcal{NP}$-difficile au sens ordinaire [226, 87, 86, 98]. Des algorithmes pseudo-polynomiaux de programmation dynamique sont proposés dans [36], [86], [87] et [98]. Plusieurs cas polynomiaux ont été identifiés et sont rappelés dans [126, 209].

Sachant que le cas avec durées opératoires et pénalités d'avance/retard quelconques est $\mathcal{NP}$-difficile, les chercheurs ont eu recours aux heuristiques et aux métaheuristiques. Comme métaheuristiques appliquées au problème considéré, nous pouvons citer : algorithme génétique [152], recherche tabou [89] et [125], recuit simulé [67]. D'autres chercheurs ont proposé des méthodes approchées hybrides [10] et [93]. Pour le cas restrictif, la plupart des articles ne considèrent pas le cas général avec pénalités d'avance et retard quelconques à l'exception des résultats de Lee et Kim [152] et James [125]. Pour tous ces résultats, notons tout de même que les auteurs considèrent toujours que le premier travail est exécuté à la date zéro, ce qui est très contraignant puisque dans une solution optimale on peut avoir un temps mort au début de l'ordonnancement des travaux.

Comme heuristique, on peut citer les travaux de Biskup et Feldmann [26], Feldmann et Biskup [67] et Nearchou [167]. Dans [26], les auteurs proposent deux heuristiques pour résoudre le problème à une machine avec pénalités d'avance et retard quelconques dans lesquelles un générateur d'instances est développé. 280 instances

"benchmark" ont été choisies comme base de tests pour étudier et comparer les performances des méthodes de résolution. Pour ces 280 instances, le nombre de travaux varie entre 10 et 1000 avec une date de fin souhaitée commune calculée selon un certain pourcentage fixé entre 20% et 80% de la somme des durées opératoires des travaux [1]. Pour ces instances, les auteurs fournissent également les meilleurs résultats connus (meilleures bornes supérieures [26] et [67]).

Dans [167], les auteurs proposent une heuristique basée sur la méthode dite "évolution différentielle" introduite par [204] pour résoudre les problèmes d'optimisation avec variables continues. Plus précisément, cette heuristique permet d'obtenir des solutions meilleures (dans 60% des instances) ou équivalentes (20% de cas). Ying [224] propose un algorithme de recherche par faisceau filtré et recouvrement des erreurs (RBS - Recorvering Beam Search en anglais). Cet algorithme introduit initialement par Della Croce et T'kindt en 2002 [46], est une méthode prometteuse pour la résolution des problèmes d'optimisation combinatoire [45],[63], [74] et [211]. Inspiré de la méthode de recherche par faisceau filtré (BS - Beam Search en anglais), RBS n'est autre qu'une procédure par séparation et évaluation tronquée, i.e. à chaque niveau de l'arbre de recherche seuls les noeuds les plus prometteurs sont retenus. La taille des noeuds retenus est fixée par un paramètre qui est dit "largeur de faisceau". Étant donné qu'on ne stocke qu'un nombre limité de noeuds à chaque niveau de l'arbre de recherche, le temps d'exécution de RBS en fonction de la taille de l'entrée est polynomial [211]. Contrairement à BS où aucune procédure ne permet de récupérer un noeud déjà coupé (surtout s'il s'agit de la branche qui nous permet d'atteindre une solution optimale), la RBS propose une phase de récupération de noeuds qui correspondent aux meilleures solutions partielles. Ces noeuds sélectionnés par le faisceau en nombre limité, remplacent ceux qui se situent à la même profondeur de l'arbre de recherche.

Comme méthodes exactes, nous pouvons citer les travaux de Sourd [202] dans lesquels une méthode exacte performante de type procédure par séparation et évaluation est proposée. Cette méthode permet de résoudre en un temps n'excédant pas les 15 minutes en moyenne (25 minutes dans le pire des cas) des instances de grande taille (1000 travaux). Notons que cette méthode a été proposée initialement par l'auteur pour résoudre de façon optimale le cas général avec dates d'arrivée des travaux,

[1] Générateur de tests disponible à l'adresse http ://people.brunel.ac.uk/~mastjjb/jeb/orlib/schinfo.html ou sur le web site de l'Imperial College Management School Web site "http ://www.ms.ic.ac.uk/jeb/orlib/schinfo.html"

$1|r_i, d_i|\sum((\alpha_i E_i + \beta_i T_i)$ où les résultats expérimentaux montrent qu'on peut résoudre des instances allant jusqu'à 50 travaux. Cependant, à notre connaissance, le meilleur algorithme connu pour résoudre le problème $1|r_i, d_i|\sum((\alpha_i E_i + \beta_i T_i)$ a été proposé par Tanaka et Fujikuma [207]. Cet algorithme permet de résoudre des instances allant jusqu'à 200 travaux.

Contrairement au cas d'une seule machine, moins de résultats ont été publiés avec m machines parallèles. Cependant, afin de minimiser la somme des avances et des retards des travaux avec durées opératoires unitaires et une date de fin souhaitée commune sur m machines parallèles identiques, Mosheiov and Yovel [165] montrent qu'une solution optimale peut être obtenue en $O(n^3)$. Dans [184, 222], les auteurs proposent une méthode de recherche par voisinage pour le cas de durées opératoires quelconques et date de fin souhaitée commune restrictive. Pour ce dernier problème, des bornes inférieures efficaces ont été développées par Kedad-Sidhoum et al. [132]. Ces bornes inférieures peuvent être adaptées au cas de dates de fin souhaitées générales quelconques.

D'autres travaux ont été consacrés dans le but de définir des schémas d'approximation. Avec des pénalités symétriques ($\alpha_i = \beta_i = w_i$) et une date de fin souhaitée commune non-restrictive, Hall et Posner [87] développent un FPTAS sous hypothèse que la valeur maximale des pénalités est bornée par une fonction en n. Cependant, sous cette hypothèse, le problème a été montré polynomial [49, 127]. Quelques années plus tard, Kovalyov et Kubiak [140] montrent en effet que le problème symétrique, $1|d_i = d, non-restrictive|\sum((w_i E_i + w_i T_i)$, admet un FPTAS. Pour le cas restrictif, avec des pénalités unitaires ($\alpha_i = \beta_i = 1$), Hoogeveen et al. [103] proposent une heuristique avec garantie de performance $\frac{4}{3}$.

3.2 Propriétés pour le cas d'une date de fin souhaitée commune

Propriété 3.2.1 *[20] Dans une solution optimale, il n'existe pas de temps mort entre deux travaux adjacents.*

Propriété 3.2.2 *[195] Une séquence optimale respecte le "V-shape" autour de la date fin souhaitée commune, i.e. les travaux en avance (terminés avant ou à la date d) sont ordonnancés selon l'ordre non-croissant des p_i/α_i, et les travaux en retard sont ordonnancés selon l'ordre non-décroissant des p_i/β_i.*

Problème	Méthode & Complexité	Reference
$1\|p_i = p\|\sum(E_i + T_i)$	Polynomial $O(n \log n)$	[72]
$1\|\|\sum(E_i + T_i)$	$\mathcal{N}P$-difficile	[72]
$1\|d_i = d, non-restrictive\|\sum(E_i + T_i)$	Polynomial $O(n \log n)$	[129]
$1\|d_i = d, restrictive\|\sum(E_i + T_i)$	$\mathcal{N}P$-difficile	[86, 98]
	PD $O(n\sum p_i)$ et $O(n^2 d)$	[86, 98]
	4/3 approximation	[103]
$1\|seq\|\sum w_i(E_i + T_i)$	Polynomial $O(n \log n)$	[72]
$1\|d_i = d, p_i = w_i\|\sum w_i(E_i + T_i)$	Polynomial $O(n)$	[87]
$1\|d_i = d, p_i = 1\|\sum w_i(E_i + T_i)$	Polynomial $O(n \log n)$	[87]
$1\|d_i = d\|\sum w_i(E_i + T_i)$	Polynomial si $w_{max} = Poly(n)$	[127]
$1\|d_i = d, non-restrictive\|\sum w_i(E_i + T_i)$	$\mathcal{N}P$-difficile	[87]
	PD $O(n\sum p_i)$ et $O(n\sum w_i)$	[87, 127]
	FPTAS	[140]
$1\|d_i \in [d, d + p_i]\|\sum w_i(E_i + T_i)$	$\mathcal{N}P$-difficile	[87]
	PD $O(n^2\sum p_i)$	[101]
$1\|seq\|\sum(\alpha_i E_i + \beta_i T_i)$	Polynomial $O(n^2)$	[47, 206]
$1\|p_i = 1, d_i\|\sum(\alpha_i E_i + \beta_i T_i)$	Polynomial $O(n^4)$	[139]
$1\|p_i = p, d_i = d, non-restrictive\|\sum(\alpha_i E_i + \beta_i T_i)$	Polynomial $O(n^3)$	[165, 109, 110]
$1\|p_i = p, d_i \in D, \|D\| = k\|\sum(\alpha_i E_i + \beta_i T_i)$	Polynomial	[109, 110]
$1\|d_i = d, non-restrictive\|\sum(\alpha_i E_i + \beta_i T_i)$	PSE n=1000	[202]
$Q\|d_i = d, non-restrictive\|\sum(\alpha E_i + \beta T_i)$	Polynomial $O(n \log n)$	[61]
$R\|d_i = d, non-restrictive\|\sum(E_i + T_i)$	Polynomial $O(n^3)$	[142, 9]
$P\|d_i = d, nonrestrictive\|\sum w_i(E_i + T_i)$	$\mathcal{N}P$-difficile au sens fort	[216]
$Pm\|d_i = d, p_i = w_i\|\sum w_i(E_i + T_i)$	$\mathcal{N}P$-difficile au sens ordinaire	[205]
	PD $O(mnd^m P^{m-1})$ [205]	
$Qm\|d_i = d, non-restrictive\|\sum(\alpha E_i + \beta T_i)$	Polynomial $O(n \log n)$	

TAB. 3.1 – État de l'art sur la minimisation des avances et des retards

Propriété 3.2.3 *[98] Dans le cas restrictif, pour une solution optimale, il existe un travail qui se termine à la date d si aucun travail ne commence à la date zéro.*

Propriété 3.2.4 *Dans le cas non-restrictif, il existe une solution optimale où exactement un travail se termine à l'heure.*

Preuve. Soit S^* une solution optimale. Nous supposons qu'il n'existe pas dans S^* un travail qui se termine à la date d.

Soit J_k le travail pour lequel $S_k \leq< d$ et $C_k > d$. J_k est dit "*splitting job*". Par conséquent, nous définissons ΔW comme suit : $\Delta W = \sum_{C_i>d}(\beta_i) - \sum_{C_i<d}(\alpha_i)$. Selon la valeur de ΔW, nous avons :

- Si ($\Delta W < 0$), nous pouvons alors construire une solution $S^{'}$ à partir de S^* en décalant tous les travaux d'une unité de temps vers la droite. Nous obtenons donc $f(S^{'}) < f(S^*)$ ce qui est contradictoire avec S^* une solution optimale.
- Si ($\Delta W > 0$), nous pouvons alors construire une solution $S^{'}$ à partir de S^* en décalant tous les travaux d'une unité de temps vers la gauche. Nous obtenons donc $f(S^{'}) < f(S^*)$ ce qui est contradictoire avec S^* une solution optimale.
- Sinon ($\Delta W = 0$), nous pouvons alors construire une solution $S^{'}$ à partir de S^* en décalant tous les travaux de $(C_k - d)$ unités de temps vers la droite, i.e. J_k est ordonnancé à l'heure dans $S^{'}$. Il est donc facile de montrer que $f(S^{'}) = f(S^*)$. $S^{'}$ est aussi une solution optimale. □

3.3 Durées opératoires identiques

Dans cette section, les problèmes à une seule machine et à m machines parallèles (identiques et uniformes) sont étudiés pour la minimisation de la somme pondérée des avances et des retards. n travaux indépendants J_i $(i = 1, \ldots, n)$ avec des durées opératoires identiques devraient être exécutés sans préemption au plus près possible de leur date de fin souhaitée d_j respective ($d_j \in D$ et $|D| = \ell$). Nous supposons que tous les travaux sont disponibles à l'instant zéro.

Récemment, un cas particulier a été traité par Mosheiov et Yovel [165]. Ils montrent que le problème d'ordonnancement sur des machines parallèles identiques avec durées opératoires unitaires ($p_i = 1$) et une date de fin souhaitée commune ($\ell = 1$) est résolu en temps polynomial. Contrairement au cas $p_i = 1$, une petite difficulté s'ajoute au cas $p_i = p$ (p quelconque). En effet, nous ne pouvons pas assurer l'existence d'une solution

optimale avec un travail juste-à-temps (se termine à la date d). Néanmoins, la méthode de résolution proposée par Mosheiov et Yovel peut être utilisée pour résoudre le cas de durées opératoires identiques pour la version non-restrictive du problème considéré.

Toutefois, pour la version restrictive avec des durées opératoires identiques ($p_i = p$), nous montrons qu'une solution optimale peut être obtenue en temps polynomial. Par la suite, nous montrons que le cas avec ℓ dates de fin souhaitées communes données (une famille de dates de fin) est également polynomial.

L'idée globale de notre démarche peut être résumée comme suit. Étant donné que tous les travaux ont la même durée opératoire, l'ordonnancement peut être réalisé en deux étapes :

1. Déterminer toutes les dates de fin d'exécution possibles sans considérer l'affectation de ces dates de fin aux travaux. Nous pouvons montrer qu'il est possible de construire en temps polynomial cet ensemble de dates de fin d'exécution. En effet, concernant les positions des travaux en avance, nous avons au pire n dates de fin d'exécution possibles. De façon symétrique, on peut montrer également que la cardinalité de l'ensemble des dates de fin possibles des travaux ordonnancés en retard (les positions des travaux en retard) est bornée par n.
2. Attribuer à chaque travail la date de fin d'exécution calculée la moins coûteuse. Il s'agit donc d'un problème d'affectation.

Rappelons que le premier algorithme proposant de résoudre le problème d'affectation, dit "méthode hongroise", a été introduit par Kuhn en 1955 [143]. Pour une matrice d'affectation de $n \times n$, la complexité de l'algorithme est en $O(n^3)$. Plusieurs algorithmes proposant une amélioration de la complexité de la recherche d'une meilleure affectation ont été présentés dans [42], [94] et [131]. Toutefois, notre principale motivation n'est pas de proposer un nouvel algorithme pour résoudre les problèmes d'affectation mais de prouver que les problèmes étudiés sont équivalents au problème d'affectation. Par la suite, nous considérons une version étendue du problème d'affectation des x éléments aux y différents items avec $y \geq x$. Ce problème peut être résolu en $O(x^2y)$.

3.3.1 Une seule machine avec une date de fin souhaitée commune

Selon la littérature, pour une date de fin souhaitée commune donnée, deux versions du problème doivent être considérées : la version *non-restrictive* et la version *restrictive*.

Kanet [129] considère le cas non-restrictif lorsque la date d est supérieure ou égale à la somme des durées opératoires ($d \geq \sum_{i=1}^{n} p_i$).

Avant de rentrer dans le vif du sujet, rappelons quelques propriétés bien connues qui seront utiles par la suite. De plus, pour le cas non-restrictif, nous montrons qu'il existe toujours une solution optimale avec un travail juste-à-temps.

1. Dans une solution optimale, il n'existe pas de temps mort intermédiaire entre deux travaux adjacents [20].
2. L'ordonnancement optimal forme un 'V-shape' autour de la date de fin souhaitée commune. Cela signifie que les travaux terminés avant ou à la date d sont ordonnancés selon l'ordre non-croissant du ratio p_i/α_i (WLPT) et les travaux achevés après d sont ordonnancés selon l'ordre non-décroissant du ratio p_i/β_i (WSPT) [195].
3. Dans le cas restrictif, pour une solution optimale, il existe un travail qui se termine à la date d si aucun travail ne commence à la date zéro [98].
4. Dans le cas non-restrictif, il existe une solution optimale avec un travail juste-à-temps [109, 110, 106].

Soit J_{sp} un *splitting job* avec $S_{sp} \leq d < C_{sp}$. Soit k le nombre maximum de travaux en avance. Si d est un multiple de p, alors $k = \left\lfloor \frac{d}{p} \right\rfloor - 1$; sinon $k = \left\lfloor \frac{d}{p} \right\rfloor$. Nous pouvons donc écrire $k = \left\lfloor \frac{d-1}{p} \right\rfloor$.

Il est clair que le travail ordonnancé à la $(k+1)$-ième position, noté par $J_{[k+1]}$ est le *splitting job*, noté J_{sp}.

Soit $\delta_1 = d - kp$, $\delta_2 = p - \delta_1$ les distances entre S_{sp}, C_{sp} et d (voir Figure 3.1).

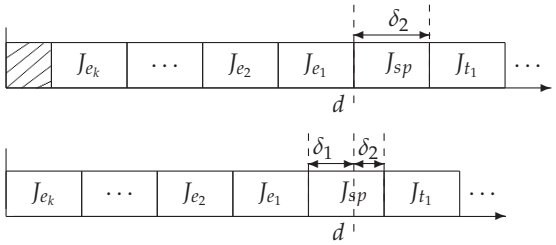

FIG. 3.1 – Positions des travaux sur une machine avec une date due donnée

Les positions des travaux sont définies comme suit (voir Figure 3.1) :

– concernant les travaux en avance : le dernier travail exécuté juste avant J_{sp} est affecté à la première position en avance noté J_{e_1}, ce travail peut être juste-à-temps ;

le travail exécuté juste avant J_{e_1} occupe la seconde position en avance, il est noté J_{e_2} ; et ainsi de suite,

– de façon symétrique, concernant les travaux en retard : le premier travail exécuté juste après J_{sp} est à la première position en retard, il est noté par J_{t_1} ; le second travail en retard est exécuté juste après J_{t_1}, il est noté par J_{t_2} ; et ainsi de suite.

Ainsi, le coût d'affectation d'un travail à la i-ème position en avance est donné par $\alpha_{[k-i+1]}((i-1)p+\delta_1)$, ce qui correspond à la contribution de ce travail au coût total des pénalités d'avances. De même, le coût d'affectation d'un travail à la i-ème position en retard est donné par $\beta_{[k+1+i]}(ip+\delta_2)$.

Par conséquent, les coûts d'affectation définissent la matrice $Q_{((k+n)\times n)}$ dans laquelle chaque élément est noté par $(q_{i,j})_{1\leq i\leq 2u+1;1\leq j\leq n}$ où la j-ème colonne correspond au travail J_j ; la première ligne correspond au travail J_{sp} ; la $(2i)$-ème ligne correspond au travail J_{e_i} $(1\leq i\leq k)$; la $(2i+1)$-ème ligne correspond au travail J_{t_i} $(1\leq i\leq k)$; et la i-ème ligne correspond au travail $J_{t_{i-k-1}}$ $(2k+2\leq i\leq n+k)$ (voir Figure 3.2).

Par la suite, un élément $(q_{i,j})$ est donné par :

$$\begin{cases} q_{1,j}=\delta_2\alpha_j & (j=1,\ldots,n)\\ q_{2i,j}=(\delta_1+(i-1)p)\alpha_j & (i=1,\ldots,k;j=1,\ldots,n)\\ q_{2i+1,j}=(\delta_2+ip)\beta_j & (i=1,\ldots,k;j=1,\ldots,n)\\ q_{i,j}=(\delta_2+(i-k-1)p)\beta_j & (i=2k+2,\ldots,n+k;j=1,\ldots,n)\end{cases}$$

D'un autre point de vue, la matrice $Q_{((k+n)\times n)}$ peut être définie comme suit :

$$\begin{aligned} Q^T = [&\delta_2 w_i^{(2)^T} \quad \delta_1 w_i^{(1)^T} \quad (\delta_2+p)w_i^{(2)^T} \quad (\delta_1+p)w_i^{(1)^T}\\ &(\delta_2+2p)w_i^{(2)^T} \quad \ldots \quad (\delta_1+(k-1)p)w_i^{(1)^T}\\ &(\delta_2+kp)w_i^{(2)^T} \quad (\delta_2+(k+1)p)w_i^{(2)^T}\\ &(\delta_2+(k+2)p)w_i^{(2)^T} \quad \ldots \quad (\delta_2+(u-1)p)w_i^{(2)^T}]\end{aligned}$$

Un ensemble $S=(q_{k_1,1},q_{k_2,2},\ldots,q_{k_n,n})$ de n éléments de Q est appelé une *solution faisable* avec $Z=f(S)=\sum_{r=1}^{n}q_{k_r,r}$ si et seulement si, il n'y a pas deux éléments issus de la même ligne (évidemment, deux éléments de S ne peuvent pas appartenir à la même colonne).

Par conséquent, pour déterminer une solution optimale S^*, nous devons trouver un ensemble faisable de n éléments de Q qui minimise Z. Il s'agit donc d'un problème

		Travaux			
		J_1	J_2	$\dots$	J_n
Positions	J_{sp}	$\delta_2\beta_1$	$\delta_2\beta_2$	$\dots$	$\delta_2\beta_n$
	J_{e_1}	$\delta_1\alpha_1$	$\delta_1\alpha_2$	$\dots$	$\delta_1\alpha_n$
	J_{e_2}	$(\delta_1+p)\alpha_1$	$(\delta_1+p)\alpha_2$	$\dots$	$(\delta_1+p)\alpha_n$
	J_{t_2}	$(\delta_2+2p)\beta_1$	$(\delta_2+2p)\beta_2$	$\dots$	$(\delta_2+2p)\beta_n$
	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
	J_{e_k}	$(\delta_1+(k-1)p)\alpha_1$	$(\delta_1+(k-1)p)\alpha_2$	$\dots$	$(\delta_1+(k-1)p)\alpha_n$
	J_{t_k}	$(\delta_2+kp)\beta_1$	$(\delta_2+kp)\beta_2$	$\dots$	$(\delta_2+kp)\beta_n$
	$J_{t_{k+1}}$	$(\delta_2+(k+1)p)\beta_1$	$(\delta_2+(k+1)p)\beta_2$	$\dots$	$(\delta_2+(k+1)p)\beta_n$
	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
	$J_{t_{n-1}}$	$(\delta_2+(n-1)p)\beta_1$	$(\delta_2+(n-1)p)\beta_2$	$\dots$	$(\delta_2+(n-1)p)\beta_n$

FIG. 3.2 – Matrice des coûts d'affectation des travaux

d'affectation [143].

Remarque : en utilisant la même approche, nous pouvons déterminer la matrice des coûts d'affectation des travaux pour le cas de date de fin souhaitée non-restrictive et montrer ainsi que le problème est équivalent au problème d'affectation. Bien entendu, dans ce cas, les lignes correspondantes aux k-ièmes positions en avance pout tout $k > n$ sont ignorées.

Théorème 3.3.1 *Une solution optimale pour le problème d'ordonnancement* $1|p = p_i, d_i = d|\sum(\alpha_i E_i + \beta_i T_i)$ *avec date de fin souhaitée commune non-restrictive/restrictive peut être calculée en* $O(n^3)$.

Exemple 3.3.1 *5 travaux à ordonnancer sur une seule machine. Les durées opératoires, les dates de fin souhaitées, les pénalités d'avance et de retard sont données dans la Table 3.2.*

TAB. 3.2 – Durées opératoires, dates de fin souhaitées, et pénalités d'avance/retard

	J_1	J_2	J_3	J_4	J_5
p_i	3	3	3	3	3
d_i	7	7	7	7	7
α_i	7	5	4	3	5
β_i	5	2	7	4	5

Selon la date de début du travail $J_{[3]}$, nous avons alors deux valeurs possibles pour le couple (δ_1, δ_2) : $(0, 3)$ (Figure 3.3.Cas(a)) et $(1, 2)$ (Figure 3.3.Cas(b)).

Pour chacun des deux cas, nous pouvons donc écrire la matrice Q (voir Figure 3.4).

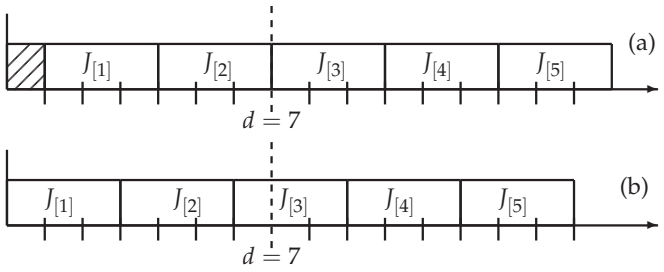

FIG. 3.3 – Exemple avec 5 travaux

		Travaux				
		J_1	J_2	J_3	J_4	J_5
Positions	J_{sp}	$\delta_2\beta_1$	$\delta_2\beta_2$	$\delta_2\beta_3$	$\delta_2\beta_4$	$\delta_2\beta_5$
	J_{e_1}	$\delta_1\alpha_1$	$\delta_1\alpha_2$	$\delta_1\alpha_3$	$\delta_1\alpha_4$	$\delta_1\alpha_5$
	J_{t_1}	$(\delta_2+p)\beta_1$	$(\delta_2+p)\beta_2$	$(\delta_2+p)\beta_3$	$(\delta_2+p)\beta_4$	$(\delta_2+p)\beta_5$
	J_{e_2}	$(\delta_1+p)\alpha_1$	$(\delta_1+p)\alpha_2$	$(\delta_1+p)\alpha_3$	$(\delta_1+p)\alpha_4$	$(\delta_1+p)\alpha_5$
	J_{t_2}	$(\delta_2+2p)\beta_1$	$(\delta_2+2p)\beta_2$	$(\delta_2+2p)\beta_3$	$(\delta_2+2p)\beta_4$	$(\delta_2+2p)\beta_5$
	J_{t_3}	$(\delta_2+3p)\beta_1$	$(\delta_2+3p)\beta_2$	$(\delta_2+3p)\beta_3$	$(\delta_2+3p)\beta_4$	$(\delta_2+3p)\beta_5$
	J_{t_4}	$(\delta_2+4p)\beta_1$	$(\delta_2+4p)\beta_2$	$(\delta_2+4p)\beta_3$	$(\delta_2+4p)\beta_4$	$(\delta_2+4p)\beta_5$

FIG. 3.4 – Matrice d'affectation avec 5 travaux

Dans le cas (a) avec $(\delta_1, \delta_2) = (0, 3)$, la matrice correspondante et les éléments sélectionnés pour minimiser les coûts d'affectation sont représentés dans la Figure 3.5.

		Travaux				
		J_1	J_2	J_3	J_4	J_5
Positions	J_{sp}	15	6	21	12	**15**
	J_{e_1}	**0**	0	0	0	0
	J_{t_1}	30	12	42	**24**	30
	J_{e_2}	21	15	**12**	9	15
	J_{t_2}	45	**18**	63	36	45
	J_{t_3}	60	24	84	48	60
	J_{t_4}	75	30	105	60	75

FIG. 3.5 – Matrice d'affectation avec 5 travaux et $(\delta_1, \delta_2) = (0, 3)$

La solution est définie par la séquence (3,1,5,4,2) et sa valeur optimale pour le cas (a) est **69** (voir la Figure 3.6).

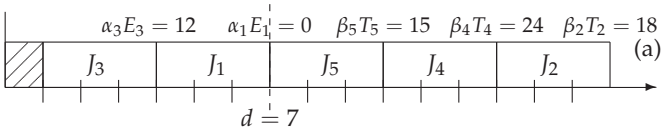

FIG. 3.6 – Solution optimale correspondant au Cas (a)

Dans le cas (b) avec $(\delta_1, \delta_2) = (1, 2)$, la matrice correspondante et les éléments sélectionnés pour minimiser le coût d'affectation sont représentés dans la figure 3.7.

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_{sp}	10	4	14	8	**10**
J_{e_1}	7	5	**4**	3	5
J_{t_1}	**25**	10	35	20	25
J_{e_2}	28	20	16	**12**	20
J_{t_2}	40	**16**	56	32	40
J_{t_3}	55	22	77	44	55
J_{t_4}	70	28	98	56	70

FIG. 3.7 – Matrice d'affectation avec 5 travaux et $(\delta_1, \delta_2) = (1, 2)$

La solution optimale correspond donc à la séquence (4, 3, 5, 1, 2) et est obtenue selon le cas (b) pour une valeur de **67** (voir Figure 3.8).

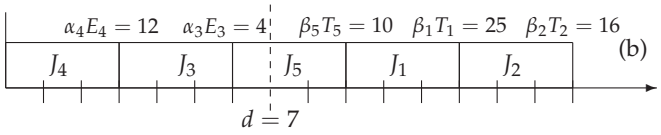

FIG. 3.8 – Solution optimale selon le cas (b)

Par conséquent, la solution optimale correspond à la meilleure des deux solutions précédentes. La valeur optimale est alors **67** définie par la séquence optimale (4, 3, 5, 1, 2) ordonnancée selon le cas (b).

Nous allons maintenant étudier le cas de ℓ dates de fin souhaitées données où les durées opératoires sont identiques. Afin d'illustrer notre approche, nous étudions d'abord le cas de deux dates de fin souhaitées distinctes ($\ell = 2$) notées par D_1 et D_2. Sans perte de généralité, nous supposons que $D_1 \leq D_2$. Dans la suite, D_0 correspond à la date zéro ($D_0 = 0$) et pour une date D_j, un travail est dit *splitting job* si $S_i \leq D_j < C_i$ même si $d_i \neq D_j$.

3.3.2 Une seule machine avec deux dates de fin souhaitées

En général, il est possible d'avoir une solution optimale avec un temps mort entre D_1 et D_2. En particulier, si la distance entre D_1 et D_2 est assez grande, il est facile de montrer que le problème peut être résolu en temps polynomial. Autrement dit, si $D_2 - D_1 \geq T_{D_1} + E_{D_2}$ où T_{D_1} (resp. E_{D_2}) représente la somme des durées opératoires des travaux avec date de fin souhaitée D_1 (resp. D_2) ordonnancés en retard (resp. en avance). T_{D_1} et E_{D_2} sont obtenues en résolvant indépendamment chaque sous-problème correspondant à l'ordonnancement du sous-ensemble des travaux pour une date de fin

$D_{\ell,\ell=1,2}$ en $O(n^3)$ (voir section 3.3.1).

Considérons maintenant le cas général. Sans perte de généralité, notons par J_{sp_1} et J_{sp_2} les *s*plitting jobs. Par définition, nous avons $S_{sp_1} \leq D_1 < C_{sp_1}$ et $S_{sp_2} \leq D_2 < C_{sp_2}$.

Propriété 3.3.2 *Il existe une solution optimale qui satisfait les conditions suivantes :*

1. *au maximum, il existe un seul temps mort dans l'intervalle du temps* $[D_0, D_1[$,
2. *au maximum, il existe un seul temps mort dans l'intervalle du temps* $[D_1, D_2[$, *et pas de temps mort entre les travaux ordonnancés après la date* D_2,
3. *s'il n'y a pas de temps mort dans l'intervalle* $[D_1, D_2[$, *il existe alors un travail* J_i *tel que* $S_i \in \{D_0, D_1, D_2\}$,
4. *s'il y a un temps mort dans l'intervalle* $[D_1, D_2[$, *il existe alors deux travaux* J_{i_1} *et* J_{i_2} *tels que* $S_{i_1} \in \{0, D_1\}$ *et* $S_{i_2} = D_2$.

Preuve :

Pour montrer la propriété 3.3.2.(1 et 2), nous considérons une solution S avec des temps morts intermédiaires ne respectant pas les deux premières propriétés (voir la Figure 3.9). Nous pouvons toujours améliorer cette solution en déplaçant une sous-séquence de travaux ordonnancée entre deux temps morts intermédiaires de façon à les rapprocher de leur date de fin souhaitée respective. Ainsi, le coût d'avance et retard total sera réduit. Il est à noter que les tâches placées entre D_1 et D_2 doivent être rangées selon l'ordre de leur date de fin souhaitée.

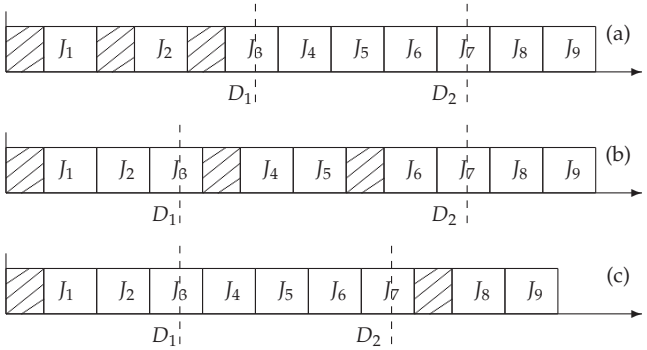

FIG. 3.9 – Exemple avec temps morts dans $[D_0, D_1]$ et $]D_1, D_2]$ et après D_2.

Pour montrer la propriété 3.3.2.(3), nous supposons qu'il existe une solution S dans laquelle il n'y a pas de temps morts entre D_1 et D_2 et il n'existe aucun travail J_i tel que $S_i \in \{D_0, D_1, D_2\}$ (Voir Figure 3.10), i.e. il existe un temps mort à la date zéro.

Soient $\Delta_1 = min(S_{[1]}; D_1 - S_{sp_1}; D_2 - S_{sp_2})$ et $\Delta_2 = min(C_{sp_1} - D_1; C_{sp_2} - D_2)$; soit A_1 (resp. A_2) l'ensemble des travaux ordonnancés en avance (resp. en retard) selon S. En fait, Δ_1 (resp. Δ_2) représente la quantité de temps minimale pour laquelle le déplacement vers la gauche (resp. vers la droite) de la séquence des travaux permet d'avoir $S_i \in \{D_0, D_1, D_2\}$. Soit α (resp. β) la somme des pondérations d'avance (resp. retard) des travaux de A_1 (resp. A_2). Par conséquent, nous avons $\alpha = \sum_{i \in A_1} \alpha_i$ et $\beta = \sum_{i \in A_2} \beta_i$.

Nous remarquons que :

- Si la séquence S est déplacée vers la gauche de Δ_1 unités de temps, la variation du coût global est définie par $\Delta_1(\sum_{i \in A_1} \alpha_i - \sum_{i \in A_2} \beta_i) = \Delta_1(\alpha - \beta)$.
- Si la séquence S est déplacée vers la droite de Δ_2 unités de temps, la variation du coût global est définie par $\Delta_2(\sum_{i \in A_2} \beta_i - \sum_{i \in A_1} \alpha_i) = \Delta_2(\beta - \alpha)$.

Ainsi, si $\alpha - \beta \geq 0$, nous pouvons alors améliorer cette solution en déplaçant la séquence S vers la gauche, sinon, la séquence doit être déplacée vers la droite. Par conséquent, il existe un travail J_i tel que $S_i \in \{0, D_1, D_2\}$.

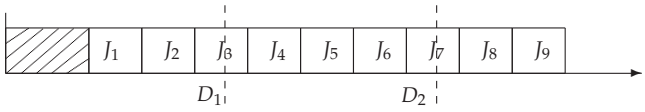

FIG. 3.10 – Exemple sans temps mort dans $]D_1, D_2]$

Pour montrer la propriété 3.3.2.(4), nous considérons maintenant une solution S pour laquelle, nous avons un temps mort entre D_1 et D_2. Ainsi, il existe deux sous-séquences de travaux : l'une définie par les travaux ordonnancés autour de D_1 notée par σ_1 et l'autre définie par les travaux ordonnancés autour de D_2 notée par σ_2. Par définition, on note l'absence de temps mort dans σ_1 et dans σ_2 (voir la Figure 3.11).

Notons par J_{k_1} le dernier travail de la sous-séquence σ_1 et par J_{k_2} le premier travail de la sous-séquence σ_2 (selon la Figure 3.11, $J_{k_1} = J_5$ et $J_{k_2} = J_6$).

S'il n'y a pas de temps mort au début de l'ordonnancement (Figure 3.11.Cas (a)), alors $S_{i_1} = 0$.

Considérons le cas (b) avec un temps mort au début de S (Figure 3.11.Cas (b)). Soit $\Delta_1 = min(S_{[1]}; C_{sp_1} - D_1)$ et $\Delta_2 = min(D_1 - S_{sp_1}; S_{k_2} - C_{k_1})$; soit A_1 (resp. A_2) l'ensemble des travaux en avance (resp. en retard) de la sous-séquence σ_1 ; et soit α (resp. β) la somme des pondérations d'avance (resp. retard) des travaux de A_1 (resp. A_2). Par conséquent, nous avons $\alpha = \sum_{i \in A_1} \alpha_i$ et $\beta = \sum_{i \in A_2} \beta_i$.

Selon la même approche utilisée pour montrer la propriété 3.3.2.(3), nous avons :

- si $\alpha - \beta \leq 0$ nous pouvons améliorer la solution en déplaçant σ_1 vers la gauche de Δ_1 unités de temps. Ainsi, soit le *splitting job* J_{sp_1} se termine à la date D_1 soit $S_{[1]} = 0$;
- sinon, la sous-séquence σ_1 doit être déplacée vers la droite de Δ_2 unités de temps pour obtenir une meilleure solution. Si $\Delta_2 = D_1 - S_{sp_1}$, alors $S_{sp_1} = D_1$; sinon on supprime le temps mort intermédiaire.

Montrons maintenant que s'il y a un temps mort dans l'intervalle $[D_1, D_2[$, il existe alors un travail J_{i_2} avec $S_{i_2} = D_2$. Considérons la sous-séquence σ_2 des travaux ordonnancés autour de la date D_2. Soient $\Delta'_1 = min(S_{k_2} - C_{k_1}; C_{sp_2} - D_2)$ et $\Delta'_2 = D_2 - S_{sp_2}$. Notons par A'_1 (resp. A'_2) l'ensemble des travaux définissant la deuxième sous-séquence ordonnancés en avance (resp. en retard) et α' (resp. β') la somme des pondérations d'avance (resp. retard) des travaux de A'_1 (resp. A'_2). Par conséquent, $\alpha' = \sum_{i \in A'_1} \alpha_i$ et $\beta' = \sum_{i \in A'_2} \beta_i$. Nous avons donc :

- si $\alpha' - \beta' \leq 0$, la solution peut être alors améliorée en déplaçant la sous-séquence σ_2 vers la gauche de Δ'_1 unités de temps. Si $\Delta'_1 = C_{sp_2} - D_2)$, alors J_{sp_2} se termine à la date D_2 ;
- sinon, la deuxième sous-séquence doit être déplacée vers la droite de Δ'_2 unités de temps. Ainsi, $S_{sp_2} = D_2$.

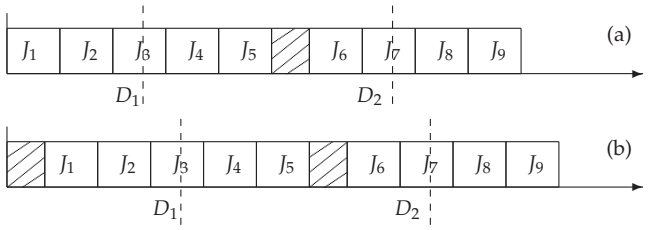

FIG. 3.11 – Exemple avec un temps mort dans $]D_1, D_2]$

Pour résumer, il existe une solution optimale respectant au moins une des configurations suivantes (voir Figure 3.12) :

- Cas (a) : $S_{sp_1} = D_1$ et $S_{sp_2} = D_2$;
- Cas (b) : $S_{sp_1} = D_1$ et un seul temps mort au début (pas de temps mort entre D_1 et D_2) ;
- Cas (c) : $S_{sp_2} = D_2$ et un seul temps mort au début (pas de temps mort entre D_1 et D_2) ;
- Cas (d) : $S_{sp_2} = D_2$ et un seul temps mort entre D_1 et D_2 (pas de temps mort au début) ;

– Cas (e) : $S_{sp_1} \neq D_1$ et $S_{sp_2} \neq D_2$ et il n'y a pas de temps mort au début ni dans l'intervalle $[D_1, D_2]$.

Nous déduisons les deux cas suivants :

– s'il n'existe pas un temps mort entre D_1 et D_2 alors les travaux forment un seul bloc respectant la propriété 3.3.2.(3). Pour chaque cas $S_i \in \{0, D_1, D_2\}$, il est facile de déterminer tous les instants possibles définissant par la suite les dates de début d'exécution des travaux. Le nombre total de dates possibles est $2 \times n$, puisque nous devons considérer n positions avant S_i et $n - 1$ positions après S_i.
– s'il existe un temps mort entre D_1 et D_2, alors les travaux forment deux blocs respectant la propriété 3.3.2.(4). Le problème consiste donc à chercher la position du temps mort. Cette position correspond au nombre de travaux ordonnancés à partir de D_1 et avant le temps mort. Au maximum n positions de temps mort possibles doivent être considérées. Ainsi, pour chaque bloc de travaux, le nombre des instants de début d'exécution des travaux est borné par $2 \times n$. Nous avons donc au maximum $4 \times n$ dates de débuts qui doivent respecter les conditions suivantes :
  – les différentes dates de début d'exécution définissant toutes les positions possibles du premier bloc ne doivent pas être plus grandes que la date de début du temps mort,
  – les différentes dates de début d'exécution définissant toutes les positions possibles du second bloc ne doivent pas être plus petites que la date de début du temps mort.

En conclusion, nous avons $n + 1$ configurations différentes : sans temps mort intermédiaire et n positions possibles pour ce temps mort. Ainsi, pour chacune des $n + 1$ configurations, on détermine les différents instants de début d'exécution – sans se préoccuper de chercher quel travail doit être exécuté à quelle date. Nous construisons par la suite la matrice des coûts d'affectation. La taille de cette matrice est bornée par $(4n \times n)$. Ainsi, pour chaque configuration, une solution optimale peut être déterminée en $O(n^3)$.

La solution optimale globale correspond donc à la meilleure des $2n + 3$ solutions trouvées : 3 solutions différentes pour la première configuration (sans temps mort) tout en considérant la propriété 3.3.2.(3) et 2 solutions obtenues pour chacune des n configurations restantes tout en considérant la propriété 3.3.2.(4).

Théorème 3.3.3 *Une solution optimale pour le problème d'ordonnancement* $1|d_i \in \{D_1, D_2\}|\sum(\alpha_i E_i + \beta_i T_i)$ *peut être déterminée en* $O(n^4)$.

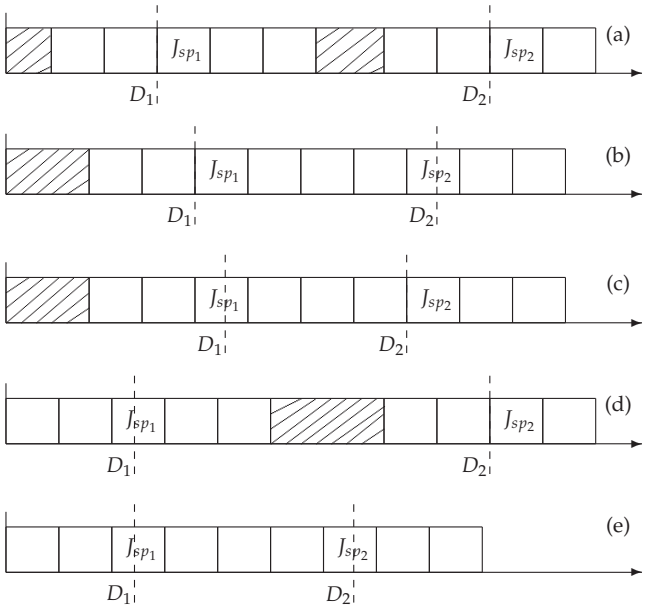

FIG. 3.12 – Exemple avec 5 travaux autour de D_1 et 4 travaux autour de D_2

3.3.3 Une seule machine avec famille de dates de fin souhaitées

Pour généraliser le résultat au cas de l'ordonnancement des travaux sur une seule machine avec ℓ dates de fin souhaitées distinctes, nous allons montrer comment construire les différentes matrices de coûts d'affectation des travaux à partir d'une configuration d'affectation des temps morts intermédiaires.

Sans perte de généralité, nous supposons que les dates de fin souhaitées sont rangées selon l'ordre non-décroissant. Soit D_j la j-ème date de fin souhaitée ($D_j \in D, 0 \leq j \leq \ell$). L'instant 0 est modélisé par D_0.

Soit X_j une variable entière indiquant la position du temps mort entre D_{j-1} et D_j. X_j correspond donc au nombre de travaux ordonnancés entre la date D_{j-1} et la date de début du temps mort. Nous avons donc :

- s'il n'existe pas un temps mort au début de l'ordonnancement, alors $X_0 = 0$; sinon $X_0 = 1$. Ainsi, $X_0 \in \{0, 1\}$,
- s'il n'existe pas un temps mort entre D_{j-1} et D_j, alors $X_j = 0$; sinon $X_j = y$ avec $1 \leq y \leq n-1$. En effet, le nombre de positions possibles d'un temps mort entre D_{j-1} et D_j est borné par $n-1$ (cf. 3.3.1). Nous avons donc $X_j \in \{0, 1, \ldots, n-$

$1\}(0 < j < \ell)$. Si $X_j = 1$ alors il n'existe pas un *splitting job* à la date D_j.

Par exemple, examinons les différentes valeurs de X_j selon la Figure 3.12 :

- Cas (a) : d'une part, il existe un temps mort au début de l'ordonnancement, nous avons donc $X_0 = 1$. D'autre part, trois travaux sont exécutés à partir de la date D_1 et avant le temps mort. Le *splitting job* est à la position 1, les deux autres travaux sont respectivement à la position 2 et 3. Ainsi, le temps mort se trouve à la position 4. D'où $X_1 = 4$;
- Cas (b) : $X_0 = 1$ et $X_1 = 0$;
- Cas (c) : $X_0 = 1$ et $X_1 = 0$;
- Cas (d) : $X_0 = 0$ et $X_1 = 4$;
- Cas (e) : $X_0 = 0$ et $X_1 = 0$.

Pour déterminer une solution optimale, nous devons donc analyser les cas suivants :

- $X_0 = 0$: il n'existe pas de temps mort initial
- $X_0 = 1$: il existe un temps mort initial
- $X_j = 0$ et $j > 0$: il n'existe pas de temps mort entre D_{j-1} et D_j
- $X_j = 1$ et $j > 0$: le seul temps mort entre D_{j-1} et D_j est à la position du *splitting job*,
- $X_j = y, y > 1$ et $j > 0$: le seul temps mort entre D_{j-1} et D_j est à la position y. Dans ce cas, nous avons un *splitting job* qui débute au plus tard à la date D_{j-1}

Notons par X le domaine de définition des valeurs possibles de X_j, $\forall j, j = 0, \ldots, \ell - 1$. La taille de X est donc bornée par $O(n^{\ell-1})$.

Pour chaque vecteur X (définissant une configuration possible des différentes positions des temps morts), nous pouvons déduire alors les différents blocs, au maximum ℓ. Pour l'instant, aucune affectation des travaux aux blocs n'est considérée. Ces blocs représentent en effet les futures sous-séquences de travaux ordonnancés entre deux temps morts successifs. Néanmoins, il est évident qu'il existe une solution optimale où l'intervalle de temps d'exécution des travaux d'une sous-séquence donnée contient une ou plusieurs dates de fin souhaitées (au maximum ℓ dates de fin souhaitées).

Ainsi, nous présentons quelques propriétés du problème étudié.

Propriété 3.3.4 *Il existe une solution optimale qui satisfait les conditions suivantes :*

1. *au maximum, il existe un seul temps mort dans l'intervalle* $[D_0, D_1[$
2. *au maximum, il existe un seul temps mort entre deux dates de fin souhaitées adjacentes* D_{j-1} *et* D_j*, et pas de temps mort entre les travaux ordonnancés après la date* D_ℓ*,*

3. *Pour une sous-séquence donnée, il existe un travail J_i et $D_j, j \geq 0$ tel que $S_i = D_j$.*

Preuve : nous n'allons pas présenter ici les preuves car elles peuvent être obtenues de la même façon que pour le cas de deux dates de fin souhaitées communes (cf. 3.3.2).

Par conséquent, pour une sous-séquence donnée, les dates de début d'exécution définissant les différentes positions des travaux peuvent être calculées de la façon suivante : en partant d'une date $D_j, j \geq 0$, on retranche p de D_j autant que possible à condition que la variable indiquant la position du temps mort précédant la sous-séquence ne s'annule pas. De façon symétrique, on additionne p à D_j autant que possible à condition que la variable indiquant la position du temps mort suivant la sous-séquence ne s'annule pas.

Par exemple, soient les valeurs des deux premières variables X_j données par : $X_0 = 1$ et $X_1 = 2$. Nous avons donc un premier bloc de travaux avec un temps mort au début de l'ordonnancement. Comme $X_1 = 2$, cela implique que la dernière position de ce bloc n'est autre que celle du *splitting job*. Dans ce cas, le *splitting job* débute à la date D_1. Ainsi, nous avons au maximum $k = \left\lfloor \frac{D_1 - 1}{p} \right\rfloor$ positions avant D_1. Les différentes dates de début d'exécution sont données par : $S_{[r]} = D_1 - (k - r) \times \left\lfloor \frac{D_1 - 1}{p} \right\rfloor$, pour toute valeur de $r \in \{1, \ldots, k\}$.

Par conséquent, pour un bloc donné, on génère au maximum $\ell + 1$ sous-séquences différentes. Chacune présente une certaine configuration de dates de début d'exécution calculées à partir d'une date de fin souhaitée D_j appartenant à l'intervalle du temps couvrant l'exécution de l'ensemble des travaux pouvant appartenir au bloc considéré. Comme pour le cas $\ell = 2$, on peut montrer que le nombre de positions est limité par $O(2n\ell)$. En effet, pour chaque sous-séquence, nous avons au maximum n positions avant (resp. après) le travail débutant à une date de fin souhaitée.

Après avoir déterminé toutes les positions possibles de toutes les sous-séquences (au plus $(\ell + 1)^\ell$ cas possibles), on détermine le coût minimum de l'affectation d'un travail à une date de début d'exécution parmi celles définies précédemment.

L'affectation optimale pour une configuration des temps morts donnée, est déterminée en $O(\ell n^3)$.

Théorème 3.3.5 *Une solution optimale pour le problème d'ordonnancement $1|p_i = p, d_i \in D, |D| \leq \ell| \sum(\alpha_i E_i + \beta_i T_i)$ peut être déterminée en $O(\ell^{\ell+1} n^{\ell+2})$.*

3.3.4 Machines uniformes avec famille de dates de fin souhaitées

En suivant la même démarche que précédemment, nous proposons de généraliser les résultats obtenus au cas de m machines parallèles uniformes avec ℓ dates de fin souhaitées.

Soit $X_{(i,j)}$ une variable entière indiquant la position du temps mort entre D_{j-1} et D_j sur la machine M_i avec :

- s'il n'existe pas de temps mort au début de l'ordonnancement sur M_i alors $X_{(i,0)} = 0$; sinon $X_{(i,0)} = 1$. Nous avons donc $X_{(i,0)} \in \{0,1\}$,
- sur la machine M_i, s'il n'existe pas un temps mort entre D_{j-1} et D_j, alors $X_{(i,j)} = 0$; sinon, $X_{(i,j),0<j<\ell} = y$ avec $1 \leq y \leq n-1$. En effet, le nombre de positions possibles d'un temps mort entre D_{j-1} et D_j est borné par $n-1$ (cf. 3.3.1). Nous avons donc $X_{(i,j),0<j<\ell} \in \{0,1,\ldots,n-1\}$. Notons que si $X_j = 1$ alors il n'existe pas un *splitting job* à la date D_j.

X représente le domaine de définition des différentes valeurs possibles des $X_{(i,j),i=1,\ldots,m;j=0,\ldots,(\ell-1)}$. La taille de X est donc bornée par $O(2^m n^{m(\ell-1)})$.

Comme précédemment, pour chaque vecteur dans X définissant une configuration possible des temps morts, et pour chaque machine nous déterminons l'ensemble des sous-séquences de travaux formant un bloc (pas de temps mort intermédiaire entre les travaux appartenant à la même sous-séquence). Nous avons au maximum $m\ell$ sous-séquences.

Conformément à la propriété 3.3.4 et pour chaque sous-séquence, il existe un travail pour lequel sa date de début d'exécution correspond à une date de fin souhaitée ou 0 (au maximum $\ell+1$ cas possibles). Ainsi, nous pouvons déterminer l'ensemble des dates de début d'exécution des autres travaux qui vont être affectés à la sous-séquence considérée.

Pour une configuration des temps morts donnée, nous avons donc au maximum $(\ell+1)^{m\ell}$ cas pour déterminer toutes les dates de début d'exécution possibles des travaux sur les m machines. Pour chaque cas, une matrice d'affectation est définie. Le nombre de positions total est borné par $O(2nm\ell)$. Par conséquent, l'affectation optimale des travaux aux différentes positions est calculée en $O(m\ell n^3)$.

Théorème 3.3.6 *Une solution optimale pour le problème d'ordonnancement* $Qm|p_i = p, d_i \in D, |D| \leq \ell| \sum(\alpha_i E_i + \beta_i T_i)$ *peut être déterminée en* $O(m2^m \ell^{m\ell+1} n^{m(\ell-1)+3})$.

Remarque 3.3.1 *La complexité pour le cas de machines identiques est la même (i.e.* $O(m2^m\ell^{m\ell+1}n^{m(\ell-1)+3})$*). En effet, la complexité ne dépend pas de la performance des machines. Néanmoins, les vitesses des machines doivent être considérées pour déterminer les matrices d'affectation.*

3.4 Durées opératoires quelconques

Nous considérons maintenant le cas général où n travaux indépendants doivent être ordonnancés sur m machines parallèles générales pour la minimisation de la somme de l'avance et retard pondérés. Le problème est noté $Pm|d_i = d, non-restrictive|\sum(\alpha_i E_i + \beta_i T_i)$. Connu $\mathcal{N}P$-difficile, nous nous sommes focalisés sur l'étude de l'existence d'un schéma d'approximation polynomial (PTAS). En effet, en s'appuyant sur la technique dite *Partitionnement des ratios* proposée par Afrati et Milis [4], nous montrons que le problème étudié admet un PTAS. La technique utilisée a été initialement proposée pour montrer l'existence d'un PTAS pour le problème $Rm||\sum w_i C_i$ [3].

Tout d'abord, étudions le cas d'une seule machine.

3.4.1 PTAS pour le cas d'une seule machine

Tous les travaux J_i $(i = 1, \ldots, n)$ doivent être réalisés pour une date de fin souhaitée d assez grande, i.e. cas non-restrictif. Le problème d'ordonnancement considéré est noté $1|d_i = d, non-restrictive|\sum(\alpha_i E_i + \beta_i T_i)$. Nous présentons d'abord quelques nouvelles notations utilisées dans cette partie.

- I_1, I_2 : l'intervalle de temps $(0, d]$ and $[d, +\infty)$;
- $J_{[i]}$ ou $[i]$: le travail à la i-ème position pour une séquence donnée ;
- $W^{(1)}_{[i]}$: la somme des pondérations d'avance des $(i-1)$ premiers travaux avec $J_{[i]}$ le i-ème travail ordonnancé en avance :

$$W^{(1)}_{[i]} = \begin{cases} \sum_{l=1}^{i-1} \alpha_{[l]} & \text{si } C_{[i]} \leq d \\ 0 & \text{sinon} \end{cases}$$

- $W^{(1)}_i$: la somme des pondérations d'avance des travaux ordonnancés avant le

travail J_i avec J_i un travail ordonnancé en retard :

$$W_i^{(1)} = \begin{cases} W_{[l]}^{(1)} & \text{si } J_i \text{ est ordonnancé à la } l\text{-ème position et } C_{[l]} \leq d \\ 0 & \text{sinon} \end{cases}$$

– $W_{[i]}^{(2)}$: la somme des pondérations de retard des travaux ordonnancés en retard entre la position i et la dernière position n :

$$W_{[i]}^{(2)} = \begin{cases} \sum_{l=i}^{n} \beta_{[l]} & \text{si } C_{[i]} > d \\ 0 & \text{sinon} \end{cases}$$

– $W_i^{(2)}$: la somme des pondérations de retard du travail J_i et des travaux ordonnancés après J_i, J_i un travail en retard :

$$W_i^{(2)} = \begin{cases} W_{[l]}^{(2)} & \text{si } J_i \text{ est ordonnancé à la } l\text{-ème position et } C_{[l]} > d \\ 0 & \text{sinon} \end{cases}$$

– $W_{[i]}$: la somme des pondérations d'avance et retard total par rapport au travail ordonnancé à la i-ème position (travail $J_{[i]}$) ; $W_{[i]} = W_{[i]}^{(1)} + W_{[i]}^{(2)}$
– W_i : la somme des pondérations d'avance et retard total par rapport au travail J_i) ; $W_i = W_i^{(1)} + W_i^{(2)}$
– $Z = f(S) = \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i)$: l'objectif à minimiser.

3.4.1.1 Reformulation de la fonction objectif

Selon la propriété 3.2.4, la fonction objectif d'une solution optimale $Z = \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i)$ peut être écrite sous la forme suivante : $Z = \sum_{i=1}^{n} p_i W_i$.

En effet, pour une séquence optimale S^*, soit $J_{[k]}$ le travail juste-à-temps. Nous avons donc :

$$\begin{aligned} \sum_{i=1}^{n} \alpha_i E_i &= \sum_{i=1}^{k} \alpha_{[i]} E_{[i]} = \sum_{i=1}^{k} \left(\alpha_{[i]} \sum_{l=i+1}^{k} p_{[l]} \right) \\ &= \sum_{l=1}^{k} \left(p_{[l]} \sum_{i=1}^{l-1} \alpha_{[i]} \right) = \sum_{l=1}^{k} p_{[l]} W_{[l]}^{(1)} = \sum_{l=1}^{k} p_{[l]} W_{[l]} \end{aligned}$$

et

$$\begin{aligned}\sum_{i=1}^{n}\beta_i T_i &= \sum_{i=k+1}^{n}\beta_{[i]}T_{[i]} = \sum_{i=k+1}^{n}\left(\beta_{[i]}\sum_{l=k+1}^{i}p_{[l]}\right)\\ &= \sum_{l=k+1}^{n}\left(p_{[l]}\sum_{i=l}^{n}\beta_{[i]}\right) = \sum_{l=k+1}^{n}p_{[l]}W_{[l]}^{(2)} = \sum_{l=k+1}^{n}p_{[l]}W_{[l]}.\end{aligned}$$

$\Rightarrow Z = \sum_{l=1}^{k} p_{[l]}W_{[l]} + \sum_{l=k+1}^{n} p_{[l]}W_{[l]} = \sum_{l=1}^{n} p_{[l]}W_{[l]}$.

Exemple : soient 9 travaux ordonnancés selon le diagramme de Gantt de la figure 3.13.

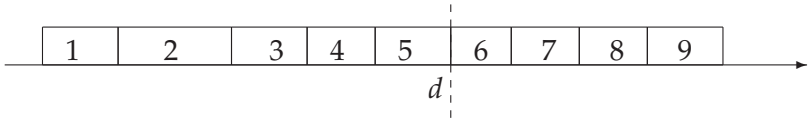

FIG. 3.13 – Reformulation de la fonction objectif pour 9 travaux

La valeur de l'objectif est donc :

$$\begin{aligned}Z &= \alpha_1(p_2+p_3+p_4+p_5)+\alpha_2(p_3+p_4+p_5)+\alpha_3(p_4+p_5)+\alpha_4(p_5)\\ &\quad +\beta_6(p_6)+\beta_7(p_6+p_7)+\beta_8(p_6+p_7+p_8)+\beta_9(p_6+p_7+p_8+p_9)\\ &= 0.p_1+\alpha_1p_2+(\alpha_1+\alpha_2)p_3+(\alpha_1+\alpha_2+\alpha_3)p_4+(\alpha_1+\alpha_2+\alpha_3+\alpha_4)p_5\\ &\quad +(\beta_6+\beta_7+\beta_8+\beta_9)p_6+(\beta_7+\beta_8+\beta_9)p_7+(\beta_8+\beta_9)p_8+\beta_9p_9\\ &= W_1^{(1)}p_1+W_2^{(1)}p_2+W_3^{(1)}p_3+W_4^{(1)}p_4+W_5^{(1)}p_5\\ &\quad +W_6^{(2)}p_6+W_7^{(2)}p_7+W_8^{(2)}p_8+W_9^{(2)}p_9\\ &= \sum_{i=1}^{9}p_iW_i.\end{aligned}$$

La réécriture de la fonction objectif nous permet donc de nous référer aux résultats de la littérature spécialisée définissant des schémas d'approximation pour les problèmes d'ordonnancement monocritères de type min-sum (voir l'état de l'art dans [4]). En effet, pour notre problème, nous allons suivre la démarche qui a permis de définir un PTAS pour le $Rm||\sum w_iC_i$ [3].

Toutefois, on peut penser que les deux problèmes suivants $R2||\sum C_i$ (notons ici $w_i = 1$, pour tout i) et $1|p_i = 1, d_i = d, non-restrictive|\sum(\alpha_iE_i+\beta_iT_i)$ sont équivalents en faisant la correspondance suivante : les durées opératoires des travaux sur

la première (resp. deuxième) machine du $R2||\sum C_i$ correspondent aux α_i (resp. β_i). Malheureusement, une solution optimale pour le problème $1|p_i = 1, d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ n'est pas forcément optimale pour le cas polynomial $R2||\sum C_i$. Ci-dessous un exemple montrant qu'une solution optimale pour l'un n'est pas forcément optimale pour l'autre.

Exemple : Soit une instance de 5 travaux. Les pénalités d'avances et retards pour le problème $1|p_j = 1, d_j = d, non - restrictive\ |\sum \alpha_i E_i + \beta_i T_i$ sont présentées dans la Table 3.3.

TAB. 3.3 – Durées opératoires et les poids (pénalités d'avance et de retard) des travaux

	J_1	J_2	J_3	J_4	J_5
p_i	1	1	1	1	1
α_i	6	5	7	2	5
β_i	5	4	2	4	4

Soit $p_{1,i}$ (resp. $p_{2,i}$) la durée opératoire de $J_{i,i=1,\ldots,n}$ sur la première (resp. la deuxième) machine pour le problème $R2||\sum C_i$, avec : $p_{1,i} = \alpha_i$; $p_{2,i} = \beta_i$. Le diagramme de Gantt de la figure 3.14 illustre une solution optimale avec : $Z(S^*) = \sum C_i = 2 + 8 + 2 + 6 + 10$ = **28**.

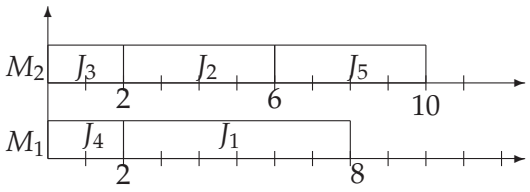

FIG. 3.14 – Solution optimale S^* pour le $R2||\sum C_i$

La séquence S^* définit une partition de tous les travaux en deux sous-ensembles. A partir de cette solution nous pouvons construire une meilleure solution pour le $1|d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ comme suit : les travaux ordonnancés sur M_1 (resp. M_2) sont en avance (en retard). Ainsi, la meilleure solution pour le problème $1|p_i = 1, d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ est de **20**, illustrée par le diagramme de Gantt de la figure 3.15).

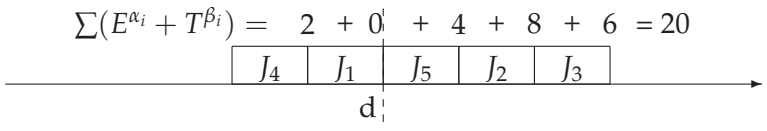

FIG. 3.15 – Une meilleure solution déduite de la solution optimale du $R2||\sum C_i$

Or, pour le $1|p_i = 1, d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$, on peut trouver une solution optimale S' de valeur **17** (cf. Figure 3.16).

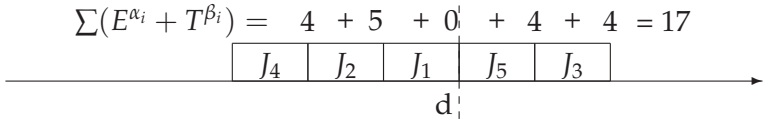

FIG. 3.16 – Solution optimale du problème $1|d_j = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$

Evidemment, selon S', la répartition des travaux sur les machines ne correspond à aucune solution optimale du $R2||\sum C_i$. Au mieux, nous obtenons une solution respectant une telle partition des travaux avec $\sum C_i = 2 + 7 + 13 + 2 + 6 = \mathbf{30}$ (cf. figure 3.17).

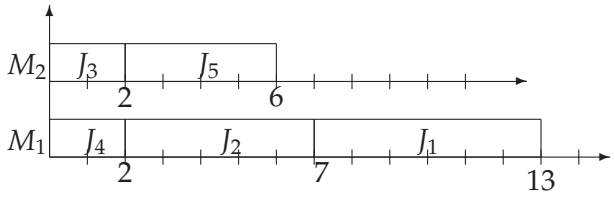

FIG. 3.17 – Solution déduite à partir de S' pour le $R2||\sum C_i$

En conclusion, l'affectation des travaux aux machines ne nous permet pas de déduire l'ensemble des travaux en avance et en retard pour le problème JàT. Néanmoins, pour le $R2||\sum w_i C_i$, la durée opératoire dépend de la machine sur laquelle le travail est exécuté. Par analogie, pour le $1|d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$, les pénalités dépendent des intervalles de temps $[0, d]$ et $]d, +\infty)$. Par conséquent, nous montrons qu'une adaptation de l'algorithme de programmation dynamique présenté dans [3] appliqué aux matrices d'incidence (voir définition 3.4.3) du problème $1|d_i = dnon - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ permet de définir un PTAS.

3.4.1.2 Schéma d'Approximation en Temps Polynomial

Pour le cas d'une seule machine, si on connaît les deux sous-ensembles de travaux qui doivent être ordonnancés en avance et en retard, alors une solution optimale est obtenue selon la règle 'V-shape'. Pour le PTAS, deux étapes principales seront donc nécessaires. L'idée générale est la suivante : la première étape a pour objectif de simplifier la taille des entrées en utilisant la technique d"'arrondi géométrique". En effet, cette technique nous permet de prendre une décision pour un certain nombre de tra-

vaux (ceux ordonnancés en avance et ceux en retard). Nous obtenons donc deux sous-ensembles de travaux : "décidés" et "non-décidés". Par la suite, nous nous concentrons seulement sur les décisions à prendre pour les travaux "non-décidés". C'est l'objectif de la deuxième étape où nous allons montrer que l'ensemble de toutes les répartitions possibles des travaux non-décidés dans l'ensemble des travaux en avance ou dans l'ensemble des travaux en retard est borné. Ainsi, le programme dynamique définissant le PTAS est déduit.

Simplification de la taille des entrées

Proposition 3.4.1 *Transformer les pénalités d'avance, les pénalités de retard et les durées opératoires en des puissances entières de* $(1+\varepsilon)$*, le coût global sera donc augmenté au plus de* $(1+O(\varepsilon))$*.*

Preuve : Les deux étapes suivantes nous permettent en effet d'arrondir les différentes données du problème. Tout d'abord, on multiplie chaque durée opératoire par $1+\varepsilon$. La valeur de la solution sera donc augmentée d'un montant égal, i.e. $1+\varepsilon$. Puis on remplace chaque nouvelle valeur par une plus grande puissance de $1+\varepsilon$ mais inférieure à la nouvelle valeur. Il est à noter que la valeur finale obtenue est toujours supérieure à la valeur initiale. En opérant de la même façon sur les pénalités d'avance et de retard, on peut montrer que l'augmentation du coût total de la solution est bornée par $(1+\varepsilon)^2 = 1+2\varepsilon+\varepsilon^2 \leq 1+3\varepsilon$. □

Soient S une solution optimale et $J_{[1]}$ le travail exécuté en première position dans S.

Proposition 3.4.2 *Sous l'hypothèse de* $\alpha_{min} \leq \alpha_i \leq \alpha_{max}$ *et* $\alpha_{max}/\alpha_{min} \leq 2$*, l'ordonnancement des travaux ayant* $\alpha_i < \varepsilon\beta_i$ *en avance et l'ordonnancement des travaux ayant* $\beta_i < \varepsilon\alpha_i$ *en retard (sauf le travail* $J_{[1]}$*) augmentera le coût global d'au plus de* $(1+O(\varepsilon))f(S)$*.*

Preuve : Pour une solution S optimale, soit $\mathcal{A}$ (resp. $\mathcal{B}$) l'ensemble des travaux ordonnancés en retard tels que $\alpha_i < \varepsilon\beta_i$ (resp. en avance tels que $\beta_i < \varepsilon\alpha_i$). Soit S' l'ordonnancement construit à partir de S où tous les travaux de $\mathcal{A}$ sont déplacés et ordonnancés en avance selon l'ordre inversé de leur apparition dans S. Leur placement en avance ne respecte pas nécessairement le "V-shape".

Considérant un travail $J_i \in \mathcal{A}$. J_i est inséré avant la date d et juste après le dernier travail J_j satisfaisant l'inégalité $W_j^{(1)}(S) \leq W_i^{(2)}(S)$; de même pour tous les travaux de

$\mathcal{A}$. Si deux travaux de $\mathcal{A}$ doivent être insérés à la même position, nous devons donc respecter l'ordre inverse de leur apparition initiale, c-à-d dans S.

Ainsi, dans S', tous les travaux ordonnancés après d ont un poids de retard cumulé inférieur ou égal à la somme des pondérations des retards des mêmes travaux dans S. Les travaux ordonnancés en avance dans S' ont un nouveau poids d'avance cumulé (la somme des pondérations de l'avance). On note par $W_i^{(1)}(S')$ le nouveau poids d'avance cumulé de chaque travail $J_i \in \mathcal{A}$ dans S'. Par hypothèse, du fait que $\sum_{k\in\mathcal{A},W_k^{(2)}(S)\leq W_i^{(2)}(S)} \beta_k \leq W_i^{(2)}(S)$, nous avons :

$$\begin{aligned}
W_i^{(1)}(S') &= W_j^{(1)}(S) + \sum_{k\in\mathcal{A},W_k^{(1)}(S)<W_i^{(2)}(S)} \alpha_k + \alpha_j \\
&\leq W_i^{(2)}(S) + \sum_{k\in\mathcal{A},W_k^{(1)}(S)<W_i^{(2)}(S)} \alpha_k + \alpha_j \\
&\leq W_i^{(2)}(S) + \sum_{k\in\mathcal{A},W_k^{(1)}(S)<W_i^{(2)}(S)} \alpha_k + 2\alpha_i \\
&< W_i^{(2)}(S) + 2 \sum_{k\in\mathcal{A},W_k^{(1)}(S)\leq W_i^{(2)}(S)} \alpha_k \\
&\leq W_i^{(2)}(S) + 2 \sum_{k\in\mathcal{A},W_k^{(1)}(S)\leq W_i^{(2)}(S)} \varepsilon\beta_k \\
&\leq W_i^{(2)}(S) + 2\varepsilon W_i^{(2)}(S) \\
&\leq (1+2\varepsilon)W_i^{(2)}(S).
\end{aligned}$$

Il faut maintenant calculer la nouvelle somme des pondérations d'avance (poids d'avances cumulés) de chaque travail initialement ordonnancé en avance dans S et exécuté après la position d'insertion des travaux de $\mathcal{A}$. Nous avons :

$$\begin{aligned}
W_k^{(1)}(S') &= W_k^{(1)}(S) + \sum_{i\in\mathcal{A},W_k^{(1)}(S)>W_i^{(2)}(S)} \alpha_i \\
&\leq W_k^{(1)}(S) + \sum_{i\in\mathcal{A},W_k^{(1)}(S)>W_i^{(2)}(S)} \varepsilon\beta_i \\
&\leq W_k^{(1)}(S) + \varepsilon \max_{i\in\mathcal{A},W_k^{(1)}(S)>W_i^{(2)}(S)} W_i^{(2)}(S) \\
&\leq W_k^{(1)}(S) + \varepsilon W_k^{(1)}(S) \leq (1+\varepsilon)W_k^{(1)}(S).
\end{aligned}$$

D'où : $f(S') \leq (1+2\varepsilon)f(S)$.

Maintenant nous considérons le déplacement des travaux de $\mathcal{B}$. Soit S'' l'ordonnancement construit à partir de S' où tous les travaux de $\mathcal{B}$ ont été déplacés et ordonnancés en retard selon l'ordre inversé de leur apparition dans S'. Leur placement en retard ne respecte pas nécessairement le "V-shape". Considérant un travail $J_i \in \mathcal{B}$. J_i est inséré après la date d et juste avant le premier travail J_j satisfaisant l'inégalité $W_j^{(2)}(S') \leq W_i^{(1)}(S')$; de même pour tous les travaux de $\mathcal{B}$. Si deux travaux de $\mathcal{A}$ doivent être insérés à la même position, nous devons donc respecter l'ordre inverse de leur apparition initiale, c-à-d dans S'.

Ainsi, dans S'', tous les travaux ordonnancés avant d ont un poids de retard cumulé inférieur ou égal à la somme des pondérations des retards des mêmes travaux dans S'. Les travaux ordonnancés en retard dans S'' ont un nouveau poids de retard cumulé (la somme des pondérations du retard). On note par $W_i^{(2)}(S'')$ le nouveau poids de retard cumulé de chaque travail $J_i \in \mathcal{B}$ dans S''. Par hypothèse, du fait que $\sum_{k\in\mathcal{B},W_k^{(1)}(S')<W_i^{(1)}(S')} \alpha_k \leq W_i^{(1)}(S')$ et $\alpha_i \leq 2\alpha_{min} \leq 2W_i^{(1)}(S')$ (car $J_i \neq J_{[1]}$), nous avons :

$$\begin{aligned}
W_i^{(2)}(S'') &= W_j^{(2)}(S') + \sum_{k\in\mathcal{B},W_k^{(1)}(S')\leq W_i^{(1)}(S')} \beta_k \\
&\leq W_i^{(1)}(S') + \sum_{k\in\mathcal{B},W_k^{(1)}(S')\leq W_i^{(1)}(S')} \varepsilon\alpha_k \\
&\leq (1+\varepsilon)W_i^{(1)}(S') + \varepsilon\alpha_i \\
&\leq (1+3\varepsilon)W_i^{(1)}(S').
\end{aligned}$$

Il faut maintenant calculer la nouvelle somme des pondérations de retard (poids de retard cumulés) de chaque travail initialement ordonnancé en retard dans S' et exécuté avant la position d'insertion des travaux de $\mathcal{B}$. Nous avons :

$$\begin{aligned}
W_k^{(2)}(S'') &= W_k^{(2)}(S') + \sum_{i\in\mathcal{B},W_k^{(2)}(S')>W_i^{(1)}(S')} \beta_i \\
&\leq W_k^{(2)}(S') + \sum_{i\in\mathcal{B},W_k^{(2)}(S')>W_i^{(1)}(S')} \varepsilon\alpha_i \\
&\leq W_k^{(2)}(S') + \varepsilon \max_{i\in\mathcal{B},W_k^{(2)}(S')>W_i^{(1)}(S')} (W_i^{(1)}(S') + \alpha_i) \\
&\leq W_k^{(2)}(S') + 3\varepsilon \max_{i\in\mathcal{B},W_k^{(2)}(S')>W_i^{(1)}(S')} W_i^{(1)}(S') \\
&\leq (1+3\varepsilon)W_k^{(2)}(S').
\end{aligned}$$

D'où : $f(S'') \leq (1+3\varepsilon)f(S') \leq (1+3\varepsilon)(1+2\varepsilon)f(S)$.

Remarque : la solution S' peut être améliorée en exécutant les travaux selon la règle V-shape. La solution obtenue est toujours bornée par $(1+O(\varepsilon))f(S)$. □

Définition 3.4.1 *Concernant les décisions à prendre, les trois situations suivantes sont à considérer :*

1. *Travaux décidés*
   - *Travail J_i est dit* `décidé en avance` ***ssi*** $\alpha_i < \varepsilon\beta_i$
   - *Travail J_i est dit* `décidé en retard` ***ssi*** $\beta_i < \varepsilon\alpha_i$
2. *Travail avance/retard non-décidé*
   - *Travail J_i est dit un travail* `avance/retard non-décidé` ***ssi*** $\varepsilon\beta_i \leq \alpha_i \leq \beta_i/\varepsilon \Leftrightarrow \varepsilon\alpha_i \leq \beta_i \leq \alpha_i/\varepsilon$

Par la suite, nous allons montrer que l'ensemble de toutes les répartitions possibles des travaux non-décidés en deux sous-ensembles (avance et retard) est borné. Pour cela, nous allons d'abord définir les types de travaux comme suit.

Définition 3.4.2 *Le type $T_{r,r'}$ est l'ensemble des travaux ayant la même paire de ratios $(\alpha_i/p_i, \beta_i/p_i)$. $T_{r,r'}$ est défini comme suit : $T_{r,r'} = \{i | r_i^{(1)} = r \wedge r_i^{(2)} = r'\}$ où $r_i^{(1)} = \alpha_i/p_i$ et $r_i^{(2)} = \beta_i/p_i$.*

Soit $W_{r,r'}$ la somme des poids (pénalités d'avance et de retard) totale des travaux de type $T_{r,r'}$, et soit $P_{r,r'}$ la somme des durées opératoires des travaux de type $T_{r,r'}$.

Pour une séquence S donnée, soit $T_{r,r'}^{(1)}(S)$ l'ensemble des travaux de type $T_{r,r'}$ ordonnancés dans l'intervalle $[0,d]$, et soit $T_{r,r'}^{(2)}(S)$ l'ensemble des travaux de type $T_{r,r'}$ ordonnancés en retard. Soient $W_{r,r'}^{(l)}(S)$ et $P_{r,r'}^{(l)}(S)$ la somme des poids et la somme des durées opératoires des travaux de $T_{r,r'}^{(l)}(S)$ pour tout $l = 1,2$. On définit : $f_{r,r'}^{(l)}(S) = P_{r,r'}^{(l)}(S)/P_{r,r'}$

Nous considérons l'ensemble des travaux ayant des durées opératoires inférieures à $\varepsilon \times P_{r,r'}$, noté par $T_{r,r',\varepsilon}$. Soit $P_{r,r',\varepsilon}$ la somme des durées opératoires des travaux de $T_{r,r',\varepsilon}$. Pour la séquence S, $T_{r,r',\varepsilon}^{(1)}(S)$ (resp. $T_{r,r',\varepsilon}^{(2)}(S)$) définit l'ensemble des travaux de $T_{r,r',\varepsilon}$ ordonnancés dans l'intervalle $[0,d]$. (resp. dans l'intervalle $(d,+\infty)$). Soient $W_{r,r',\varepsilon}^{(l)}(S)$ et $P_{r,r',\varepsilon}^{(l)}(S)$ la somme des poids et la somme des durées opératoires des travaux de $T_{r,r',\varepsilon}^{(l)}(S)$ pour tout $l = 1,2$. Par conséquent, $P_{r,r',\varepsilon} = P_{r,r',\varepsilon}^{(1)}(S) + P_{r,r',\varepsilon}^{(2)}(S)$. Nous définissons $f_{r,r',\varepsilon}^{(l)}(S) = P_{r,r',\varepsilon}^{(l)}(S)/P_{r,r',\varepsilon}(S)$ pour tout $l = 1,2$.

Afin de montrer que la taille de chaque type $T_{r,r'}$ est bornée par une fonction en ε, nous devons concaténer les petits travaux (i.e. de petites durées opératoires) du même type pour former de nouveaux travaux. Ces nouveaux travaux sont dits "*travaux fictifs*".

Proposition 3.4.3 *Pour une séquence S quelconque, si $f^{(1)}_{r,r',\varepsilon}(S) \leq \varepsilon$, alors tous les petits travaux appartenant à $T_{r,r',\varepsilon}$ peuvent être déplacés et ordonnancés selon la règle WSPT après la date d.*

Preuve. Les travaux du même type ordonnancés en avance (appartenant à $T^{(1)}_{r,r',\varepsilon}(S)$) seront déplacés et insérés juste après les travaux $T^{(2)}_{r,r',\varepsilon}(S)$. Ainsi, la date de fin d'exécution de chaque travail ordonnancé après ceux de $T^{(2)}_{r,r',\varepsilon}(S)$ sera augmenté d'un facteur borné par :

$$\frac{P_{r,r',\varepsilon}(S)}{P^{(1)}_{r,r',\varepsilon}(S)} = \frac{P_{r,r',\varepsilon}(S)}{P_{r,r',\varepsilon}S) - P^{(2)}_{r,r',\varepsilon}(S)} \leq \frac{1}{1-\varepsilon} = 1 + \frac{\varepsilon}{1-\varepsilon}$$

Le nouveau coût des travaux de $T_{r,r',\varepsilon}$ est donc augmenté d'un facteur borné par :

$$\frac{W_{r,r',\varepsilon}(S)}{W^{(1)}_{r,r',\varepsilon}(S)} = \frac{W_{r,r',\varepsilon}(S)}{W_{r,r',\varepsilon}(S) - W^{(2)}_{r,r',\varepsilon}(S)} \leq \frac{1}{1-\varepsilon} = 1 + \frac{\varepsilon}{1-\varepsilon}$$

Le coût global est par conséquent augmenté d'un facteur borné par $(1+\varepsilon/(1-\varepsilon))^2$.
□

Soient deux sous-ensembles suivants : $A_1 \subseteq T^{(1)}_{r,r',\varepsilon}(S)$ et $A_2 \subseteq T^{(2)}_{r,r',\varepsilon}(S)$ avec $|A_1| = |A_2|$ et $|\sum_{J_i \in A_1} p_i - \sum_{J_i \in A_2} p_i| \leq \varepsilon \min(\sum_{J_i \in A_1} p_i; \sum_{J_i \in A_2} p_i)$, nous avons :

Proposition 3.4.4 *L'augmentation du coût dûe aux permutations des travaux des deux sous-ensembles A_1 et A_2 est négligeable.*

Preuve. idem que la preuve de la proposition 3.4.3. □

Proposition 3.4.5 *Nous pouvons supposer que : $p_i \geq \varepsilon^4 P_{r,r',\varepsilon}$ pour tout $i \in T_{r,r',\varepsilon}$. Par conséquent, chaque type de travail $T_{r,r'}$ a au plus $(2/\varepsilon^4)$ éléments. Sous cette hypothèse, l'augmentation du coût global est négligeable.*

Preuve. Considérons tous les travaux du même type dont la durée opératoire est inférieure à $\varepsilon^4 P_{r,r'}$. Les petits travaux du même type sont partitionnés en sous-groupes de sorte que la somme des durées opératoires des travaux de chaque sous-groupe est dans l'intervalle $[\varepsilon^4 P_{r,r',\varepsilon}, 3\varepsilon^4 P_{r,r',\varepsilon}]$. L'ensemble des sous-groupes peut être obtenu comme suit : on classe d'abord ces travaux selon l'ordre SPT ; puis en parcourant la liste obtenue on ajoute au sous-groupe un nouveau travail tant que la durée opératoire cumulée est inférieure à $\varepsilon^4 P_{r,r',\varepsilon}$. Chaque sous-groupe obtenu est appelé "travail fictif". Le poids

(pénalité d'avance-retard) d'un travail fictif est donné par la somme des poids cumulés des petits travaux formant le sous-groupe en question. La durée opératoire du travail fictif n'est autre que la durée opératoire cumulée.

Comme les travaux du même type ont les mêmes ratios, ils peuvent donc être ordonnancés consécutivement dans un ordre quelconque. Ainsi, nous pouvons supposer que ces travaux sont ordonnancés en avance (resp. en retard) selon l'ordre LPT (resp. SPT).

Selon les propositions 3.4.3 et 3.4.4, les petits travaux de même type peuvent être remplacés par leurs travaux fictifs correspondants.

Pour une séquence donnée S, le remplacement des petits travaux s'effectue de la manière suivante :

- pour chaque travail J_i en avance, sa date de début d'exécution est modifiée comme suit : $S_i - \varepsilon(d - S_i)$; de même, la nouvelle date de fin d'exécution de chaque travail J_i en retard est donnée par : $C_i + \varepsilon(C_i - d)$. Par conséquent, la séquence est allongée d'un facteur de $(1 + \varepsilon)$.
- on forme des blocs constitués de travaux du même type. Pour chaque bloc de travaux en retard, on décale ses travaux, à l'exception du premier, vers la gauche pour obtenir un temps mort à la fin du bloc. De même, on décale les travaux en avance du même bloc, à l'exception du dernier, vers la droite pour former un temps mort au début du bloc. Par la suite, le premier (resp. dernier) bloc de travaux en retard (resp. avance) doit être décalé vers la gauche (resp. droite) jusqu'à la date d.
- de chaque bloc, on supprime les petits travaux pour les remplacer par leurs travaux fictifs
- pour les positions libérées d'un bloc d'un type donné, on leur affecte les travaux fictifs du même type que ceux du bloc considéré. Un travail fictif d'un type donné peut être affecté à une position en avance ou en retard occupée auparavant par les petits travaux du même type supprimés.

L'ensemble des travaux fictifs du même type insérés avant la date d (resp. après la date d) est noté par $TF^{(1)}_{r,r',\varepsilon^4}(S)$ (resp. $TF^{(2)}_{r,r',\varepsilon^4}(S)$). Ainsi, le coût engendré par le remplacement des petits travaux par ceux de $TF^{(1)}_{r,r',\varepsilon^4}(S)$ (resp. $TF^{(2)}_{r,r',\varepsilon^4}(S)$) est diminué (resp. augmenté).

Examinons le cas d'un seul travail fictif. S'il est en avance (resp. en retard), le coût global est diminué (resp. augmenté) d'une quantité bornée par $9r\varepsilon^8 P^2_{r,r',\varepsilon}$ (resp. $9r'\varepsilon^8 P^2_{r,r',\varepsilon}$). Par exemple, selon la Figure 3.18 :

- le coût dû au remplacement des petits travaux formant le travail fictif J_{k_1} (i.e groupe k_1) est diminué d'une quantité suivante : $\alpha_1(d-C_1)+\alpha_2(d-C_2)+\alpha_3(d-C_3)-\alpha_{k_1}(d-C_{k_1}) = \alpha_{k_1}C_{k_1}-\alpha_1C_1-\alpha_2C_2-\alpha_2C_3 = (\alpha_1+\alpha_2+\alpha_3)C_{k_1}-\alpha_1C_1-\alpha_2C_2-\alpha_2C_3 = \alpha_1(p_2+p_3)+\alpha_2p_3 = r(p_1p_2+p_1p_3+p_2p_3) < r(p_1+p_2+p_3)^2 = rp^2_{k_1} \leq 9r\varepsilon^8 P^2_{r,r',\varepsilon}$.
- le coût dû au remplacement des petits travaux formant le travail fictif J_{k_2} (i.e groupe k_2) est augmenté d'une quantité suivante : $\alpha_{k_2}(d-C_{k_2})-\alpha_4(C_4-d)+\alpha_5(C_5-d)+\alpha_6(C_6-d) = (\alpha_4+\alpha_5+\alpha_6)C_{k_2}-\alpha_4C_4-\alpha_5C_5-\alpha_6C_6 = \alpha_4(p_4+p_5+p_6)+\alpha_5(p_5+p_6) = r'(p^2_4+p_4p_5+p_4p_6+p^2_5+p_5p_6) < r'(p_4+p_5+p_6)^2 = r'p^2_{k_2} \leq 9r'\varepsilon^8 P^2_{r,r',\varepsilon}$.

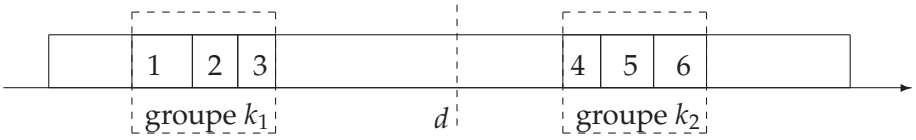

FIG. 3.18 – Exemple de remplacement des petits travaux

Soient $S^{(l)}_{r,r'}$ et $C^{(l)}_{r,r'}$ la date de début d'exécution et la date de fin d'exécution des travaux du type $T_{r,r'}$ dans l'intervalle I_l. Soient K_1 et K_2 le nombre des travaux fictifs d'un même type insérés avant et après la date d. Par conséquent, $K_1 \leq 1/\varepsilon^4$ et $K_2 \leq 1/\varepsilon^4$. Le coût induit par le remplacement des travaux fictifs est diminué (resp. augmenté) d'une quantité bornée par $9r\varepsilon^4 P^2_{r,r',\varepsilon}$ (resp. $9r'\varepsilon^4 P^2_{r,r',\varepsilon}$)

- Pour l'ensemble des travaux en avance, la nouvelle contribution au coût des travaux d'un même type est donnée par : $W^{(1)}_{r,r'}(d-C^{(1)}_{r,r'})+r\sum_{i\in T^{(1)}_{r,r'}}\sum_{j\in T^{(1)}_{r,r'},j\neq i}p_ip_j$. Selon la proposition 3.4.3, on ne considère que le cas $f^{(1)}_{r,r',\varepsilon}(S) \geq 4\varepsilon$ (dans le cas contraire, tous les travaux fictifs peuvent être ordonnancés en retard). Nous

avons :

$$\begin{aligned} r \sum_{i \in T^{(1)}_{r,r'}} \sum_{j \in T^{(1)}_{r,r'}, j \neq i} p_i p_j &\geq r \sum_{i \in TF^{(1)}_{r,r',\varepsilon^4}(S)} \sum_{j \in TF^{(1)}_{r,r',\varepsilon^4}(S), j \neq i} p_i p_j \\ &\geq (1/2) r \Big(\sum_{i \in TF^{(1)}_{r,r',\varepsilon^4}(S)} p_i \Big)^2 - (1/2) r \sum_{j \in TF^{(1)}_{r,r',\varepsilon^4}(S)} p_j^2 \\ &\geq 8 r \varepsilon^2 P^2_{r,r',\varepsilon} - (1/2) K_1 \varepsilon^8 P^2_{r,r',\varepsilon} \\ &\geq 8 r \varepsilon^2 P^2_{r,r',\varepsilon} - (1/2) \varepsilon^4 P^2_{r,r',\varepsilon} \\ &\geq (1/\varepsilon) 9 r \varepsilon^4 P^2_{r,r',\varepsilon}. \end{aligned}$$

Ainsi, la diminution du coût global induit par le remplacement des petits travaux ordonnancés en avance est bornée par $O(\varepsilon) f(S)$.

– Pour l'ensemble des travaux ordonnancés en retard, la nouvelle contribution au coût des travaux d'un même type est : $W^{(2)}_{r,r'}(S^{(2)}_{r,r'} - d) + r \sum_{i \in T^{(2)}_{r,r'}} \sum_{j \in T^{(2)}_{r,r'}} p_i p_j$. Si $f^{(2)}_{r,r',\varepsilon}(S) \geq \varepsilon$, comme pour le cas des travaux ordonnancés en avance, nous avons :

$$\begin{aligned} r \sum_{i \in T^{(2)}_{r,r'}} \sum_{j \in T^{(2)}_{r,r'}} p_i p_j &\geq r' \sum_{i \in TF^{(2)}_{r,r',\varepsilon^4}(S)} \sum_{j \in TF^{(2)}_{r,r',\varepsilon^4}(S)} p_i p_j \\ &\geq (1/2) r' \Big(\sum_{i \in TF^{(1)}_{r,r',\varepsilon^4}(S)} p_i \Big)^2 \\ &\geq (1/2) r' \varepsilon^2 P^2_{r,r',\varepsilon} \geq (1/\varepsilon) 9 r' \varepsilon^4 P^2_{r,r',\varepsilon}. \end{aligned}$$

Si $f^{(2)}_{r,r'}(S) < \varepsilon$, trois cas sont à examiner. Nous montrons donc :

1. s'il n'existe aucun travail J_i avec $p_i \geq \varepsilon P_{r,r'}$, alors $P_{r,r',\varepsilon} = P_{r,r'}$. De ce fait, pour une valeur $\varepsilon \leq 1/2$, il existe au moins deux travaux ordonnancés en avance. Tous les petits travaux en retard sont déplacés pour être ordonnancés en avance juste avant ceux du même type. Pour la nouvelle solution, les coûts d'avances les poids sont augmentés d'un facteur borné par $(1 + \varepsilon/(1-\varepsilon))$ et les poids sont augmentés d'un facteur borné par $(1 + 2\varepsilon/(1-2\varepsilon))$ (la preuve est obtenue de la même façon que celle de la proposition 3.4.3) ;

2. s'il existe un travail J_i ordonnancé en avance avec $p_i \geq \varepsilon P_{r,r'}$, tous les petits travaux en retard sont alors déplacés pour être ordonnancés en avance juste avant ceux du même type. Ainsi, les coûts d'avances et les poids sont augmentés d'un facteur borné par : $(1 + \varepsilon/(1-\varepsilon))$;

3. s'il existe un travail ordonnancé en retard J_i avec $p_i \geq \varepsilon P_{r,r'}$, alors la pénalité de retard de ce travail est supérieure ou égale à $\beta_i p_i = r' p_i^2 \geq r' \varepsilon^2 P_{r,r'}^2 \geq r' \varepsilon^2 P_{r,r',\varepsilon}^2 \geq (1/\varepsilon) 9 r' \varepsilon^4 P_{r,r',\varepsilon}^2$.

Par conséquent, avec une performance garantie en $O(\varepsilon)$, le nombre de travaux de chaque type $T_{r,r'}$ est borné par $1/\varepsilon + 1/\varepsilon^4 \leq 2/\varepsilon^4$. □

Borner les affectations des travaux non-décidés

Définition 3.4.3 *Supposons que $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) représente l'ensemble des travaux en avance (resp. en retard). Nous considérons deux matrices à deux dimensions $\mathcal{Q}_l$, $_{l=1,2}$ où chacune correspond à l'ensemble des travaux $\mathcal{I}_l$ défini comme suit :*

$$\forall i,j \in \mathcal{J} \ : \mathcal{Q}_1[i,j] = \begin{cases} \frac{1}{2} p_i \alpha_j & si\ i \neq j\ et\ \frac{p_i}{\alpha_i} = \frac{p_j}{\alpha_j} \\ p_i \alpha_j & si\ \frac{p_i}{\alpha_i} < \frac{p_j}{\alpha_j} \\ 0 & sinon \end{cases}$$

$$\forall i,j \in \mathcal{J} \ : \mathcal{Q}_2[i,j] = \begin{cases} p_i \beta_j & si\ i = j \\ \frac{1}{2} p_i \beta_j & if\ i \neq j\ et\ \frac{p_i}{\beta_i} = \frac{p_j}{\beta_j} \\ p_i \beta_j & si\ \frac{p_i}{\beta_i} < \frac{p_j}{\beta_j} \\ 0 & sinon \end{cases}$$

L'élément $\mathcal{Q}_1[i,j]$ (resp. $\mathcal{Q}_2[i,j]$) représente la contribution du travail J_i sur le coût d'avance (resp. retard) du travail J_j qui correspond à la valeur de $\alpha_j * p_i$ (resp. $\beta_j * p_i$).

La fonction du coût d'avance-retard des travaux pour une séquence S respectant les propriétés précédentes, peut donc être reformulée comme suit :

$$f(S) = \sum_{C_i \leq d, C_j \leq d} \mathcal{Q}_1[i,j] + \sum_{C_i > d, C_j > d} \mathcal{Q}_2[i,j]$$

Ainsi, d'après cette fonction de coût, nous pouvons supposer que si les ratios $(\alpha_i / p_i, \beta_i / p_i)$ de deux travaux sont significativement différents, leur interaction peut donc être négligée. D'où la proposition suivante

Proposition 3.4.6 *Pour chaque couple de travaux (J_i, J_j) tel que $r_j^{(l)} \leq \varepsilon^2 r_j^{(l)}$ avec $l = 1,2$, nous remplaçons $\mathcal{Q}_l[i,j]$ par 0. Cette modification diminue le coût global $f(S)$ d'au plus $4\varepsilon f(S)$.*

Preuve. Considérons d'abord l'ensemble I_1 (travaux en avance). A partir de S, nous déterminons les couples $(J_i, J_j) \in \mathcal{I}_1 \times \mathcal{I}_1$ ayant les propriétés suivantes : $r_i^{(1)} \leq \varepsilon^2 r_j^{(1)}$.

Considérons d'abord les couples (J_i, J_j) dont le poids cumulé satisfait $W_j^{(1)} < \varepsilon W_i^{(1)}$, le travail J_j est ordonnancé avant le travail J_i. Dans ce cas, nous disons que le travail J_j a un effet marginal sur le poids cumulé de J_i. Nous avons donc :

$$\begin{aligned}\sum_{i,C_i\le d}\sum_{j,C_j\le d\wedge W_j^{(1)}<\varepsilon W_i^{(1)}} \mathcal{Q}_1[i,j] &= \sum_{i,C_i\le d} p_i \sum_{j,C_j\le d\wedge W_j^{(1)}<\varepsilon W_i^{(1)}} \alpha_j \\ &\le \sum_{i,C_i\le d} p_i \max_{j,C_j\le d\wedge W_j^{(1)}<\varepsilon W_i^{(1)}} W_j^{(1)} \le \sum_{i,C_i\le d} p_i W_i^{(1)} \le \varepsilon f(S)\end{aligned}$$

Examinons maintenant les couples (J_i, J_j) dont le poids cumulé satisfait $W_j^{(1)} \ge \varepsilon W_i^{(1)}$. Pour ce type de couples, nous avons :

$$\begin{aligned}\sum_{i,C_i\le d}\sum_{j,C_j\le d\wedge W_j^{(1)}\ge\varepsilon W_i^{(1)})} \mathcal{Q}_1[i,j] &= \sum_{i,C_i\le d} p_i \sum_{j,C_j\le d\wedge W_j^{(1)}\ge\varepsilon W_i^{(1)}} \alpha_j = \sum_{i,C_i\le d} r_i^{(1)}\alpha_i \sum_{j,C_j\le d\wedge W_j^{(1)}\ge\varepsilon W_i^{(1)}} \alpha_j \\ &\le \varepsilon^2 \sum_{i,C_i\le d}\sum_{j,C_j\le d\wedge W_j^{(1)}\ge\varepsilon W_i^{(1)}} r_j^{(1)}\alpha_i\alpha_j \le \varepsilon^2 \sum_{i,C_i\le d}\sum_{j,C_j\le d\wedge W_j^{(1)}\ge\varepsilon W_i^{(1)}} p_j\alpha_i \\ &\le \varepsilon^2 \sum_{j,C_j\le d} p_j \sum_{i,C_i\le d\wedge W_i^{(1)}\le W_j^{(1)}/\varepsilon} \alpha_i \le \varepsilon^2 \sum_{j,C_j\le d} p_j \max_{i,C_i\le d\wedge W_i^{(1)}\le W_j^{(1)}/\varepsilon} W_i^{(1)} \\ &\le \varepsilon^2 \sum_{j,C_j\le d} p_j W_j^{(1)}/\varepsilon \le \varepsilon f(S)\end{aligned}$$

En conclusion, le coût total sera diminué d'au maximum de $2\epsilon C(S)$. De la même façon, en considérant l'ensemble I_2, on peut montrer que le coût total peut être diminué d'au maximum de $2\epsilon C(S)$.

Proposition 3.4.7 *Pour chaque couple de travaux (J_i, J_j) avec :*

1. $(J_i, J_j) \in \mathcal{I}_1 \times \mathcal{I}_1$ *avec* ($\alpha_i < \varepsilon\beta_i$ *et* $\alpha_j < \varepsilon\beta_j$) *ou* ($\varepsilon\beta_i \le \alpha_i \le \frac{\beta_i}{\varepsilon}$ *et* $\varepsilon\beta_j \le \alpha_j \le \frac{\beta_j}{\varepsilon}$) *et* $r_i^{(l)} < \varepsilon^4 r_j^{(l)}$

   OU

2. $(J_i, J_j) \in \mathcal{I}_2 \times \mathcal{I}_2$ *avec* ($\beta_i < \varepsilon\alpha_i$ *et* $\beta_j < \varepsilon\alpha_j$) *ou* ($\varepsilon\beta_i \le \alpha_i \le \frac{\beta_i}{\varepsilon}$ *et* $\varepsilon\beta_j \le \alpha_j \le \frac{\beta_j}{\varepsilon}$) *et* $r_i^{(l)} < \varepsilon^4 r_j^{(l)}$,

on remplace les deux éléments $\mathcal{Q}_1[i,j]$ et $\mathcal{Q}_2[i,j]$ par 0. Le coût global $f(S)$ est diminué d'au plus de $2\varepsilon C(S)$.

Proof. Pour le cas où $l = 1$ (resp. $l = 2$), selon la proposition 3.4.6, en remplaçant $\mathcal{Q}_1[i,j]$ (resp. $\mathcal{Q}_2[i,j]$) par 0, $f(S)$ sera diminué d'au plus de $4\varepsilon C(S)$.

Pour un couple de travaux (J_i, J_j), si l'un des deux est décidé en avance, i.e. ordonnancé avant la date d, nous posons donc $\mathcal{Q}_2[i,j] = 0$.

Rappelons qu'un travail ne peut être que dans l'un de ces trois états : décidé en avance, décidé en retard et non-décidé (J_i est dans l'état non-décidé si : $\varepsilon\beta_i \leq \alpha_i \leq \beta_i/\varepsilon \Leftrightarrow \varepsilon\alpha_i \leq \beta \leq \alpha_i/\varepsilon$).

Dans le cas où le couple de travaux (J_i, J_j) est dans l'état non-décidé, nous avons alors : $\beta_i \leq \alpha_i/\varepsilon$ et $\alpha_j \leq \beta_j/\varepsilon$. Par conséquent :

$$r_i^{(2)} = \frac{\beta_i}{p_i} \leq \frac{1}{\varepsilon}\frac{\alpha_i}{p_i} \leq \frac{\varepsilon^4}{\varepsilon}\frac{\alpha_j}{p_j} \leq \frac{\varepsilon^4}{\varepsilon}\frac{1}{\varepsilon}\frac{\beta_j}{p_j} \leq \varepsilon^2 r_j^{(2)}$$

Selon la proposition 3.4.6, $\mathcal{Q}_2[i,j]$ est remplacé par 0.

Définition 3.4.4 *Soit R le ratio maximum des $\{\alpha_i/p_i, \beta_i/p_i | i\}$. Soit $\mathcal{F}_k$ la k-ème fenêtre des ratios pour laquelle il existe un travail où au minimum un de ses deux ratios est dans l'intervalle $(a^k\mathcal{R}, a^{k-1}\mathcal{R}]$ avec $a = \varepsilon^4$. $\mathcal{F}_k$ est dite une fenêtre active si et seulement si elle contient au minimum un ratio.*

Remarquons que :

1. Pour un travail non-décidé J_i, $r_i^{(1)}$ et $r_i^{(2)}$ sont dans la même fenêtre des ratios ou au maximum dans deux fenêtres adjacentes
2. Pour chaque couple (J_i, J_j) non pris en compte par la proposition 3.4.7 :
   - $r_i^{(1)}$ et $r_j^{(1)}$ sont dans la même fenêtre ou dans deux fenêtres adjacentes
   - $r_i^{(2)}$ et $r_j^{(2)}$ sont dans la même fenêtre ou dans deux fenêtres adjacentes

Proposition 3.4.8 *Le nombre des travaux non-décidés ayant au moins un ratio dans la fenêtre $\mathcal{F}_k$ est borné par $O(log^2(1/\varepsilon)/\varepsilon^2)$.*

Proof. Selon la proposition 3.4.1, pour chaque fenêtre $\mathcal{F}_i$, nous avons au maximum $\lceil log_{1+\varepsilon}(1+a) \rceil$ ratios différents. En considérant la définition 3.4.1 obtenue à partir de la proposition 3.4.2, un travail non-décidé ayant un seul ratio dans $\mathcal{F}_k$, son deuxième ratio serait donc dans l'intervalle $(a^k R\varepsilon, a^{k-1}R/\varepsilon]$. Par conséquent, le nombre total de types de travaux ($T_{r,r'}$) est au maximum : $\lceil log_{1+\varepsilon}(1/a\varepsilon^2) \rceil$. Selon la proposition 3.4.5, la taille de type de travaux est bornée par $2/\varepsilon^4$. Ainsi, le nombre des travaux non-décidés ayant au moins un ratio dans la fenêtre $\mathcal{F}_k$ est au plus $O(\lceil (log_{1+\varepsilon}(1/a) log_{1+\varepsilon}(1/a\varepsilon^2))/\varepsilon^4 \rceil)$, bornée par $O(log^2(1/\varepsilon)/\varepsilon^4)$.

En conclusion, nous déduisons le programme dynamique suivant.

Programme dynamique

1. Simplifier les données du problème et concaténer les petits travaux pour former des grands travaux.
2. En supposant qu'au plus un des travaux décidés en retard sera exécuté en avance, faire :
   (a) Pour un travail J_i décidé en avance (resp. en retard), affecter la valeur 0 aux éléments $\mathcal{Q}_2$ (resp. $\mathcal{Q}_1$) correspondants
   (b) À l'itération k, considérer l'ensemble des travaux ayant au moins un ratio dans une fenêtre active $\mathcal{F}_k$:
      i. Effectuer au plus $2^{O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)}$ affectations pour l'ensemble des travaux non-décidés.
      ii. Pour chacune de ces affectations, déterminer la meilleure décision parmi les $2^{O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)}$ obtenues à l'itération précédente, compatible avec l'affectation en question.

Preuve Rappelons que le premier travail exécuté dans la solution optimale S est noté par $J_{[1]}$. Selon la proposition 3.4.2, les travaux décidés en avance peuvent être ordonnancés en avance et les travaux décidés en retard sauf $J_{[1]}$ peuvent être ordonnancés en retard sous l'hypothèse $\alpha_{min} \leq \alpha_i \leq \alpha_{max}$ et $\alpha_{max}/\alpha_{min} \leq 2$. Pour vérifier si $J_{[1]}$ est du type décidé en retard, nous énumérons tous les cas possibles : soit tous les travaux décidés en retard sont exécutés en retard, soit un travail décidé en retard est exécuté en avance.

Selon la proposition 3.4.7, prendre une décision pour le travail J_i ne dépend pas d'une décision prise pour J_j si les ratios de J_i et J_j ne sont pas dans la même fenêtre des ratios ni dans les fenêtres de ratios adjacentes. Par conséquent, cette procédure permet de déterminer une solution avec une garantie de performance bornée par $O(\varepsilon)$.

Puisqu'il existe au plus $2n$ fenêtres de ratios actives, la complexité de l'algorithme est donc bornée par $O_\varepsilon(n^2)$, plus précisément de $O(n^2 2^{O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)})$.

Théorème 3.4.9 *Le problème d'ordonnancement* $1|d_i = d, non-restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ *accepte un PTAS en* $O_\varepsilon(n^2)$ *sous l'hypothèse* $\alpha_{min} \leq \alpha_i \leq \alpha_{max}, \forall i = 1 \ldots n$ *et* $\alpha_{max}/\alpha_{min} \leq 2$.

3.4.2 PTAS pour le cas de m machines parallèles identiques

Avec une date de fin souhaitée non-restrictive, le problème considéré est traditionnellement dénoté par $Pm|d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$.

Présentons d'abord quelques propriétés qui peuvent être montrées sans difficulté particulière.

3.4.2.1 Propriétés

Propriété 3.4.10 *Il n'y a pas de temps mort entre deux travaux adjacents.*

Propriété 3.4.11 *Sur chaque machine, la séquence optimale est sous la forme "V-shape" autour de la date de fin souhaitée.*

Propriété 3.4.12 *Il existe une solution optimale où exactement un travail juste-à-temps est exécuté sur chaque machine.*

3.4.2.2 PTAS pour le cas de m machines identiques

En suivant la démarche présentée pour le cas d'une seule machine, nous pouvons montrer les propositions suivantes :

- Le nombre des travaux ayant au moins un ratio dans la fenêtre $\mathcal{F}_k$ est borné par $O(\log_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)$.
- On doit déterminer l'affectation des travaux décidés et non-décidés aux différentes machines.
- La proposition 3.4.5 reste valide, c-à-d la taille de chaque type de travail $T_{r,r'}$ est d'au plus $2/\varepsilon^4$ éléments.
- Pour une fenêtre des ratios active F_k, nous avons au plus $\lceil \log_{1+\varepsilon}(1/a) \rceil$ ratios différents.

Par conséquent, l'ensemble des travaux décidés en avance ayant un ratio dans $\mathcal{F}_r$ est borné par $O(\log_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)$.

Donc chaque ensemble de travaux décidés en avance, en retard et non-décidés ayant un ratio dans $\mathcal{F}_r$, est borné par $O(\max(\log_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4); \log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4))$ $= O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)$.

Ainsi, nous pouvons proposer le programme dynamique suivant :

Programme dynamique

1. Simplifier les données du problème et concaténer les petits travaux pour former des grands travaux.
2. En supposant que sur chaque machine au plus un des travaux décidés en retard sera exécuté en avance, faire :
   (a) Pour un travail J_i décidé en avance (resp. en retard), affecter la valeur 0 aux éléments $\mathcal{Q}_2$ (resp. $\mathcal{Q}_1$) correspondants
   (b) À l'itération k, considérer l'ensemble des travaux ayant au moins un ratio dans une fenêtre active $\mathcal{F}_k$:
      i. Effectuer au plus $(2m)^{O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)}$ affectations pour l'ensemble des travaux non-décidés en avance/retard.
      ii. Pour chacune de ces affectations, déterminer la meilleure décision parmi les $(2m)^{O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)}$ obtenues à l'itération précédente, compatible avec l'affectation en question.

Le nombre de cas possible est borné par n^m pour que sur chaque machine, soit tous les travaux décidés en retard seront exécutés en retard, soit un travail décidé en retard sera exécuté en avance. Puisqu'il existe au plus $2n$ fenêtres de ratios actives, la complexité de l'algorithme est donc bornée par $O_\varepsilon(n^{m+1})$, plus précisément de $O(n^{m+1}(2m)^{O(\log^2_{1+\varepsilon}(1/\varepsilon)/\varepsilon^4)})$.

Théorème 3.4.13 *Le problème d'ordonnancement* $Pm|d_i = d, non-restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ *accepte un PTAS en* $O_\varepsilon(n^{m+1})$ *sous l'hypothèse* $\alpha_{min} \leq \alpha_i \leq \alpha_{max}, \forall i = 1 \ldots n$ *et* $\alpha_{max}/\alpha_{min} \leq 2$.

Conclusion

Dans ce chapitre, quelques problèmes d'ordonnancement de type JàT avec dates de fin souhaitées données ont été abordés. Comme nous l'avons vu, pour la résolution du cas particulier $1|p_i = p, d_i = d|\sum(\alpha_i E_i + \beta_i T_i)$, le problème est décomposé en deux sous-problèmes où chacun n'est autre qu'un problème d'affectation. Nous nous sommes posé la question sur la limite d'une telle approche pour résoudre les problèmes d'ordonnancement de type JàT. Nous sommes parvenus aux résultats suivants :

– $1|p_i = p, d_i \in D, |D| \leq \ell|\sum(\alpha_i E_i + \beta_i T_i)$ peut être résolu en $O(l^{l+1}n^{l+2})$

– $Qm|p_i = p, d_i \in D, |D| \leq \ell| \sum(\alpha_i E_i + \beta_i T_i)$ peut être résolu en $O(m2^m l^{ml+1} n^{m(l-1)+3})$

Pour le cas général, avec durées opératoires quelconques et une date de fin souhaitée commune non-restrictive, nous avons montré que les problèmes suivants :

– $1|d_i = d, non - restrictive \mid \sum(\alpha_i E_i + \beta_i T_i)$

– $Pm|d_i = d, non - restrictive \mid \sum(\alpha_i E_i + \beta_i T_i)$

admettent un schéma d'approximation en temps polynomial (PTAS) sous l'hypothèse $\alpha_{min} \leq \alpha_i \leq \alpha_{max}, \forall i = 1 \ldots n$ et $\alpha_{max}/\alpha_{min} \leq k$ où k est une constante.

Il serait donc intéressant dans l'avenir, de programmer le PTAS pour étudier ses performances (qualité de la solution par rapport au temps de calcul). Néanmoins, une autre question demeure toujours ouverte : existe-t-il un FPTAS ou PTAS pour le cas de date de fin souhaitée restrictive ?

Chapitre 4

Avance et retard pondérés avec dates de fin contrôlables

Dans ce chapitre, nous allons étudier quelques problèmes JàT lorsque les travaux ont une date de fin souhaitée contrôlable. La date de fin souhaitée est donc une variable de décision. L'objectif ici est de calculer un ordonnancement avec une solution de meilleur compromis entre les coûts d'avance, les coûts de retards et le coût lié à la valeur de la date de fin commune souhaitée à déterminer. On parle de problèmes d'ordonnancement JàT avec "affectation de date de fin souhaitée". Dans ce chapitre, n travaux sont à ordonnancer sur une seule machine puis sur m machines parallèles. Pour le cas de durées opératoires identiques, l'ordonnancement des travaux sur m machines identiques ou uniformes est montré polynomial, même s'il s'agit de ℓ dates de fin souhaitées contrôlables $(\ell < n)$. Pour le cas général, nous étudions d'abord l'équivalence entre ce type de problème et le cas de date de fin souhaitée non-restrictive. A la fin de ce chapitre, une borne inférieure basée sur la relaxation Lagrangienne est proposée.

4.1 Introduction

Dans ce chapitre, on s'intéresse aux critères d'avance-retard, encore appelés critères de "juste à temps". L'avance est mesurée par rapport aux souhaits des clients pour la partie aval (dates de fin souhaitées). Il existe deux types de dates de fin au plus tard, selon qu'elles sont "souhaitées" (comme nous l'avons vu dans les chapitres précédents) ou"impératives", selon les souhaits du client. Mais si on regarde un atelier de production de près, il arrive qu'on parle d'un troisième type de date de fin souhaitée, appelé "date de fin pour l'atelier" [1]. Cette dernière date de fin peut correspondre à une certaine marge qui permet de compenser les aléas de production d'une part et le délai de livraison d'autre part. L'affectation des dates de fin souhaitées reflète en général une certaine coopération/interaction possible entre les divers acteurs d'une chaîne logistique : la pratique montre que pendant les négociations entre les clients et les producteurs, l'accord sur les dates de livraison dépendent de la politique de la production (contraint par la limitation des espaces de stockage et/ou la capacité de production). Par conséquent, le chef d'atelier aura besoin d'une méthode efficace pour déterminer à la fois les dates de fin souhaitées et l'ordonnancement de ses travaux ([36, 77, 126, 78, 128]).

Ces dates de fins peuvent être communes à tous les travaux. Dans le cas où la localisation de la date de fin est une variable de décision, le problème est dit "à dates de fin souhaitée (i.e. au plus tard) commune contrôlable". Le problème classique où l'ensemble des travaux doivent être exécutés sur une seule machine pour être finis au plus près d'une date de fin à déterminer, a été largement abordé dans la littérature spécialisée. On parle du problème "CON" *Constant Flow Allowance* [33, 36, 77].

Dans ce chapitre, nous abordons quelques problèmes de type JàT avec une famille de dates de fin souhaitées contrôlables. Chaque famille de travaux peut être considérée comme une commande d'un client, qui doit être livrée en une fois. Deux cas sont étudiés : le cas de travaux à durées opératoires identiques et le cas de travaux à durées opératoires quelconques. Le premier cas peut être rencontré dans les systèmes de production par lot, où la durée de chaque lot est identique, connue et fixe. Citons par exemple une société de conditionnement et d'emballage de bouteilles de shampooing. Chaque opération est définie par un lot de bouteilles de shampooing et pour alléger le processus de production, les lots sont constitués selon les opérations exigeant le même

[1] Voir une introduction présentée sur le web site www.ocea.li.univ-tours.fr/gdr-ro/JE_11h15_G.D.R. RO Ordo-Coo-Sc MCP.ppt

temps de traitement (le temps de préparation des machines est supposé négligeable par rapport au temps de traitement). Dans la société en question, la durée opératoire de chaque lot est d'une demi-heure.

En général, il s'agit de déterminer un ordonnancement où l'objectif est défini par une combinaison linéaire des trois facteurs de coûts suivants : coûts d'avance, coûts de retard et coût lié à la valeur de la date de fin souhaitée (plus la date est éloignée, plus le coût est élevé). On cherche un compromis pour avoir une petite date de fin souhaitée commune tout en minimisant l'avance et le retard des travaux.

D'après nos connaissances, ce type de problème a été étudié pour la première fois par Panwalkar et al. [169]. En effet, les auteurs proposent un algorithme efficace en $O(n \log n)$ pour résoudre le cas d'une seule machine avec pénalités identiques, noté $1|d_j = d|\sum(\alpha E_j + \beta T_j + \gamma d_j)$. Quant au cas de deux machines parallèles identiques, $P2|d_j = d|\sum(\alpha E_j + \beta T_j + \gamma d_j)$, le problème est montré $\mathcal{NP}$-difficile au sens ordinaire [40, 50]. Pour calculer un ordonnancement optimal, De et al proposent un algorithme de programmation dynamique en $O(n^{2m+1} p_{\max}^{2m})$ [50]. Cependant, si le nombre de machines n'est pas fixe, le problème $P|d_j = d|\sum(\alpha E_j + \beta T_j + \gamma d_j)$ est montré $\mathcal{NP}$-difficile au sens fort [50].

Comme méthodes approchées proposées pour la résolution du cas m machines parallèles identiques, Cheng [34] proposent une heuristique basée sur la règle V-shape. L'étude expérimentale montre que pour des instances de petites tailles, la déviation d'une solution obtenue par rapport à une solution optimale n'excède pas en moyenne 6%. En 2002, Xiao et Li [221] proposent un algorithme en $O(n \log n)$ à garantie de performance de 3 (3-approximation) et un schéma d'approximation de type FPTAS.

Le cas de durées opératoires identiques a été étudié. Le cas de machines parallèles identiques a été étudié par Cheng et Chen [40]. Les auteurs montrent que le problème $Pm|p_i = p, d_i = d|\sum(\alpha E_i + \beta T_i + \gamma d_i)$ est polynomial. Quant au cas machines parallèles uniformes, Mosheiov et Sarig [164] montrent que le problème à deux machines, $Q2|p_i = p, d_i = d|\sum(\alpha E_i + \beta T_i + \gamma d)$, peut être résolu en temps constant. Par la suite, les auteurs proposent de généraliser leur approche au cas de m machines ; le temps de calcul de la méthode proposée est de $O(3^m)$. D'après Mosheiov et Sarig, cette méthode ne permet de résoudre que des instances allant jusqu'à 9 machines en un temps raisonnable. En effet, le temps de calcul augmente exponentiellement avec le nombre de machines. Néanmoins, dans [165], les auteurs montrent qu'une solution optimale pour

le cas de m machines parallèles identiques avec durées opératoires identiques peut être obtenu en $O(n^4)$ si la date de fin souhaitée est une variable continue. Pour le cas de machines parallèles générales, Adamopoulos et Pappis [2] proposent une heuristique où les résultats expérimentaux montrent que la déviation d'une solution obtenue par leur heuristique par rapport à une solution optimale est inférieure à 9%. Cependant, ces résultats expérimentaux ont été effectués sur des instances de petites tailles.

Beaucoup de résultats ont été dédiés au cas $\gamma = 0$, c-à-d. le cas de date de fin souhaitée non-restrictive (cf. Chapitre 3). Il s'agit donc d'un cas particulier du problème étudié dans ce chapitre. Certains auteurs définissent ce dernier cas comme un problème avec date de fin souhaitée à déterminer [33, 61, 35, 48, 142, 9, 89]

En effet, Emmons [61] propose un algorithme en $O(n \log n)$ pour résoudre le problème $Q|d_i = d| \sum(\alpha E_i + \beta T_i)$. La recherche d'une solution optimale pour le cas de machines parallèles quelconques $R|d_i = d| \sum(E_i + T_i)$ est montré polynomial [142, 9]. En effet, les auteurs montrent qu'il s'agit d'un problème de transport. Ainsi, une solution optimale peut être obtenue en $O(n^3)$.

Pour le cas symétrique ($\alpha_i = \beta_i$) à une machine, le problème $1|d_i = d, non - restrictive| \sum w_i(E_i + T_i)$, Cheng [33] identifient des règles de dominance et développe une heuristique efficace pour les instances de petites tailles. L'autre propose dans [35] une amélioration de son heuristique où le temps de calcul est borné par $O(n^4)$. Cependant, De et al [48] proposent un programme dynamique (ProgDyn). Les auteurs montrent que l'heuristique basée sur leur ProgDyn, permet de résoudre efficacement des instances allant jusqu'à 100 travaux en un temps n'excédant pas les 50 secondes. Cependant, cette heuristique n'est efficace que si les durées opératoires et les poids sont agréables. Pour résoudre le problème avec des instances de grande taille, Hao et al [89] proposent une recherche tabou. Les résultats obtenus par cette métaheuristique sont en général meilleures que celles obtenues en utilisant les approches citées précédemment [33, 48].

Pour déterminer des bornes supérieures pour le problème à une seule machine avec pénalités d'avance-retard quelconques, $1|d_i = d| \sum(\alpha_i E_i + \beta_i T_i)$, les auteurs se basent en général sur des algorithmes génétiques [152], sur des techniques de génération de colonnes et de relaxation Lagrangienne [54], des modélisations mathématiques [181, 19] en bornant le temps de calcul, ou encore des algorithmes de listes [83, 57, 51].

Cependant, nous avons trouvé très peu de résultats consacrés au cas d'une famille

de dates de fin souhaitées. Citons par exemple, les travaux de Dickman et al. [55] pour le cas de machines parallèles identiques, noté $Pm|d_i \in D, |D| = \ell| \sum(E_i + T_i)$ avec un nombre donné de dates de fin souhaitées ($\ell \geq 1$). Notons que ce dernier problème est simple puisqu'aucune pénalité n'est imposée sur les dates de fin souhaitées. Le cas de dates de fin souhaitées contrôlables générales, noté par $1|d_i| \sum(\alpha E_i + \beta T_i + \gamma d_i)$, a été montré polynomial par Seidmann et al, où une solution optimale peut être obtenue en $O(n \log n)$ [191].

Les principaux résultats connus sont présentés dans le tableau 4.1.

Problème	Méthode & Complexité	Reference
$1\|d_i = d\| \sum(\alpha E_i + \beta T_i + \gamma d)$	Polynomial $O(n \log n)$	[169]
$1\|\| \sum(\alpha E_i + \beta T_i + \gamma \max(0; d_i - d)$	Polynomial $O(n \log n)$	[191]
$1\|d_i = d\| \sum(\alpha E_i + \beta T_i)$	Polynomial $O(n \log n)$	[17]
$P\|p_i = p, d_i = d\| \sum(\alpha E_i + \beta T_i + \gamma d)$	Polynomial	[40]
$Pm\|d_i = d\| \sum(\alpha E_i + \beta T_i + \gamma d)$	NP-difficile au sens ordinaire	[40, 50]
	PD $O(n^{2m+1} p_{\max}^{2m})$	[50]
	3-approximation en $O(n \log n)$	[221]
	FPTAS	[221]
$P\|d_i = d\| \sum(\alpha E_i + \beta T_i + \gamma d)$	NP-difficile au sens fort	[50]
$P\|d_i = d\| \sum w_i(E_i + T_i)$	NP-difficile au sens fort	[216]
$Qm\|p_i = p, d_i = d\| \sum(\alpha E_i + \beta T_i + \gamma d)$	Polynomial $O(3^m)$	[40]
$Q\|d_i = d\| \sum(\alpha E_i + \beta T_i)$	Polynomial $O(n \log n)$	[61]
$R\|d_i = d\| \sum(E_i + T_i)$	Polynomial $O(n^3)$	[142, 9]

TAB. 4.1 – Problèmes JàT avec affectation de la date de fin souhaitée

4.1.1 Quelques notations spécifiques à ce chapitre

Dans ce chapitre, nous supposons que les travaux et les machines sont disponibles. Les travaux doivent être exécutés, au plus près d'une date de fin souhaitée à déterminer. Les travaux forment ℓ sous-groupes fixes et connus. Chaque sous-groupe partage la même date de fin souhaitée. Nous avons donc ℓ dates dues communes notées $\{D_1, D_2, \ldots, D_\ell\}$. La particularité des problèmes traités par rapport aux problèmes du chapitre précédent, est que la solution recherchée doit nous permettre de minimiser à la fois la somme pondérée des avances et retards des travaux et les coûts pondérés relatifs aux valeurs des dates de fin souhaitées. Autrement dit, on cherche le meilleur compromis entre l'ordonnancement des travaux autour de leur date de fin souhaitée et leur emplacement dans le temps pour que cette date soit le plus proche de 0. Quelques nouvelles notations suivantes sont utilisées dans ce chapitre.

- $\mathcal{N}_k$: les travaux partageant la même date due $D_k, k = 1, \ldots, \ell$;
- $n_k = |\mathcal{N}_k|$: le nombre de travaux pour une date $D_k, k = 1, \ldots, \ell$;
- D : l'ensemble des ℓ dates de fins souhaitées ;
- d_i : date due du travail J_i, $i = 1, 2, \ldots, n$;
- γ_i : le coût unitaire de pénalité de la date due $d_i \in D$;
- $\gamma(\mathcal{N}_k)$: le coût unitaire moyen de pénalité par rapport à $D_k \in D$, donnée par $\sum_{i \in \mathcal{N}_k}(\gamma_i / n_k)$;
- λ_i : la performance de la machine M_i. Dans le cas de machines uniformes, on suppose que les machines sont rangées selon l'ordre non-croissant de leur performance (M_1 est la plus rapide et M_m est la plus lente) ;
- X_i : temps mort initial sur la machine M_i (i.e. entre la date zéro et la date du début d'exécution du premier travail ordonnancé sur M_i) ;
- pX_i : durée du temps mort initial sur la machine M_i ;
- y_i^e : le nombre des travaux exécutés sur M_i dans l'intervalle $[0, d]$;
- y_i^t : le nombre des travaux exécutés sur M_i dans l'intervalle $(d, +\infty)$;
- $Z = f(S) = \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$: l'objectif à minimiser.

Comme pour le chapitre précédent, un travail peut se trouver dans l'une des trois situations suivantes :

- J_i est en avance i.e. $C_i \leq d_i$, (J_i est dit juste-à-temps si $C_i = d_i$) ;
- J_i est un splitting job i.e. $S_i \leq d_i < C_i$ (sous-entendu, le premier travail en retard),
- J_i est totalement en retard i.e. $d_i < C_i$.

Remarque 4.1.1 *Dans la suite, s'il s'agit d'une seule date de fin souhaitée commune contrôlable, la pénalité liée à cette date est notée par :* $\gamma = \sum_i \gamma_i / n$.

Selon la notation classique en trois champs, les problèmes abordés dans ce chapitre sont notés comme suit :

1. Durées opératoires identiques ($p_i = p, \forall i, i = 1, \ldots, n$)
   - $1|p_i = p, d_i = d| \sum(\alpha_i E_i + \beta_i T_i + \gamma d_i)$ (cf. section 4.2.1)
   - $1|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ (cf. section 4.2.1.1)
   - $Pm|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ (cf. section 4.2.2)
   - $Qm|p_i = p, d_i = d| \sum(\alpha E_i + \beta T_i + \gamma d_i)$ (cf. section 4.2.3)
2. Durées opératoires quelconques
   - $1|d_i = d| \sum(\alpha_i E_i + \beta_i T_i + \gamma d_i)$ (cf. section 4.3.1)

– $Pm|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d_i)$ (cf. section 4.3.2)

4.2 Durées opératoires identiques

Dans cette section, selon les valeurs des α_i et β_i deux types de problèmes JàT sont abordés : le cas identique et le cas quelconque pour l'ordonnancement des travaux sur une seule machine et m machines parallèles. Pour le cas quelconque avec une seule date de fin contrôlable, nous proposons une méthode de résolution permettant d'obtenir une solution optimale [108] avec une meilleure complexité que celle proposée par [165] (en $O(n^3)$ au lieu de $O(n^4)$). Par la suite, nous montrons comment une solution optimale peut être déterminée en temps polynomial pour le cas de m machines parallèles identiques avec ℓ dates de fin souhaitées contrôlables [108]. Le cas de pénalités d'avance-retard identique sera abordé à la fin de cette section. En effet, même si le problème $Qm|p_i = p, d_i = d|\sum(\alpha E_i + \beta T_i + \gamma d_i)$ a été montré polynomial et où une solution optimale peut être obtenue en $O(3^m)$ [164], nous montrons qu'il est possible de déterminer une solution optimale en $O(m^3)$ [107]. L'intérêt particulier de ce travail est de montrer que si le nombre de machines m n'est pas donné, le problème $Q|p_i = p, d_i = d|\sum(\alpha E_i + \beta T_i + \gamma d_i)$ est polynomial.

4.2.1 Une seule machine avec une seule date de fin souhaitée

Le problème abordé dans cette section se note $1|p_i = p, d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d_i)$. Rappelons d'abord quelques propositions connues [77].

Proposition 4.2.1 *Il existe une solution optimale où :*

1. *le premier travail commence à la date zéro et il n'y a pas de temps mort intermédiaire entre deux travaux adjacents quelconques (aucun temps mort dans une séquence optimale) [20],*
2. *l'ordonnancement des travaux respecte le "V-shape" [195],*
3. *si $n\gamma \geq \sum_{j=1}^{n} \beta_j$, alors $d = 0$; sinon, il existe un travail J_i juste-à-temps [169].*

Pour déterminer une solution optimale, Mosheiov et Yovel [165] proposent un algorithme en $O(n^4)$ où l'idée est la suivante : une valeur optimale de la date due d correspond à une des valeurs de l'ensemble $\{0, p, 2p, \ldots, np\}$. Il s'agit donc d'énumérer toutes les valeurs possibles de d. Pour chaque valeur, on résout un problème d'affectation en $O(n^3)$. Une solution optimale correspond donc à la meilleure solution parmi les $(n+1)$ solutions obtenues. Le temps de calcul total est donc borné par $O(n^4)$ [165].

Cependant, pour ce problème particulier, le temps de calcul peut être amélioré.

Théorème 4.2.2 *Une solution optimale du problème d'ordonnancement* $1|p_i = p, d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$ *peut être obtenue en* $O(n^3)$.

Preuve. Afin d'améliorer la complexité de l'algorithme de résolution, nous montrons qu'il n'est pas nécessaire d'énumérer toutes les valeurs possibles de la date de fin souhaitée. En effet, il suffit d'ajouter le coût lié à la valeur de la date de fin souhaitée dans la partie des coûts d'avances des travaux. En effet, si on connaît l'ensemble des travaux ordonnancés en avance on peut donc déduire la valeur de d. Nous avons par conséquent : $d = \sum_{i;C_i \leq d} p$. Le coût par unité de temps de d correspond donc à la valeur $n\gamma$. Ainsi, on peut considérer que la contribution de la valeur de la date due au coût d'avance de chaque travail est de $n\gamma p$. Autrement dit, la fonction objectif peut être réécrite comme suit.

$$\begin{aligned}
\sum_{j=1}^{n}(\alpha_i E_i + \beta_i T_i + \gamma d) &= \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i) + n\gamma d \\
&= \sum_{j=1, C_i \leq d}^{n} \alpha_i E_i + \sum_{i=1, C_i > d}^{n} \beta_i T_i + n\gamma d \\
&= \sum_{j=1, C_i \leq d}^{n} \alpha_i E_i + \sum_{i=1, C_i > d}^{n} \beta_i T_i + n\gamma \sum_{i=1, C_i \leq d}^{n} p \\
&= \sum_{j=1, C_i \leq d}^{n} (\alpha_i E_i + np\gamma) + \sum_{i=1, C_i > d}^{n} (\beta_i T_i)
\end{aligned}$$

Par conséquent, pour obtenir une solution optimale du problème considéré, il suffit de déterminer l'ensemble des travaux en avance et ceux en retard, où les pénalités d'avance et retard sont définies comme suit :

- le coût d'avance d'un travail J_i ($C_i \leq d$) est $\alpha_i E_i + np\gamma$
- le coût de retard d'un travail J_i ($C_i > d$) est $\beta_i T_i$

Ainsi, pour trouver une solution optimale, nous pouvons construire une matrice d'affectation comme définie dans la Figure 4.1 : n lignes et $2n$ colonnes. Selon la proposition 4.2.1, la date de fin souhaitée optimale est un multiple de p. Ainsi, chaque élément de la matrice Q, qui correspond à un travail J_i et une position, est donnée par :

- $p\beta_i$, s'il s'agit de la position du "splitting job",
- $kp\alpha_i + np\gamma$, s'il s'agit de la k-ième position en avance,
- $(k-1)p\beta_i$, sinon.

Chercher une solution faisable avec un coût minimal à partir de la matrice d'affectation donne alors la position optimale de chaque travail et par la suite la valeur de la date de fin souhaitée optimale. La date de fin souhaitée optimale correspond donc au nombre de travaux en avance (k^*) facteur de p, i.e. $d^* = k^* \times p$.

		Travaux			
		J_1	J_2	...	J_n
Positions	J_s	$p\beta_1$	$p\beta_2$	...	$p\beta_n$
	J_{e_1}	$np\gamma$	$np\gamma$	...	$np\gamma$
	J_{t_1}	$2p\beta_1$	$2p\beta_2$	...	$2p\beta_n$
	J_{e_2}	$p\alpha_1 + np\gamma$	$p\alpha_2 + np\gamma$	...	$p\alpha_n + np\gamma$
	J_{t_2}	$3p\beta_1$	$3p\beta_2$	...	$3p\beta_n$
	J_{e_3}	$2p\alpha_1 + np\gamma$	$2p\alpha_2 + np\gamma$	...	$2p\alpha_n + np\gamma$
	J_{t_3}	$4p\beta_1$	$4p\beta_2$	...	$4p\beta_n$
	...	...	...	...	...
	...	...	...	...	...
	$J_{e_{n-1}}$	$(n-2)p\alpha_1 + np\gamma$	$(n-2)p\alpha_2 + np\gamma$	...	$(n-2)p\alpha_n + np\gamma$
	$J_{t_{n-1}}$	$np\beta_1$	$np\beta_2$	...	$np\beta_n$
	J_{e_n}	$(n-1)p\alpha_1 + np\gamma$	$(n-1)p\alpha_2 + np\gamma$	...	$(n-1)p\alpha_n + np\gamma$

FIG. 4.1 – La matrice d'affectation des travaux modifiée

Donc, pour déterminer une solution optimale, nous considérons seulement $\delta_1 = 0$ et $\delta_2 = p$ avec la matrice modifiée Q' de la Figure 4.1. Remarquons que cette méthode de résolution est seulement valable sur une seule machine avec une date de fin souhaitée commune.

Exemple.

Soient 5 travaux à ordonnancer. Les durées opératoires et les diverses pénalités sont présentées dans la table 4.2.

	J_1	J_2	J_3	J_4	J_5
p_i	4	4	4	4	4
α_i	7	5	4	3	5
β_i	5	2	7	4	5
γ	2	2	2	2	2

FIG. 4.2 – Durées opératoires et pénalités

Selon la matrice de coûts d'affectation des travaux (cf. figure 4.3), nous obtenons une valeur optimale de 172 qui correspond à la séquence $(J_3, J_1, J_5, J_4, J_2)$, avec J_3 le seul travail en avance et J_1 le travail JàT. La date due optimale est donc de 8 (Figure 4.4).

En suivant la démarche proposée par Mosheiov et Yovel [165], nous devons donc énumérer toutes les valeurs possibles de d. Ainsi, pour l'exemple choisi, on aura 5 ma-

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_s	20	8	28	16	**20**
J_{e_1}	**40**	40	40	40	40
J_{t_1}	40	16	56	**32**	40
J_{e_2}	68	60	**56**	52	60
J_{t_2}	60	**24**	84	48	60
J_{e_3}	96	80	72	64	80
J_{t_3}	80	32	112	64	80
J_{e_4}	124	100	88	76	100
J_{t_4}	100	40	140	80	100
J_{e_5}	152	120	104	88	120

FIG. 4.3 – Matrice des coûts d'affectation des 5 travaux

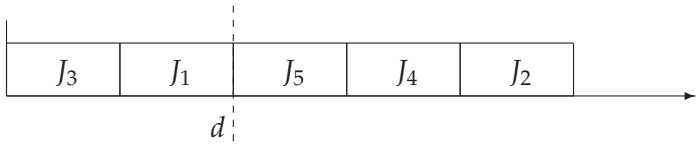

FIG. 4.4 – Solution optimale pour l'exemple choisi

trices d'affectations (cf. figure 4.5) :

1. $d = 0$: tous les travaux sont en retard. La valeur de la solution optimale est de 232 qui correspond à la séquence $(J_3, J_5, J_1, J_4, J_2)$.

2. $d = 1 \times p = 4$: la valeur de la solution optimale est donc de 180 qui correspond à la séquence $(J_3, J_5, J_1, J_4, J_2)$ (J_3 est le travail JàT).

3. $d = 2 \times p = 8$: la valeur de la solution optimale est donc de 172 qui correspond à la séquence $(J_3, J_1, J_5, J_4, J_2)$ (J_1 est le travail JàT).

4. $d = 3 \times p = 12$: la valeur de la solution optimale est donc de 196 qui correspond à la séquence $(J_4, J_3, J_1, J_5, J_2)$ (J_1 est le travail JàT).

5. $d = 4 \times p = 16$: la valeur de la solution optimale est donc de 256 qui correspond à la séquence $(J_4, J_3, J_1, J_5, J_2)$ (J_1 est le travail JàT).

6. $d = 5 \times p = 20$: la valeur de la solution optimale est donc de 356 qui correspond à la séquence $(J_4, J_3, J_2, J_5, J_1)$ (J_1 est le travail JàT).

Par conséquent, les deux algorithmes donnent la même valeur optimale.

Case $d = p$

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_s	20	8	28	16	**20**
J_{e_1}	0	0	**0**	0	0
J_{t_1}	**40**	16	56	32	40
J_{t_2}	60	24	84	**48**	60
J_{t_3}	80	**32**	112	64	80

Case $d = 2p$

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_s	20	8	28	16	**20**
J_{e_1}	**0**	0	0	0	0
J_{t_1}	40	16	56	**32**	40
J_{e_2}	28	20	**16**	12	20
J_{t_2}	60	**24**	84	48	60

Case $d = 3p$

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_s	20	8	28	16	**20**
J_{e_1}	**0**	0	0	0	0
J_{t_1}	40	**16**	56	32	40
J_{e_2}	28	20	**16**	12	20
J_{e_3}	56	40	32	**24**	40

Case $d = 4p$

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_s	20	**8**	28	16	20
J_{e_1}	**0**	0	0	0	0
J_{e_2}	28	20	16	12	**20**
J_{e_3}	56	40	**32**	24	40
J_{e_4}	84	60	48	**36**	60

Case $d = 5p$

Positions	Travaux J_1	J_2	J_3	J_4	J_5
J_{e_1}	**0**	0	0	0	0
J_{e_2}	28	20	16	12	**20**
J_{e_3}	56	**40**	32	24	40
J_{e_4}	84	60	**48**	36	60
J_{e_5}	112	80	64	**48**	80

FIG. 4.5 – Matrices d'affectations selon [165]

4.2.1.1 Une seule machine et famille de dates de fin souhaitées contrôlables

Nous étudions maintenant le $1|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$. Pour commencer, regardons les différentes propriétés de la proposition suivante.

Proposition 4.2.3 *Il existe une solution optimale où :*

1. *le premier travail commence à la date zéro et il n'y a pas de temps mort intermédiaire entre deux travaux adjacents quelconques (aucun temps mort dans une séquence optimale),*
2. *deux travaux adjacents J_i et J_j sont ordonnancés selon*
   - *l'ordre WLPT, s'ils sont en avance et* $\max(C_i; C_j) \leq \min(d_i; d_j)$,
   - *l'ordre WSPT, s'ils sont totalement en retard et* $\min(C_i; C_j) \geq \max(d_i; d_j)$.
3. *pour chaque ensemble $\mathcal{N}_k$, si $n_k\gamma \geq \sum_{J_i \in \mathcal{N}_k} \beta_i$, alors $D_k = 0$; sinon, il existe un travail JàT parmi l'ensemble des travaux de $\mathcal{N}_k$.*

Preuve. La proposition 4.2.3.(1) est triviale. La proposition 4.2.3.(2) peut être montrée par la règle d'échange. En effet, soit une solution optimale où deux travaux adja-

cents J_i et J_j sont ordonnancés en avance. Sans perte de généralité, nous supposons que $d_i \leq d_j$. Soit x une distance donnée par : $x = \min(d_i; d_j) - \max(C_i; C_j)$. Soit δ la distance entre d_i et d_j (cf. la figure 4.2.1.1).

Si $C_i < C_j$, alors la contribution de ces deux travaux au coût d'avance total est : $\alpha_i(p_j + x) + \alpha_j(x + \delta)$; sinon, leur contribution est : $\alpha_j(p_i + x + \delta) + \alpha_i x$. Donc, J_i est ordonnancé avant J_j si et seulement si :

$\alpha_i(p_j + x) + \alpha_j(x + \delta) \leq \alpha_j(p_i + x + \delta) + \alpha_i x$

$\Leftrightarrow \alpha_i p_j \leq \alpha_j p_i$

$\Leftrightarrow p_j/\alpha_j \leq p_i/\alpha_i$

Par conséquent, les travaux adjacents ordonnancés en avance respectent l'ordre non-croissant des p_i/α_i (l'ordre non-décroissant des α_i dans le cas de durées opératoires identiques). De même, les travaux adjacents ordonnancés en retard respectent l'ordre non-décroissant des p_i/β_i (ou bien l'ordre non-croissant des β_i dans le cas de durées opératoires identiques).

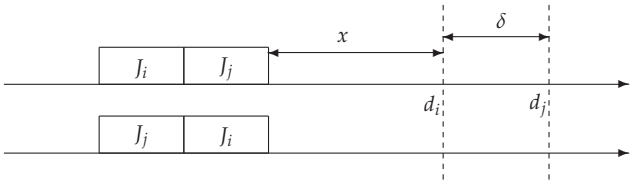

FIG. 4.6 – Deux travaux adjacents ordonnancés en avance

Analysons maintenant la proposition 4.2.3.(3). (a) Si $n_k\gamma \geq \sum_{J_i \in \mathcal{N}_k} \beta_i$ alors $D_k = 0$. La preuve ici est similaire à celle proposée par Panwalkar et al. [169] pour le cas d'une seule date de fin souhaitée contrôlable. Plus précisément, supposons qu'il existe un $D_k \neq 0$ et $n_k\gamma \geq \sum_{J_i \in \mathcal{N}_k} \beta_i$. Si D_k est réduit d'une unité de temps, alors la différence des coûts pour une même séquence est donnée par : $\sum_{J_i \in \mathcal{N}_k, C_i \geq D_k} \beta_i - n_k\gamma - \sum_{J_i \in \mathcal{N}_k, C_i < D_k} \alpha_i$. Cette valeur est non-positive (car $\sum_{J_i \in \mathcal{N}_k, C_i \geq D_k} \beta_i - n_k\gamma \leq \sum_{J_i \in \mathcal{N}_k} \beta_i - n_k\gamma \leq 0$ et $-\sum_{J_i \in \mathcal{N}_k, C_i < D_k} \alpha_i \leq 0$). Ainsi, en suivant le même raisonnement, D_k doit être nul.

(b) Considérons maintenant une solution S telle qu'il n'existe aucun travail $J_i \in \mathcal{N}_k$ avec $C_i = D_k$.

(b.1) On construit une solution S' à partir de S comme suit : on garde la même séquence des travaux, seule la valeur D_k est réduite d'une unité de temps. Nous avons donc : $\Delta = f(S) - f(S') = n_k\gamma + \sum_{J_i \in \mathcal{N}_k; C_i > D_k} \beta_i - \sum_{J_i \in \mathcal{N}_k; C_i \leq D_k} \alpha_i$. Si $\Delta \geq 0$, alors

la solution S peut être améliorée en diminuant D_k d'une unité de temps. Ainsi, en suivant le même raisonnement, on peut diminuer D_k jusqu'à $D_k = 0$ ou bien un travail en retard J_i devient JàT, c-à-d $D_k = C_i$ pour un travail $J_i \in \mathcal{N}_k$.

(b.2) Si D_k est augmentée d'une unité de temps, la déviation du nouveau coût par rapport à l'ancien est donc $-\Delta$. Cela veut dire que S peut être améliorée en augmentant D_k jusqu'à $D_k = C_i$ pour un travail $J_i \in \mathcal{N}_k$.

Par conséquent, une solution avec $D_k = 0$ ou $D_k = C_i$ pour un travail $J_i \in \mathcal{N}_k$, est une solution dominante. □

Selon la proposition 4.2.3, une solution optimale peut être déterminée comme suit :

- On détermine les $n+1$ valeurs possibles de chaque date de fin D_k, $k = 1, 2, \ldots, \ell$, données par $D_k = i \times p$, pour tout $i = 0, 1, \ldots, n$,
- Pour chaque combinaison possible des différentes valeurs des dates de fin souhaitées $(D_1, D_2, \ldots, D_\ell)$, on détermine une matrice d'affectation Q.
- Pour chaque matrice d'affectation Q, on résout un problème d'affectation en $O(n^3)$.

Ainsi, nous avons le théorème suivant.

Théorème 4.2.4 *Une solution optimale pour le problème* $1|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ *peut être obtenue en* $O(n^{\ell+3})$.

4.2.2 Machines identiques avec famille de dates de fin souhaitées

Nous étudions maintenant le problème d'ordonnancement $Pm|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ [108]. Pour commencer, regardons les différentes propriétés de la proposition suivante.

Proposition 4.2.5 *Il existe un ordonnancement optimal où :*

1. *deux travaux adjacents J_i et J_j sont ordonnancés selon*
   - *l'ordre WLPT, s'ils sont en avance et* $\max(C_i; C_j) \leq \min(d_i; d_j)$,
   - *l'ordre WSPT, s'ils sont totalement en retard et* $\min(C_i; C_j) \geq \max(d_i; d_j)$;
2. *pour chaque ensemble $\mathcal{N}_k$, si $n_k\gamma \geq \sum_{J_i \in \mathcal{N}_k} \beta_i$, alors $D_k = 0$; sinon, il existe un travail JàT parmi l'ensemble des travaux de $\mathcal{N}_k$.*

Preuve. idem que pour le cas d'une seule machine (cf. Section 4.2.1.1). □

Soit S une solution optimale et soit X_i la durée du temps mort initial de la machine M_i, c-à-d temps mort situé entre la date zéro et la date de début d'exécution du premier travail ordonnancé sur la machine M_i (X_i peut être nul).

Lemme 4.2.1 *Il existe une solution optimale avec* $X_i \leq p, \forall i = 1, \ldots, m.$

Preuve. Nous considérons trois types de machines (cf. la figure 4.7) :

1. M_{i_1} : machines avec $X_{i_1} \neq 0$,
2. M_{i_2} : machines avec $X_{i_2} = 0$ et pas de temps mort après le travail ordonnancé à la première position,
3. M_{i_3} : machines avec $X_{i_3} = 0$ et un temps mort après le travail ordonnancé à la première position.

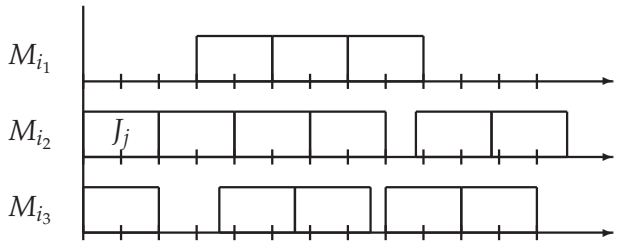

FIG. 4.7 – Différents cas de figures selon les premiers temps morts

Soit S^* une solution optimale où $\exists i \in \{1, \ldots, m\} : X_i > p$, i.e. la machine M_i est de type i_1. Si l'ensemble M_{i_2} est vide, alors nous pouvons réduire le coût total en :

– décalant à gauche d'une unité de temps les travaux ordonnancés sur les machines de type i_1,

– décalant à gauche d'une unité de temps les travaux ordonnancés sur les machines de type i_3 à l'exception du premier travail,

– réduisant d'une unité de temps toutes les dates de fin souhaitées supérieures à p

ce qui contredit le fait que S^* est optimale.

Supposons maintenant qu'il existe au moins une machine de type i_2 où le travail J_i est ordonnancé en première position.

– Si $d_i > p$, alors J_i peut être déplacé et ordonnancé à la première position sur la machine M_i. Ainsi, on obtient une meilleure solution ce qui est contradictoire avec le fait que S^* est optimale ;

– Sinon (i.e. $d_i \leq p$), on construit alors une solution S' à partir de S^* comme suit : le travail J_i est ordonnancé sur M_i avec $S_i = 0$. Ainsi, S' est aussi optimale où aucun temps mort initial n'a une durée supérieure à p.

Par conséquent, il existe un ordonnancement optimal avec X_i inférieure ou égale à $p, \forall i = 1, \ldots, m$. □

Proposition 4.2.6 *Il existe une solution optimale où toutes les dates de fin d'exécution et les dates de fin souhaitées sont des multiples de p.*

Preuve. Soit S^* une solution optimale. Selon les valeurs des dates de fin d'exécution des travaux dans S^*, on définit les deux sous-ensembles suivants : $\mathcal{J}_{mult}$ (resp. $\mathcal{J}_{nomult}$) l'ensemble des travaux dont les dates de fin d'exécution sont des multiples de p (resp. ne sont pas des multiples de p) ; De même, selon les valeurs des dates dues, on définit les deux sous-ensembles suivants : $\mathcal{D}_{mult}$ (resp. $\mathcal{D}_{nomult}$) l'ensemble des dates de fin souhaitées dont les valeurs sont des multiples de p (resp. ne sont pas des multiples de p).

Soit A_1 (resp. B_1) la somme des pondérations d'avance (resp. de retard) des travaux de $\mathcal{J}_{mult}$ ordonnancés en avance (resp. en retard) ayant des dates de fin souhaitées dans $\mathcal{D}_{nomult}$.

Soit A_2 (resp. B_2) la somme des pondérations d'avance (resp. de retard) des travaux de $\mathcal{J}_{nomult}$ ordonnancés en avance (resp. en retard) ayant des dates de fin souhaitées dans $\mathcal{D}_{mult}$. On définit $G = \gamma \times \sum_{D_k \in \mathcal{D}_{nomult}} n_k$.

Il est clair que tous les travaux de $\mathcal{J}_{nomult}$ peuvent être décalés vers la droite ou vers la gauche d'une unité de temps sans que les dates de début d'exécution des travaux de $\mathcal{J}_{mult}$ soient modifiées.

Si tous les travaux de $\mathcal{J}_{nomult}$ et toutes les dates de fin souhaitées de $\mathcal{D}_{nomult}$ sont décalés vers la gauche d'une unité de temps, le statut d'avance-retard des travaux n'est pas modifié ; peut-être que certains travaux en retard (resp. en avance) appartenant à $\mathcal{J}_{nomult}$ (resp. $\mathcal{J}_{mult}$) ayant des dates de fin souhaitées dans $\mathcal{D}_{mult}$ (resp. $\mathcal{D}_{nomult}$) deviennent JàT. La différence du coût total est donnée par : $\Delta = -A_1 + B_1 + A_2 - B_1 - G$. Comme S^* est optimale, Δ doit être inférieur ou égal à 0 ($\Delta \leq 0$).

De la même façon en décalant à droite d'une unité de temps tous les travaux de $\mathcal{J}_{nomult}$ et toutes les dates de fin souhaitées de $\mathcal{D}_{nomult}$, La différence du coût total est égale à $(-\Delta)$. Comme la S est optimale, $(-\Delta)$ doit être ≤ 0.

Par conséquent, $\Delta = 0$. Ainsi, nous pouvons construire une autre solution optimale en décalant vers la gauche tous les travaux de $\mathcal{J}_{nomult}$ ainsi que toutes les dates de fin souhaitées de l'ensemble $\mathcal{D}_{nomult}$. On répète la procédure jusqu'à l'obtention d'une solution où toutes les dates de fin d'exécution et les dates de fin souhaitées sont des multiples de p. Cette solution est aussi optimale □

Proposition 4.2.7 *Il existe une solution optimale où sur chaque machine, la durée du temps mort initial est 0 ou p (i.e. $X_i \in \{0, p\}$, $\forall i = 1, \ldots, m$).*

Preuve. Selon le lemme 4.2.1 ($X_i \leq p$, $\forall i = 1, \ldots, m$) et selon la proposition 4.2.6, nous déduisons donc $X_i \in \{0, p\}$. □

Nous allons maintenant nous intéresser au nombre de temps morts présents sur chaque machine entre deux dates de fin souhaitées successives.

Proposition 4.2.8 *Sur chaque machine, nous avons au plus un seul temps mort entre deux dates de fin souhaitées successives.*

Preuve. Supposons qu'il existe une solution optimale S^* où pour une machine donnée M_i, nous avons deux temps morts entre deux dates de fin souhaitées successives. Soit σ la séquence des travaux exécutés sur M_i entre ces deux temps morts. Par la règle d'échanges, on peut partitionner σ en deux sous-séquences ($\sigma = \sigma_1 // \sigma_2$) telles que :

i. σ_1 contient uniquement les travaux en retard (peut être vide),

ii. σ_2 contient uniquement les travaux en avance (peut être vide).

Le coût $f(S^*)$ peut être amélioré en décalant σ_1 (resp. σ_2) vers la gauche (resp. droite) d'une quantité de temps qui correspond à la durée du temps mort situé juste avant (resp. après) σ, ce qui est contradictoire avec S^* est optimale. □

Selon la proposition 4.2.6, on déduit qu'il existe un ordonnancement optimal où la durée de chaque temps mort est un multiple de p.

Proposition 4.2.9 *Il existe une solution optimale où la durée de chaque temps mort est égale à p.*

Preuve. Nous allons tout d'abord montrer que la durée de chaque temps mort ne peut pas dépasser la valeur de $2p$. Pour une solution optimale, soit les deux dates de fin souhaitées successives $D_{[k]}$ et $D_{[k+1]}$. Entre ces deux dates dues, supposons qu'il existe un temps mort sur la machine M_i de durée $3p$.

– Soit J_j un travail ordonnancé sur une machine $M_{i',i'\neq i}$ entre la date du début et de fin du temps mort considéré sur la machine M_i (cf. Figure 4.8). Nous pouvons donc réduire le coût total en déplaçant le travail J_j sur M_i : si J_j est en retard, il sera donc calé à gauche ; sinon, il sera calé à droite. Ce qui contredit l'optimalité de S^*.

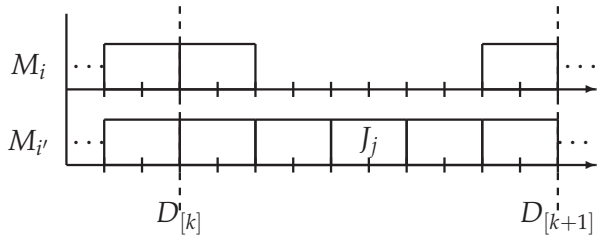

FIG. 4.8 – Exemple avec un temps mort $X = 3p$

– S'il n'existe aucune machine $M_{i',i'=1,\ldots,m\wedge i'\neq i}$ exécutant un travail J_j dans l'intervalle du temps mort considéré, alors le coût total peut être réduit comme suit : on décale de p unités de temps toutes les sous-séquences de toutes les machines exécutées après le temps mort considéré ; on réduit toutes les dates dues $D_{[k+1]}, \ldots, D_{[\ell]}$ de p unités de temps. Ce qui contredit l'optimalité de S^*.

Ainsi, on ne peut pas avoir une solution optimale avec un temps mort de durée supérieure à $2p$.

Montrons maintenant qu'il existe toujours une solution optimale sans temps mort de durée $2p$.

Pour une solution optimale, nous supposons l'existence d'un temps mort sur une machine M_i de durée $2p$.

(a) S'il existe deux travaux J_{j_1} et J_{j_2} ordonnancés sur $M_{i',i'\neq i}$ dans l'intervalle du temps mort considéré (cf. figure 4.9), alors nous pouvons déplacer le travail J_{j_2} sur la machine M_i. Ainsi, au pire, le coût total reste inchangé. Nous avons donc une nouvelle solution optimale avec un temps mort de durée p (cf figure 4.10).

(b) sinon, s'il n'existe aucune machine $M_{i',i'\neq i}$ exécutant deux travaux J_{j_1} et J_{j_2} dans l'intervalle de temps mort considéré, alors S^* n'est pas optimale car nous pouvons réduire le coût total en décalant à gauche tout travail exécuté après le temps mort de p unités de temps et en réduisant de p unités de temps toutes dates dues de valeur supérieure à la date de fin du temps mort considéré sur M_i (cf. figure 4.11).

Par conséquent, pour une machine donnée, nous avons au maximum un seul temps mort entre deux dates de fin souhaitées successives de durée p. □

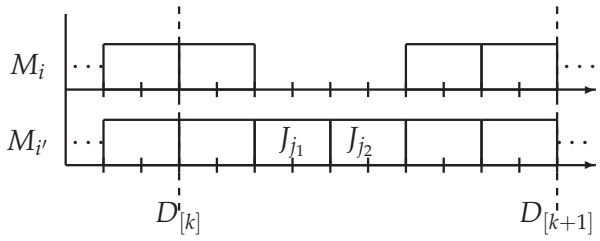

FIG. 4.9 – Exemple avec un temps mort de durée $2p$

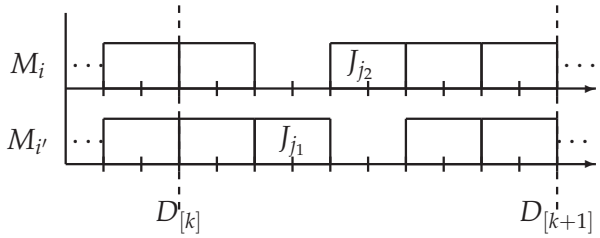

FIG. 4.10 – Construction d'une solution optimale avec un temps mort de durée p

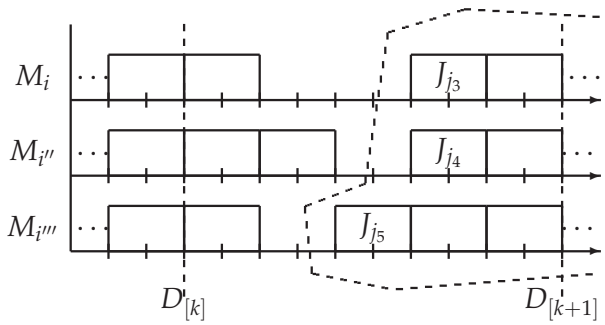

FIG. 4.11 – Exemple d'amélioration d'une solution avec $X = 2p$ par décalage à gauche

Pour résumer, selon les propositions présentées ci-dessus, nous déduisons qu'il existe toujours une solution optimale où à la fois, les dates de fin d'exécution des travaux et les valeurs optimales des dates dues sont des multiples de p. D'autre part, comme nous l'avons vu, la durée du temps mort – s'il existe – sur une machine quelconque entre deux dates dues successives, est p.

Ainsi, pour déterminer une solution optimale, nous devons considérer au maximum $(n+\ell)p$ valeurs possibles de dates de fin souhaitées. De même, nous avons au maximum $(n+l)p$ valeurs possibles de dates de fin d'exécution. Cependant, cette borne peut être améliorée en considérant $(n+m\ell)$ travaux à ordonnancer (n travaux et $(m\ell)$ travaux fictifs qui représentent les ℓ temps morts possibles sur chaque machine). D'autre part, il est clair qu'aucun travail n'est ordonnancé après qp avec $q = \left\lceil \frac{n+m\ell}{m} \right\rceil$. Ainsi, le nombre de dates de fin d'exécution et le nombre de dates de fin souhaitées sont au maximum de qp.

Par conséquent, nous proposons de résoudre le problème $Pm|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ comme suit :

- On détermine les $(q = \left\lceil \frac{n+m\ell}{m} \right\rceil + 1)$ valeurs possibles de chaque date de fin $D_{k,k=1,2,\ldots,\ell}$, i.e. $D_k \in \{0, p, 2p, \ldots, qp\}$, $k = 1, 2, \ldots \ell$.
- Pour chaque combinaison possible des différentes valeurs des dates de fin souhaitées $(D_1, D_2, \ldots, D_\ell)$, on détermine la matrice d'affectation Q de taille $(mq \times n)$. Les éléments de Q sont définis par le coût d'avance-retard des travaux affectés aux différentes positions possibles (rappelons qu'un travail J_j peut être ordonnancé sur une des m machines et $C_j \in \{p, 2p, \ldots, qp\}$).
- Pour chaque matrice Q, on résout un problème d'affectation en $O(mn^3)$ (car $q \leq \left\lceil \frac{n+mn}{m} \right\rceil < 2n$).

Une solution optimale correspond à la meilleure des solutions trouvées des différentes configurations de $(D_1, D_2, \ldots, D_\ell)$ qui peut être déterminée en $O(mn^{\ell+3})$.

Théorème 4.2.10 *Une solution optimale du problème $Pm|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ peut être déterminée en $O(mn^{\ell+3})$.*

Pour un nombre de dates de fin souhaitées ℓ constant et un nombre de machines non fixe, on déduit le théorème suivant.

Théorème 4.2.11 *Le problème d'ordonnancement $P|p_i = p, d_i \in D, |D| = \ell| \sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$ est polynomial.*

Selon les résultats expérimentaux, nous avons quelques remarks comme suit :

Remarque 4.2.1 *Les instances avec* $(m = 2, n = 200, \ell = 1)$ *ou* $(m = 2, n = 20, \ell = 4)$ *peuvent être résolues dans 10000s sur les machines de (CPU 3.00Ghz, 1 Go RAM) en utilisant le solveur CPLEX 10.2 pour résoudre le problème d'affectation.*

Remarque 4.2.2 *Le temps de calcul va diminuer si le nombre des machines augmente.*

4.2.3 Machines uniformes avec une seule date de fin souhaitée

Dans cette section nous considérons un problème d'ordonnancement à machines parallèles uniformes avec une unique date de fin souhaitée contrôlable. Ce problème noté $Qm|p_i = p, d_i = d|\sum(\alpha E_i + \beta T_i + \gamma d)$ a été étudié par [165]. Les auteurs proposent un algorithme en $O(3^m)$. Nous proposons une approche permettant de résoudre deux problèmes connexes en $O(m^2)$. Ces problèmes sont notés :

- $Qm|p_i = p, d_i = d|\sum(\alpha E_i + \beta T_i)$: le coût de la date de fin souhaitée est négligeable.
- $Qm|p_i = p, d_i = d, NSIT|\sum(\alpha E_i + \beta T_i + \gamma d)$: on interdit tout temps mort initial, i.e. avant l'exécution de la première tâche affectée à chaque machine.

Nous commençons par énoncer quelques propriétés.

4.2.3.1 Quelques propriétés triviales

Il existe une solution optimale où :

Propriété 4.2.12 *Il n'existe aucun temps mort entre deux travaux adjacents quelconques.*

Propriété 4.2.13 *Sur chaque machine, il existe un travail* J_i *tel que* $S_i = 0$ *ou* $C_i = d^{opt}$.

Propriété 4.2.14 *Il existe au moins une machine sans temps mort initial.*

Propriété 4.2.15 *Si* $d^{opt} > 0$, *il existe une machine sans temps mort initial exécutant un travail* J_i *avec* $C_i = d^{opt}$.

Propriété 4.2.16 *Sur une machine* M_i *s'il existe un temps mort initial alors* $pX_i \leq \lambda_m p$ *où* λ_m *est la performance de la machine la plus lente* M_m.

Propriété 4.2.17 *([169]) Si* $\gamma \geq \beta/m$, *alors la valeur de la date de fin souhaitée optimale est* $d^{opt} = 0$. *Une solution optimale est obtenue en* $O(n \log n)$ *par la résolution du problème* $Qm||\sum C_j$ *[30].*

Remarque 4.2.3 *Une solution optimale peut être représentée par un vecteur* $\mathcal{Y} = (y_1^e, y_2^e, \ldots, y_m^e, y_1^t, y_2^t, \ldots, y_m^t)$. *En effet, il suffit de connaître le nombre de travaux ordonnancés en avance et en retard sur chaque machine (les travaux sont identiques). Les dates de début d'exécution des travaux sont déterminées selon les propriétés présentées ci-dessus. Ainsi, le coût global* $f(S)$ *peut être calculé en* $O(m)$.

4.2.3.2 Le coût de la date de fin souhaitée est négligeable

Tout d'abord, nous considérons que la valeur de la date de fin souhaitée correspond à la somme des durées opératoires. Ainsi, $d = np\lambda_m$. Pour résoudre ce problème $Qm|p_i = p, d_i = np\lambda_m|\sum(\alpha E_i + \beta T_i)$, nous proposons un algorithme glouton intuitif (cf. la figure 4.12) où une solution optimale peut être déterminée en $O(mn)$.

Algorithme AssQm1 : $Qm|p_i = p, d_i = np\lambda_m|\sum(\alpha E_i + \beta T_i)$

```
Soit L un ensemble de travaux déjà ordonnancés
L = ⊘
Pour chaque travail J_j, j = 1,...,n
  Pour chaque machine M_i, i = 1,...,m rangées dans l'ordre non-croissant des λ_i
    Ordonnancer le travail J_j en position la moins pénalisante
    en respectant les travaux déjà placés, sachant que
    "une position en retard est plus prioritaire qu'une position en avance"
    L = L ∪ {J_j}
  FinPour
FinPour
```

FIG. 4.12 – Algorithme pour $Qm|p_i = p, d_i = np\lambda_m|\sum(\alpha E_i + \beta T_i)$

Ci-dessous une illustration de son déroulement sur un exemple avec $n = 13$, $m = 3$ où $\lambda_i = i, \forall i = \overline{1,3}$ et $p_j = \alpha_j = \beta_j = 1, \forall j$. La solution optimale obtenue est présentée dans la figure 4.13.

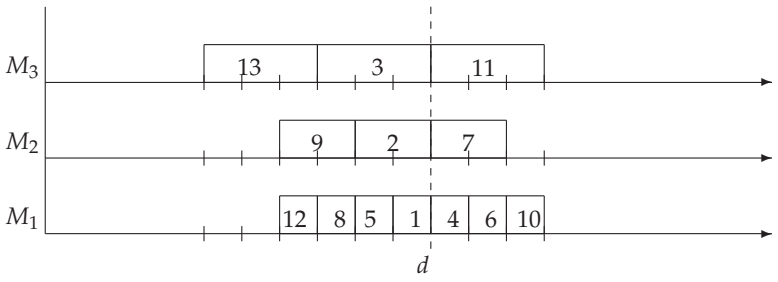

FIG. 4.13 – Exemple d'application de l'algorithme AssQm1 avec 13 travaux

Cependant, nous allons voir comment cet algorithme peut être mieux exploité pour déterminer une solution en $O(m^2)$.

Proposition 4.2.18 *L'algorithme AssQm1 peut déterminer une solution optimale en* $O(m^2)$.

Preuve. L'idée de base de cette preuve repose sur la réduction de la taille de l'instance en entrée, c-à-d. au lieu de considérer les n travaux, nous allons montrer qu'il suffit de déterminer l'affectation d'un nombre restreint de travaux, borné par $O(m)$.

Nous allons écrire l'ensemble des contraintes qui nous permet de retrouver la solution retournée par l'algorithme *AssQm1*. Comme nous allons le constater, le MILP obtenu (modèle linéaire en nombre entier) peut être considéré comme une preuve d'optimalité de l'algorithme *AssQm1*.

Soit $u_{(i,1)}$ (resp. $u_{(i,2)}$) le nombre de travaux ordonnancés en avance (resp. en retard) sur une machine M_i. Ces valeurs correspondent à la solution optimale obtenue par l'algorithme *AssQm1*.

1. Cas particuliers

On doit d'abord considérer les cas particuliers suivants :

- Cas d'une seule machine ($m = 1$) :
  1. si $\alpha \geq (n-1)\beta$ alors $u_{(1,1)} = 0$,
  2. si $\beta > (n-1)\alpha$ alors $u_{(1,2)} = 0$,
- Cas de m machines parallèles identiques ($\lambda_i = \lambda, \forall i$) :
  1. si $\alpha > \frac{(n-m)}{m}\beta$ alors $u_{(i,1)} = 0, \forall i = 1, \ldots, m$,
  2. si $\beta > \frac{(n-m)}{m}\alpha$ alors $u_{(i,2)} = 0, \forall i = 1, \ldots, m$,
- Cas de m machines parallèles uniformes :
  1. si $\alpha > (\frac{(n-m)}{\sum 1/\lambda_i} + \max \lambda_i)\beta$ alors $u_{(i,1)} = 0,\ \forall i = 1, \ldots, m$,
  2. si $\beta > (\frac{(n-m)}{\sum 1/\lambda_i} + \max \lambda_i)\alpha$ alors $u_{(i,2)} = 0,\ \forall i = 1, \ldots, m$,

2. Modèle linéaire en nombre entier.

Nous proposons de modéliser ce problème par un MILP avec $2m$ variables de décisions non-négatives $u_{(i,j)}, i = 1, \ldots, m, j = 1, 2$ telles qu'elles ont été introduites précédemment.

$$\sum_{i=1}^{m}\sum_{j=1}^{2} u_{(i,j)} = n - m \quad \textbf{(1)}$$

$$\alpha u_{(i,1)} < \beta(u_{(i,2)} + 1) \qquad \forall i = 1, \ldots, m \quad \textbf{(2)}$$

$$\beta u_{(i,2)} \leq \alpha(u_{(i,1)} + 1) \qquad \forall i = 1, \ldots, m \quad \textbf{(3)}$$

$$\lambda_i u_{(i,1)} \leq \lambda_k(u_{(k,1)} + 1) \quad \forall i = 1, \ldots, m, \forall k = i+1, \ldots, m \quad \textbf{(4)}$$

$$\lambda_k u_{(k,1)} < \lambda_i(u_{(i,1)} + 1) \quad \forall i = 1, \ldots, m\ , \forall k = i+1, \ldots, m \quad \textbf{(5)}$$

$$\lambda_i u_{(i,2)} \leq \lambda_k(u_{(k,2)} + 1) \quad \forall i = 1, \ldots, m, \forall k = i+1, \ldots, m \quad \textbf{(6)}$$

$$\lambda_k u_{(k,2)} < \lambda_i(u_{(i,2)} + 1) \quad \forall i = 1, \ldots, m\ , \forall k = i+1, \ldots, m \quad \textbf{(7)}$$

$$u_{(i,j)} \in \{0, 1, \ldots, n\} \qquad \forall i = 1, \ldots, m\ , \forall j = 1, 2 \quad \textbf{(8)}$$

Le nombre des travaux en avance et en retard correspond au nombre $n - m$ car nous avons exactement m travaux JàT (contrainte **(1)**). Pour une solution optimale obtenue par l'algorithme *AssQm1*, nous pouvons modéliser la relation entre chaque couple $(u_{(i,1)}, u_{(i,2)})$ par les contraintes **(2)** et **(3)** :

- un travail affecté à la première position sur la machine M_i ne peut pas être ordonnancé après le dernier travail en retard exécuté sur la même machine. Autrement dit, le coût d'avance du premier travail ordonnancé sur M_i ne peut pas être supérieur ou égal à son coût de retard s'il était exécuté en retard sur M_i, en dernière position (contrainte **(2)**) ;
- le dernier travail exécuté sur M_i ne peut pas être déplacé en début de la séquence des travaux exécutée sur M_i (contrainte **(3)**).

De le même façon, la relation entre chaque couple $(u_{(i,1)}, u_{(k,1)})$ est modélisée par les contraintes **(4)** et **(5)** :

- la date de fin d'exécution du premier travail affecté à M_i est supérieure ou égale à la date de début d'exécution de la séquence des travaux exécutée sur M_k pour tout $i < k$ (contrainte **(4)**) ;
- réciproquement, la date de fin d'exécution du premier travail exécuté sur M_k est strictement supérieure à la date de début d'exécution de la séquence exécutée sur M_i pour tout $i < k$ (contrainte **(5)**).

La relation entre chaque couple $(u_{(i,2)}, u_{(k,2)})$ est modélisée par les contraintes **(6)** et **(7)** :

- la date de début d'exécution du dernier travail ordonnancé sur M_i est inférieure ou égale à la date de fin d'exécution du dernier travail ordonnancé sur M_k pour

tout $i < k$ (contrainte **(6)** ;

– la date de début du dernier travail exécuté sur machine M_k est strictement inférieure à la date de fin d'exécution du dernier travail ordonnancé sur M_i pour tout $i < k$ (contrainte **(7)**.

Les contraintes **(8)** expriment l'intégrité des variables de décision.

Analysons d'abord les cas particuliers dus aux valeurs nulles $u_{(i,j)} = 0$. Il est clair que :

1. Si $u_{(i,1)} = 0$, nous pouvons alors déduire :
   - $u_{(k,1)} = 0$ $(\forall i = 1, \ldots, m, \forall k \geq i)$;
   - $u_{(k,2)} \leq \left\lfloor \frac{\lambda_k}{\lambda_i} \right\rfloor$ $(\forall i = 1, \ldots, m, \forall k < i)$;
   - $u_{(i,2)} \leq \left\lfloor \frac{\beta}{\alpha} \right\rfloor$ $(\forall i = 1, \ldots, m)$.
2. Si $u_{(i,2)} = 0$, nous pouvons alors déduire :
   - $u_{(k,2)} = 0$ $(\forall i = 1, \ldots, m, \forall k \geq i)$;
   - $u_{(k,1)} \leq \left\lfloor \frac{\lambda_k}{\lambda_i} \right\rfloor$ $(\forall i = 1, \ldots, m, \forall k < i)$;
   - $u_{(i,1)} < \left\lfloor \frac{\beta}{\alpha} \right\rfloor$ $(\forall i = 1, \ldots, m)$.

Nous montrons que l'analyse des cas particuliers s'effectue en $O(m)$. En effet, nous disposons d'un certain nombre d'indicateurs qui nous permettent d'identifier les cas particuliers. Ces indicateurs sont définis comme suit :

– si $\alpha \geq \beta$, alors :

  1. $u_{(i,2)} = 0$ ssi $n \leq m + \sum_{k=1}^{i-1} (\left\lfloor \frac{\lambda_i}{\lambda_k} \right\rfloor + \left\lfloor \frac{\lambda_i \alpha}{\lambda_k \beta} \right\rfloor)$;
  2. $u_{(i,1)} = 0$ ssi $n \leq m + \sum_{k=1}^{i-1} (\left\lfloor \frac{\lambda_i}{\lambda_k} \right\rfloor + \left\lfloor \frac{\lambda_i \alpha}{\lambda_k \beta} \right\rfloor) + \frac{\alpha}{\beta}$.

– si $\beta \geq \alpha$, alors :

  1. $u_{(i,1)} = 0$ ssi $n \leq m + \sum_{k=1}^{i-1} (\left\lfloor \frac{\lambda_i}{\lambda_k} \right\rfloor + \left\lfloor \frac{\lambda_i \beta}{\lambda_k \alpha} \right\rfloor)$;
  2. $u_{(i,2)} = 0$ ssi $n \leq m + \sum_{k=1}^{i-1} (\left\lfloor \frac{\lambda_i}{\lambda_k} \right\rfloor + \left\lfloor \frac{\lambda_i \beta}{\lambda_k \alpha} \right\rfloor) + \frac{\beta}{\alpha}$.

Ces indicateurs nous permettent donc de déterminer les valeurs nulles de certaines variables $u_{(i,j)}$. Il nous reste par la suite à calculer les valeurs des variables strictement positives $(u_{(i,j)} \geq 1)$. Ces valeurs restantes peuvent être déduites du MILP réduit (puisque certaines valeurs ont été déjà calculées) selon une procédure décrite ci-dessous.

Dans le cas général où $(u_{(i,j)} \geq 1), \forall i = 1, \ldots, m, j = 1, 2$, nous procédons de la façon suivante :

3. Relaxation du MILP en nombres réels.

En partant du MILP présenté ci-dessus, nous allons réécrire un nouveau modèle linéaire simple (PLS) où les variables de décisions sont des réelles notées par $u'_{(i,j),i=\overline{1,m},j=1,2}$. L'objectif ici est d'avoir un PLS pour lequel une solution réalisable correspond à une solution réelle réalisable du MILP relâché.

$\sum_{i=1}^{m}\sum_{j=1}^{2} u'_{(i,j)} = n - m$ **(1')**

$\alpha u'_{(i,1)} = \beta u'_{(i,2)} \quad \forall i = 1,\dots,m$ **(2')**

$\lambda_i u'_{(i,j)} = \lambda_k u'_{(k,j)} \ \forall j = 1,2, \forall i = 1,\dots,m, \forall k = 1,\dots,m$ **(3')**

Ainsi, les différentes valeurs des $u'_{(i,j)}, i = 1,\dots,m, j = 1,2$ peuvent être déduites directement comme suit :

- $u'_{(1,1)} = \frac{\alpha+\beta}{\alpha}(n-m)\sum_{i=1}^{m}\frac{\lambda_1}{\lambda_i}$;
- $u'_{(1,2)} = \frac{\alpha u'_{(1,1)}}{\beta}$;
- $u'_{(k,j)} = \frac{\lambda_1 u'_{(1,j)}}{\lambda_k}\ \forall j = 1,2, \forall k = 2,\dots,m.$

Remarque 4.2.4 *Cette étape est effectué en $O(m)$.*

4. Arrondir les valeurs.

Selon la solution réelle obtenue par l'étape précédente, les valeurs arrondies entières $u''_{(i,j)}$ peuvent être déterminées : $u''_{(i,j)} = \left\lfloor u'_{(i,j)} \right\rfloor,\ i = 1,\dots,m, j = 1,2.$

Nous pouvons ainsi vérifier que les valeurs $u''_{(i,j)}$ respectent l'ensemble des contraintes du MILP à l'exception de la contrainte **(1)**. En effet, $\forall i = 1,\dots,m$, $\forall k = i+1,\dots,m$, nous avons :

- $n - 2m \leq \sum_{i=1}^{m}\sum_{j=1}^{2} u''_{(i,j)} \leq n - m$ **(1'')**
- $\alpha u''_{(i,1)} \leq \alpha u'_{(i,1)} = \beta u''_{(i,2)} < \beta(u''_{(i,2)} + 1)$
- $\beta u''_{(i,2)} \leq \beta u'_{(i,2)} = \alpha u''_{(i,1)} < \alpha(u''_{(i,1)} + 1)$
- $\lambda_i u''_{(i,1)} \leq \lambda_i u'_{(i,1)} = \lambda_k u'_{(k,1)} < \lambda_k(u''_{(k,1)} + 1)$
- $\lambda_k u''_{(k,1)} \leq \lambda_k u'_{(k,1)} = \lambda_i u'_{(i,1)} < \lambda_i(u''_{(i,1)} + 1)$
- $\lambda_i u''_{(i,2)} \leq \lambda_i u'_{(i,2)} = \lambda_k u'_{(k,2)} < \lambda_k(u''_{(k,2)} + 1)$
- $\lambda_k u''_{(k,2)} \leq \lambda_k u'_{(k,2)} = \lambda_i u'_{(i,2)} < \lambda_i(u''_{(i,2)} + 1)$

$\Rightarrow$ Les valeurs de $u''_{(i,j)}, i = 1,\dots,m, j = 1,2$ déterminent donc une solution optimale du $Qm|p_i = p, d_i = np\lambda_m|\sum(\alpha E_i + \beta T_i)$ avec $(m + \sum_{i=1}^{m}\sum_{j=1}^{2} u''_{(i,j)})$ travaux où exactement m travaux sont JàT. Cependant, nous devons donc déterminer l'ordonnancement des $n - (m + \sum_{i=1}^{m}\sum_{j=1}^{2} u''_{(i,j)})$ travaux restants. Selon l'inégalité **(1'')**, ce nombre

est borné par m. Ainsi, l'ordonnancement des ces travaux restants est réalisé par l'algorithme *AssQm1* en $O(m^2)$. □

De ce qui vient d'être exposé, nous pouvons déduire le théorème suivant :

Théorème 4.2.19 *Une solution optimale du problème $Qm|p_i = p|\sum C_i$ peut être obtenue en $O(m^2)$.*

Preuve. En effet, il suffit de remplacer les variables de décision $u_{i,j}$ par u_i. u_i représentant le nombre de travaux exécutés sur la machine M_i. Ainsi, la preuve reste valide. □

4.2.3.3 Pas de temps mort initial sur les machines

Dans la suite, nous considérons le problème d'ordonnancement $Qm|p_i = p, d_i = d, NSIT|\sum(\alpha E_i + \beta T_i + \gamma d)$ pour lequel une solution optimale ne doit pas avoir un temps mort initial $\forall i = 1, \ldots, m$. Nous notons par d_{NSIT} la date de fin souhaitée optimale.

Proposition 4.2.20 *Une solution optimale pour le problème $Qm|p_i = p, d_i = d, NSIT|\sum(\alpha E_i + \beta T_i + \gamma d)$ peut être obtenue en $O(m^2)$.*

Preuve.
Puisqu'il s'agit de déterminer une solution optimale sans temps mort initial, l'ordonnancement des travaux est obtenu en résolvant le problème $Qm|p_i = p|\sum C_i$ en $O(m^2)$ (voir théorème 4.2.19). On peut alors construire un algorithme glouton pour déterminer la date de fin souhaitée optimale d_{NSIT} en $O(m^2n)$ en deux étapes :

1. Si la date de fin souhaitée est différente de zéro, il existe un travail JàT, nous avons donc au maximum $mn + 1$ valeurs possibles de d_{NSIT},
2. Selon une date de fin souhaitée donnée, on détermine le nombre de travaux en avance et en retard sur chaque machine. Donc, le coût sera calculé en $O(m)$ (cf. remarque 4.2.3).

Cependant, nous allons montrer que le nombre de d_{NSIT} possibles est bornée par $O(m)$. Ainsi, le calcul de d_{NSIT} peut être donné en $O(m^2)$ comme suit.

Soient n_1, n_2 et n_3 le nombre des travaux ordonnancés en avance, JàT et en retard. Nous avons donc $n = n_1 + n_2 + n_3$. Pour une date de fin souhaitée, si d_{NSIT} est déplacée

vers la gauche d'une unité de temps, le coût global ne peut pas être amélioré. Ainsi, nous avons :

$-n_1\alpha + (n_2 + n_3)\beta - n\gamma \geq 0$

$\Leftrightarrow -n_1\alpha + (n - n_1)\beta - n\gamma \geq 0$

$\Leftrightarrow -n_1(\alpha + \beta) + n(\beta - \gamma) \geq 0$

$\Leftrightarrow n_1 \leq \frac{n(\beta - \gamma)}{\alpha + \beta}$

Comme $n_2 \leq m$, nous avons donc : $n_1 + n_2 \leq \frac{n(\beta - \gamma)}{\alpha + \beta} + m$.

Si d_{NSIT} est déplacée vers la droite d'une unité de temps, le coût global ne peut pas être amélioré. Ainsi, nous avons :

$(n_1 + n_2)\alpha - n_3\beta + n\gamma \geq 0$

$\Leftrightarrow (n - n_3)\alpha - n_3\beta + n\gamma \geq 0$

$\Leftrightarrow n_3 \leq \frac{n(\alpha + \gamma)}{\alpha + \beta}$

$\Leftrightarrow n - n_1 - n_2 \leq \frac{n(\alpha + \gamma)}{\alpha + \beta}$

$\Leftrightarrow n_1 + n_2 \geq \frac{n(\beta - \gamma)}{\alpha + \beta}$

Par conséquent, $\frac{n\beta - \gamma}{\alpha + \beta} \leq n_1 + n_2 \leq \frac{n\beta - \gamma}{\alpha + \beta} + m$.

C'est-à-dire, nous avons au maximum $m + 1$ valeurs possibles pour la somme $(n_1 + n_2)$. D'où, nous devons considérer $m + 1$ valeurs possibles pour déterminer la valeur optimale d_{NSIT} selon la procédure suivante :

1. Considérons d'abord le cas : $n_1 + n_2 = \frac{n\beta - \gamma}{\alpha + \beta}$. On résout le problème $Qm||\sum C_i$ avec les $n_1 + n_2$ travaux (cf théorème 4.2.19). Ainsi, le makespan obtenu définit la première date de fin souhaitée d_0 (en $O(m)$). Le coût de la solution optimale $S(d_0)$ sur les n travaux pour la valeur de d_0 est obtenu en $O(m)$ (cf. remarque 4.2.3).

2. De façon itérative, pour $n_1 + n_2$ variant de $(\frac{n\beta - \gamma}{\alpha + \beta} + 1)$ à $(\frac{n\beta - \gamma}{\alpha + \beta} + m)$ par pas de 1 (i.e. $i = 1, \ldots, m$), on opère de la façon suivante : à chaque itération i on affecte un nouveau travail à la machine qui lui permettrait de finir au plus tôt. L'affectation est effectuée en $O(m)$. La nouvelle date de fin souhaitée d_i correspond à la nouvelle valeur du makespan, donnée . Le coût de la solution optimale $S(d_i)$ sur les n travaux pour la valeur de d_i est obtenu en $O(m)$.

La date de fin souhaitée optimale d_{NSIT} est donnée par : $d_{NSIT} = d_i | i = argmin_{j=\overline{0,m}}(S(d_j))$. La complexité au total est donc en $O(m^2)$. □

4.3 Durées opératoires quelconques

4.3.1 Une seule machine avec une seule date de fin souhaitée

Dans cette section, nous allons d'abord étudier l'équivalence entre les deux problèmes suivants : $1|d_i = d, non - restrictive|\sum(\alpha E_i + \beta T_i)$ qu'on appellera UNRES et le problème CON, noté $1|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$. Par cette étude, nous montrons qu'on peut construire une solution optimale pour l'un si on connait une solution optimale de l'autre. Par la suite, l'existence d'un schéma d'approximation sera abordé.

4.3.1.1 Equivalence entre problèmes CON et cas non-restrictif

En s'appuyant sur la proposition 4.2.1 présentée dans la section 4.2.1, nous montrons que $d^{opt} = \sum_{i,C_i \leq d} p_i$. L'idée globale de notre raisonnement se base sur l'ajout d'un travail fictif. Pour une solution optimale, nous montrons que ce travail est ordonnancé en première position en avance.

Lemme 4.3.1 *UNRES* $\propto$ *CON.*

Proof. Supposons qu'on sait résoudre CON alors il suffit de poser $\gamma = 0$. La solution obtenue dans ce cas est une solution optimale pour le problème UNRES. □

Lemme 4.3.2 *CON* $\propto$ *UNRES.*

Proof. Supposons qu'on sait résoudre UNRES, soit une instance de n travaux du problème CON quelconque. Nous construisons une instance avec $n+1$ travaux du problème UNRES comme suit :

- pour le travail J_i, $i = 1, \ldots, n$: sa durée opératoire, sa pénalité d'avance et de retard sont identiques pour les deux instances ;
- pour le travail J_{n+1} : $p_{n+1} = \max(P; 2R \times n\gamma)$; $\alpha_{n+1} = n\gamma$; $\beta_{n+1} = n\gamma + \sum_{k=1}^{n} \alpha_k$ avec $R = \max_{i=1}^{n}(p_i/\alpha_i)$

Par définition, nous avons $\frac{p_{n+1}}{\alpha_{n+1}} \geq 2R > \frac{p_i}{\alpha_i}$ pour tout $i = 1, \ldots, n$. Nous allons montrer comment construire une solution optimale $S^{*(CON)}$ pour CON à partir d'une solution optimale de UNRES $S^{*(UNRES)}$.

Concernant la décision du travail J_{n+1} dans $S^{*(UNRES)}$,

- si J_{n+1} est ordonnancé en avance ou JàT (i.e. $C_{n+1} \leq d$), ce travail occupe alors la première position (l'ordonnancement des travaux respecte la règle V-shape).

– sinon (J_{n+1} est en retard, $C_{n+1} > d$) : on construit une nouvelle solution S' par le déplacement de la séquence $S^{*(UNRES)}$ d'une unité de temps vers la gauche. Le nouveau coût global par rapport à la solution $S^{*(UNRES)}$ diminue de Δ donnée par : $\Delta = \sum_{i=1,C_i \geq d}^{n} \beta_i + \beta_{n+1} - \sum_{i=1,C_i<d}^{n} \alpha_i > 0$ (ce qui contredit l'optimalité de $S^{*(UNRES)}$).

Par conséquent, J_{n+1} est ordonnancé en première position en avance.

Construisons maintenant une solution $S^{(CON)}$ pour CON à partir de $S^{*(UNRES)}$, comme suit :

– les travaux ordonnancés en avance (resp. en retard) dans $S^{(CON)}$ correspondent aux travaux ordonnancés en avance (resp. en retard) dans $S^{*(UNRES)}$ à l'exception de J_{n+1} ;
– la valeur de la date de fin souhaitée optimale d^{opt} est donnée par la somme des durées opératoires des travaux ordonnancés en avance.

Par conséquent, le coût total de l'avance et de retard des travaux défini par $S^{*(UNRES)}$ n'est autre que le coût global correspondant à la solution $S^{(CON)}$.

Nous montrons que $S^{(CON)}$ est aussi optimale. Supposons qu'il existe une autre solution meilleure $S_1^{(CON)}$. À partir de $S_1^{(CON)}$, on construit un autre ordonnancement pour le problème UNRES, noté $S_1^{(UNRES)}$:

– l'ensemble des travaux ordonnancés en avance et l'ensemble des travaux ordonnancés en retard pour les deux solutions $S_1^{(CON)}$ et $S_1^{(UNRES)}$ sont identiques (l'ordre des travaux est bien sûr le même) ;
– le travail J_{n+1} sera ordonnancé en première position en avance.

Par conséquent, le coût global de $S_1^{(UNRES)}$ est égal au coût global de $S_1^{(CON)}$. On obtient alors une meilleure solution que $S^{*(UNRES)}$(contradiction avec l'optimalité de $S^{*(UNRES)}$). □

Nous avons donc le théorème suivant.

Théorème 4.3.1 *$CON \equiv UNRES$.*

Selon ce dernier théorème, il est donc possible de construire en temps polynomial une solution optimale pour le $1|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$ à partir d'une solution obtenue par la résolution du problème $1|d_i = d|\sum(\alpha_i E_i + \beta_i T_i)$ et vice-versa. Par exemple, on propose de résoudre CON par la procédure par séparation et évaluation (PSE) proposée par Sourd [202] pour le problème $1|d_i = d|\sum(\alpha_i E_i + \beta_i T_i)$. En effet, pour des

durées opératoires entre $[1, 20]$, Sourd montre que la PSE proposée peut aller jusqu'à 1000 travaux en un temps ne dépassant pas $1/2$ heure de calculs.

Dans le paragraphe suivant, nous allons montrer comment un schéma d'approximation polynomial peut être déduit pour le problème CON à partir des schémas définis pour le cas non-restrictif.

4.3.1.2 Schémas d'approximation

Théorème 4.3.2 *Le cas symétrique du problème CON,* $1|d_i = d|\sum(w_iE_i + w_iT_i + \gamma d)$*, admet un FPTAS.*

Preuve. Il a été montré par Kovalyov et Kubiak [140] que le problème UNRES symétrique, $1|d_i = d, non - restrictive|\sum w_i(E_i + T_i)$, accepte un FPTAS. Pour déterminer un FPTAS pour le problème CON, nous construisons une instance pour le UNRES avec $n+1$ travaux à partir d'une instance quelconque de n travaux du problème CON comme suit :

- pour le travail J_i, $i = 1 \ldots n$: sa durée opératoire, sa pénalité d'avance et de retard sont identiques pour les deux instances ;
- pour le travail J_{n+1} : $p_{n+1} = \max(2P; R \times n\gamma)$; $w_{n+1} = n\gamma$ avec $R = \max_{i=1}^{n}(p_i/\alpha_i)$.

En suivant la même démarche que celle utilisée pour le lemme 4.3.2, à partir de la solution obtenue par le FPTAS du cas symétrique du problème UNRES, nous pouvons donc construire en temps polynomial une solution pour le CON avec garantie de performance $(1 + \varepsilon)$. □

Théorème 4.3.3 *Le problème d'ordonnancement* $1|d_i = d|\sum(\alpha_iE_i + \beta_iT_i + \gamma d)$ *accepte un PTAS sous l'hypothèse* $\alpha_{min} \leq \alpha_i \leq \alpha_{max}, \forall i = 1 \ldots n$ *et* $\alpha_{max}/\alpha_{min} \leq 2$.

Preuve. Rappelons que le problème $1|d_i = d, non - restrictive|\sum(\alpha_iE_i + \beta_iT_i)$ accepte un PTAS (cf. section 3.4.1) [106]. Pour déterminer un PTAS pour le problème CON, nous construisons une instance pour le UNRES avec $n+1$ travaux à partir d'une instance quelconque de n travaux du problème CON comme suit :

- pour le travail J_i, $i = 1 \ldots n$: sa durée opératoire, sa pénalité d'avance et de retard sont identiques pour les deux instances ;
- pour le travail J_{n+1} : $p_{n+1} = \max(P; R \times n\gamma)$; $\alpha_{n+1} = n\gamma$; $\beta_{n+1} = 2\alpha_{n+1}/\varepsilon$ avec $R = \max_{k=1}^{n}(p_k/\alpha_k)$.

Notons que selon le PTAS proposé pour le problème UNRES, le travail J_{n+1} est 'décidé en avance' ($\alpha_{n+1} < \varepsilon\beta_{n+1}$ cf. section 3.4.1).

En suivant la même démarche que celle utilisée pour le lemme 4.3.2, à partir de la solution obtenue par le PTAS du problème UNRES, nous pouvons donc construire en temps polynomial une solution pour le CON avec garantie de performance $(1+\varepsilon)$. □

4.3.2 Machines parallèles avec une seule date de fin souhaitée

Pour résoudre le problème $Pm|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$, un modèle mathématique en nombres entiers est développé. A la fin de cette section, nous montrons comment obtenir une borne inférieure en utilisant la relaxation Lagrangienne.

4.3.2.1 Propriétés

Dans le cas de m machines parallèles, il est facile de montrer les propriétés suivantes :

Propriété 4.3.4 *Il existe une solution optimale où :*

1. *il n'y a pas de temps mort entre deux travaux adjacents ;*
2. *sur chaque machine, il existe un travail qui commence à la date zéro ou s'achève à la date d (le deuxième cas est possible si $d > 0$) ;*
3. *il y a au moins un travail qui commence à la date zéro ;*
4. *si $d > 0$, il existe une machine sur laquelle un travail commence à la date zéro et un autre travail se termine à la date d.*

4.3.2.2 Modèle mathématique

Nous allons présenter dans la suite un modèle basé sur une formulation indexée sur le temps (*time indexed formulation*). Soit x_{it} une variable binaire valant 1 si $C_i = t$ et 0 sinon. On introduit un horizon d'ordonnancement $\mathcal{T}$ qui est une borne supérieure du $C_{\max}$ (date de fin globale) de la solution optimale. Par exemple, $\mathcal{T}$ peut être défini par $\sum_{i=1}^{n} p_i$. Soit n le nombre de travaux, le problème d'ordonnancement $Pm|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$, avec (Z) la fonction objectif, est défini de la manière suivante :

$$Z = \min \sum_{i=1}^{n} \sum_{t=0}^{\mathcal{T}} f_i(d,t) x_{it} \tag{4.1}$$

sous les contraintes :

$$\sum_{t=0}^{\mathcal{T}} x_{it} = 1, \forall i \in \{1, 2, \ldots, n\} \tag{4.2}$$

$$\sum_{i=1}^{n} \sum_{s=t+1}^{t+p_i} x_{it} \leq m, \forall t \in [0, \mathcal{T}] \tag{4.3}$$

$$x_{it} = 0, \forall i \in \{1, 2, \ldots, n\}, \forall t \in [0, p_i - 1] \tag{4.4}$$

$$x_{it} \in \{0, 1\}, \forall i \in \{1, 2, \ldots, n\}, \forall t \in [0, \mathcal{T}] \tag{4.5}$$

L'équation 4.2 nous assure que chaque travail ne s'exécute qu'une seule fois. À un instant t, au maximum m travaux sont exécutés en parallèle (inégalité 4.3). La date de fin d'un travail exécuté au début de l'ordonnancement correspond à sa durée opératoire (équation 4.4). Les variables de décision sont binaires (contrainte 4.5), i.e. $x_{it} = 1$ si J_i se termine à la date t ; 0 sinon.

Le coût de l'ordonnancement est donné par la fonction objectif définie par l'équation 4.1. On pourra noter que cette fonction objectif est en fonction de d et t : $f_i(d, t) = \alpha_i \max(0, d - t) + \beta_i \max(0, t - d) + \gamma d$.

4.3.2.3 Borne inférieure

Les bornes inférieures de qualités jouent un rôle important pour la mise en place d'une méthode exacte telle qu'une PSE ou même pour mesurer les performances des (meta-)heuristiques. Dans la suite, nous proposons donc une borne inférieure basée sur la relaxation Lagrangienne [122]. En effet, la relaxation Lagrangienne est une technique qui consiste à relâcher des contraintes difficiles en les intégrant dans la fonction objectif et en la pénalisant si cette contrainte n'est pas respectée [82]. Pour le cas de dates de fin souhaitées données, plusieurs bornes inférieures utilisant la relaxation Lagrangienne ont été développées [203, 64, 132, 22].

En se basant sur le modèle MIP présenté ci-dessus, nous proposons de relâcher la contrainte de non-chevauchement des travaux (inégalité 4.3). Les multiplicateurs lagrangiens $\mu = (\mu_0, \mu_1, \ldots, \mu_{\mathcal{T}})$ sont introduits. Pour résoudre le problème (LR), on résout le problème (LR_μ) avec un vecteur $\mu \geq 0$ donné. Une solution optimale de (LR) correspond au $\max_\mu(LR_\mu)$. La fonction objectif de (LR_μ) est donnée comme suit :

$$
\begin{aligned}
LR_\mu &= min\left(\sum_{i=1}^{n}\sum_{t=0}^{\mathcal{T}} f_i(d,t)x_{it} + \sum_{t=0}^{\mathcal{T}}\mu_t(\sum_{i=1}^{n}\sum_{s=t+1}^{t+p_i} x_{is} - m)\right) \\
&= min\left(\sum_{i=1}^{n}\sum_{t=0}^{\mathcal{T}} f_i(d,t)x_{it} + \sum_{t=0}^{\mathcal{T}}\mu_t\sum_{i=1}^{n}\sum_{s=t+1}^{t+p_i} x_{is} - m\sum_{t=0}^{\mathcal{T}}\mu_t\right) \\
&= min\left(\sum_{i=1}^{n}(\sum_{t=0}^{\mathcal{T}} f_i(d,t)x_{it} + \sum_{t=0}^{\mathcal{T}}\mu_t\sum_{s=t+1}^{t+p_i} x_{is}) - m\sum_{t=0}^{\mathcal{T}}\mu_t\right) \\
&= min\left(\sum_{i=1}^{n}(\sum_{t=0}^{\mathcal{T}} f_i(d,t)x_{it} + \sum_{t=0}^{\mathcal{T}} x_{it}\sum_{s=t-p_i}^{t-1}\mu_s) - m\sum_{t=0}^{\mathcal{T}}\mu_t\right)
\end{aligned}
$$

$$
\Rightarrow LR_\mu = \min\left(\sum_{i=1}^{n}\sum_{t=0}^{\mathcal{T}} x_{it}(f_i(d,t) + \sum_{s=t-p_i}^{t-1}\mu_s) - m\sum_{t=0}^{\mathcal{T}}\mu_t\right) \tag{4.6}
$$

Le coût de la solution optimale du problème $(LR_{\mu\geq 0})$ n'est autre qu'une borne inférieure pour le problème (Z), puisque :

- soit S une solution réalisable du problème. Alors, $LR_\mu(S) \leq Z(S)$, du fait que la contribution de la pénalité liée à la contrainte 4.3 au coût globale de LR_μ est négative.
- soit S^{opt} une solution optimale pour le problème (Z). Alors, $LR_\mu(S^{opt}) \leq Z(S^{opt})$.
- soit S^{opt}_{LR} une solution optimale pour le problème (LR). Alors, $LR_\mu(S^{opt}_{LR}) \leq LR(S^{opt})$.

$\Rightarrow LR_\mu(S^{opt}_{LR}) \leq Z(S^{opt})$.

Algorithme pour le problème primal

Considérons maintenant la version primale du problème (LR), c-à-d. pour une valeur fixée du vecteur μ, nous cherchons une solution optimale pour le problème (LR_μ). Dans la fonction du coût de (LR_μ), la valeur $m\sum_{t=0}^{\mathcal{T}}\mu_t$ est une constante. On cherche donc à minimiser la valeur $\sum_{i=1}^{n}\sum_{t=0}^{\mathcal{T}} x_{it}f_i(d,t) + \sum_{s=t-p_i}^{t-1}\mu_s$ (équation 4.6).

Pour un travail J_i et pour une date de fin souhaitée d donnée, on définit une courbe représentant le coût d'avance et une autre pour le coût de retard définies par la fonction objectif $f_i(d,t)x_{it}$. Supposons que le travail J_i est en avance et se termine à la date t. La différence des coûts d'avance, s'il se termine d'une unité de temps plus tôt (i.e. à $t-1$), est donnée par : $f_i(d,t-1)x_{it-1} - f_i(d,t)x_{it} = \alpha_i$. De même, la différence des coûts de retard s'il se termine d'une unité de temps plus tard est : $f_i(d,t+1)x_{it+1} - f_i(d,t)x_{it} =$

β_i. Par conséquent, les courbes d'avance et de retard définies par $f_i(d,t)$ sont linéaires.

On s'intéresse maintenant au coût donné par $\sum_{s=t-p_i}^{t-1} \mu_s$ selon l'équation 4.6. Notons $f_m(t) = \sum_{s=t-p_i}^{t-1} \mu_s$. Nous avons :

$$f_m(t) - f_m(t-1) = \sum_{s=t-p_i}^{t-1} \mu_s - \sum_{s=t-p_i-1}^{t-2} \mu_s = \mu_{t-1} - \mu_{t-p_i-1} \tag{4.7}$$

Selon la formule 4.7, en calculant la valeur de la fonction objectif pour $t = 0$, nous pouvons déduire l'allure des deux courbes (d'avance et de retard) avec $t > 0$.

Selon les différentes valeurs des multiplicateurs lagrangiens μ, nos courbes d'avance et de retard d'un travail J_i, pour une date de fin souhaitée d donnée, sont définies par la fonction de coût suivante (cf. figure 4.14) :

$$f_i(t,d) + \sum_{s=t-p_i}^{t-1} \mu_s \tag{4.8}$$

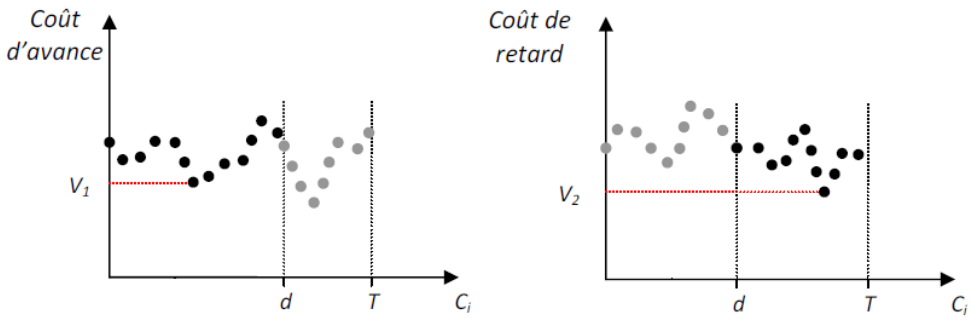

FIG. 4.14 – Courbes du coût d'avance et de retard d'un travail J_i

On s'intéresse donc à la date de fin d'exécution de chaque travail J_i. Pour une date de fin souhaitée donnée d : soit v_1 le coût d'avance minimal de J_i et v_2 son coût de retard minimal (cf. la courbe des coûts d'avance et retard de la figure 4.14). La valeur $\min(v_1, v_2)$, nous permet de déterminer la date de fin d'exécution de J_i.

Comme la valeur de la fonction du coût dépend entre autre, de d, pour une nouvelle valeur de la date due $d + x$, nous avons une nouvelle courbe des coûts d'avance (resp. retard) où la distance entre les deux courbes est $x \times \alpha_i$ (reps. $x \times \beta_i$).

Ainsi, si $d = x + 1$, on doit déterminer la nouvelle valeur minimale définie par la plus petite valeur entre le meilleur coût d'avance calculé dans l'intervalle $[0, x]$ et v'_1 correspondant à la nouvelle date de fin souhaitée. De même, pour la courbe des différents coûts de retard de J_i, la nouvelle valeur minimale v'_2 pour $d = x + 1$ est définie

par v_2 pour $d = x$ si le point correspondant à v_2 correspond à une position en retard, sinon on doit déterminer la meilleure nouvelle valeur v_2. Dans ce cas, C_i optimale est donnée par la valeur minimale entre v_1 et v_2.

Comme ici on ne s'intéresse qu'à la date de fin souhaitée qui minimise la fonction objectif, pour simplifier, on sépare les deux courbes comme suit : la courbe indiquant les différents coût d'avance d'un travail est définie par $d = \mathcal{T}$. Quant à la courbe des différents coûts de retard, elle est définie par $d = 0$. Cependant, on peut définir une liste des différents points de la courbe des coûts de retard, rangés dans l'ordre non-décroissant, afin de minimiser le temps de recherche de la meilleure valeur v_2.

Pour déterminer la date de fin souhaitée optimale pour le (LR), l'algorithme doit donc parcourir les différentes valeurs possibles de la date de fin souhaitée $d \in [0, \mathcal{T}]$. Puis, pour chaque valeur de d, déterminer la date de fin d'exécution pour chaque travail qui minimise l'équation 4.8, ainsi que le coût de la fonction objectif de LR_μ (l'équation 4.6).

La complexité de l'algorithme est donc en $O(n\mathcal{T} \log \mathcal{T})$. Dans la section suivante, nous allons montrer comment déterminer les valeurs du vecteur μ afin que la solution optimale trouvée par la résolution du problème (LR_μ) soit maximale, donc la meilleure borne inférieure pour le problème.

Algorithme pour le problème dual Pour déterminer le vecteur μ, nous utilisons la méthode du sous-gradient [219] afin d'optimiser les valeurs de μ et ainsi pouvoir minimiser l'écart entre la borne inférieure du problème (LR_μ) pour un μ donné, et une borne supérieure du problème.

Algorithme Sous-gradient

1. Initialiser $\mu^{(0)}$ à zéro.
2. Rappelons que $f(\mu) = \sum_{i=1}^{n}(\sum_{t=0}^{\mathcal{T}} f_i(d,t)x_{it} + \sum_{t=0}^{\mathcal{T}} \mu_t \sum_{s=t+1}^{t+p_i} x_{is} - m)$. Pour $\mu^{(k)}$ (μ à la k-ième itération) donné, choisir le sous-gradient $g_t^{(r)} = \delta f(\mu) = \sum_{i=1}^{n} \sum_{s=t+1}^{t+p_i} x_{is} - m$. Si $g_t^{(k)} = 0$ pour tout $t \in [0, \mathcal{T}]$, alors $\mu^{(k)}$ est un vecteur optimal. Sinon, aller à l'étape 3.
3. Soit $\mu^{(k+1)} = \mu^{(k)} + \nu^{(k)} g^{(k)}$. (La procédure pour sélectionner $\nu^{(k)}$ est donnée par la suite).

   $k \leftarrow k + 1$; Aller à l'étape 2.

La procédure pour sélectionner $\nu^{(k)}$ est la suivante :

1. $\nu^{(k)}$ est une série divergente : $\sum_{k=1}^{\infty} \nu^{(k)} = \infty$, $\nu^{(k)} \leftarrow 0$ si $k \leftarrow \infty$.
2. $\nu^{(k)}$ est une série géométrique : $\nu^{(k)} = \frac{|\overline{f} - f(\mu^{(k)})|\rho^k}{||g^{(k)}||^2}$ où $\rho \in (0,1)$ est une constante donnée, et $\overline{f}$ une borne supérieure du problème initial.

Critère d'arrêt

Enfin la dernière étape dans la recherche de borne inférieure est le critère d'arrêt. Il existe de nombreux critères d'arrêt possibles : nombre d'itérations maximum, temps de recherche maximum, évolution du coût de la meilleure solution trouvée. Le premier consiste à s'arrêter une fois que le nombre d'itérations demandées a été effectué. Le second consiste à stopper la recherche au bout d'un certain temps. Enfin le dernier implique l'arrêt de la recherche si le coût de la meilleure solution trouvée n'a pas été amélioré au bout d'un certain nombre d'itérations, c'est le critère d'arrêt retenu.

Borne supérieure Pour évaluer la qualité de la borne inférieure, nous avons développé une recherche tabou simple [75, 76, 52]. La fonction de voisinage consiste à générer un ensemble de solutions à partir d'une solution donnée. La solution initiale est générée par l'heuristique suivante : pour chaque travail, on calcule les deux ratios α_i/p_i et β_i/p_i. En fonction de ces ratios, on détermine l'ensemble des travaux ordonnancés en avance (si $\alpha_i/p_i < \beta_i/p_i$) et l'ensemble des travaux ordonnancés en retard (si $\alpha_i/p_i > \beta_i/p_i$). Les travaux avec $\alpha_i/p_i = \beta_i/p_i$, sont affectés aléatoirement à un des deux ensembles. L'ordonnancement des travaux respecte le V-shape. Pour l'affectation des travaux aux machines, on utilise la règle "machine libre au plus tôt".

La fonction de voisinage est définie comme suit : pour un travail ordonnancé en avance (resp. en retard), on affecte ce travail à une position en retard (en avance). On construit ainsi de nouvelles listes de travaux à ordonnancer en avance et en retard sur les m machines.

Résultats expérimentaux Pour évaluer expérimentalement les performances de la borne inférieure proposée, nous avons généré 100 instances pour chaque configuaration $(p_i, \alpha_i, \beta_i, n \times \gamma)$ où :

- le nombre de travaux considérés est : $n \in \{20, 30, 40, 100, 150\}$
- les durées opératoires ont été générées aléatoirement dans $[10, 100]$
- les pénalités d'avance α_i ont été générées aléatoirement dans $[1, 50]$
- les pénalités de retard β_i ont été générées aléatoirement dans $[1, 50]$

– le coût de la date de fin souhaitée a été généré aléatoirement dans $[10, 100]$

Le critère d'arrêt pour le calcul de la borne supérieure a été fixé à 300 itérations. La taille de la liste tabou est fixée à 10.

Le critère d'arrêt de la borne inférieure a été défini comme suit : arrêter la recherche au bout d'une minute si aucune amélioration de la meilleure solution de plus de 0,1% n'est trouvée. C'est un bon compromis constaté entre le temps de calcul et la qualité de la solution trouvée.

La table 4.2 donne une idée de l'écart entre la borne inférieure et supérieure dans le cas d'une seule machine.

n	nombre d'instances	moyenne(%)	écart-type (%)
20	100	3,7	7,38
30	100	5,26	8,03
40	100	7,47	5,41
100	100	32,04	12,41
150	100	54,32	18,08

TAB. 4.2 – Performance de la borne inférieure par rapport à une borne supérieure

Comme nous le constatons, le gap devient important dès que le nombre de travaux dépasse les 40 travaux. Ces résultats peuvent être expliqués pour les deux raisons suivantes :

– la borne inférieure peut être améliorée : en proposant une heuristique pour déterminer le vecteur μ pour remplacer la méthode du sous-gradient ;
– la borne supérieure peut être améliorée : en augmentant le nombre d'itérations ; en étudiant d'autres fonctions de voisinage ; en ajoutant le critère d'aspiration qui permet de sortir d'un minimum local. Il est même possible de proposer d'autres heuristiques pour la résolution du problème étudié, par exemple en construisant une solution à partir de celle obtenue par la relaxation Lagrangienne.

4.4 Conclusion

Dans ce chapitre, nous avons abordé quelques problème d'ordonnancement JàT où la date de fin souhaitée est une variable de décision. Quelques nouveaux cas polynomiaux ont été identifiés avec durées opératoires identiques et une famille de dates de fin souhaitées contrôlables. Le cas général a été également étudié où principalement, nous avons montré l'équivalence entre le problème CON et le problème JàT avec une

date de fin souhaitée commune non-restrictive, étudié dans le chapitre précédent.

Nous avons proposé une borne inférieure pour le problème $Pm|d_i = d|\sum(\alpha_i E_i + \beta_i T_i + \gamma d)$ basée sur la relaxation Lagrangienne. Cette borne peut être généralisée au cas d'une famille de dates de fin souhaitées.

Il serait intéressant de tester la PSE de F. Sourd [202] pour résoudre le problème CON et comparer par la suite les résultats obtenus avec la borne inférieure proposée. L'étude de l'existence des schémas d'approximation pour le cas de m machines reste un problème ouvert.

Chapitre 5

Conclusion et perspectives de la première partie

Face à la complexité croissante des produits et services, aux fluctuations des prix et à l'instabilité de la situation financière, les objectifs stratégiques des entreprises visent à réduire les coûts, améliorer le service et amener les produits au plus vite sur le marché. De nos jours, on nous parle de plus en plus du concept *Lean* défini par deux piliers fondamentaux : le système JàT et la Qualité de service. Adopter un processus synchrone et équilibrer est une stratégie principale sur laquelle repose le *Lean manufacturing* . "Face aux exigences du marché, on doit penser *Lean* et cela à tous les niveaux décisionnels (stratégique, tactique et opérationnel) et tout au long de la chaîne logistique", telle est la pensée d'un manager, de nos jours. Ainsi, les modèles JàT étudiés dans cette première partie de thèse, peuvent trouver leur place – au niveau opérationnel – dans un tel concept.

En particulier, nous nous sommes appuyés sur des concepts de la théorie de la recherche opérationnelle, afin de mettre en évidence son intérêt pour des applications pratiques des problèmes d'ordonnancement JàT. Nous nous sommes donc intéressés aux critères d'avance-retard des travaux. L'avance et le retard sont mesurés par rapport aux souhaits des clients pour la partie aval (dates de fin souhaitées). Deux types de dates de fin souhaitées ont été considérés :

1. dates de fin souhaitées connues et données par le chef d'atelier, voire par le client qui précise la date à laquelle il souhaite être livré. Selon la valeur de cette date de fin, deux cas se présentent : date de fin restrictive – qui a fait l'objet de l'étude

des chapitres 2 et 3 (rappelons que dans le chapitre 2, les coûts d'avances ont été considérés négligeables)- et date de fin non-restrictive – qui a fait l'objet d'étude du chapitre 3 également.

2. dates de fin souhaitées contrôlables définies comme une date de fin pour l'atelier et qui peut correspondre à une certaine marge permettant de compenser les aléas de production d'une part et le délai de livraison d'autre part.

Comme nous l'avons vu, ces dates de fin peuvent être communes à tous les travaux où à une famille de travaux.

Même si l'ordonnancement JàT a vu ses résultats considérablement approfondis ces dernières années, plusieurs problèmes JàT méritent, de la part des chercheurs, une attention particulière. En effet, si on se focalise uniquement sur des problèmes adjacents à ceux étudiés dans cette partie de thèse, nous remarquons que certaines questions demeurent ouvertes, telles que l'existence des heuristiques à garantie de performance voire des schémas d'approximation polynomiaux (cf. tableau 5.1).

				Unitaire ($\alpha_i = \beta_i = 1$)	Identique ($\alpha_i = \alpha, \beta_i = \beta$)	Symétrique ($\alpha_i = \beta_i = w_i$)	Quelconque (α_i, β_i)
Une seule machine	$d_i = d$	date de fin souhaitée donnée	non-restrictive	$\mathcal{PO}$ [129]	$\mathcal{PO}$ [61]	FPTAS [140]	**PTAS** *
		$\sum_i(\alpha_i E_i + \beta_i T_i)$	restrictive	4/3-approximation [103]	Ouvert	Ouvert	Ouvert
		date de fin souhaitée contrôlable $\sum_i(\alpha_i E_i + \beta_i T_i + \gamma d)$		$\mathcal{PO}$ [169]	$\mathcal{PO}$ [169]	**FPTAS**	**PTAS** *
	$p_i = p$, $d_i \in D$, $\lvert D\rvert = k$,	dates de fin souhaitées données $\sum_i(\alpha_i E_i + \beta_i T_i)$		**PO**			
		dates de fin souhaitées contrôlables $\sum_i(\alpha_i E_i + \beta_i T_i + \gamma d)$		**PO**			
m machines parallèles identiques	$d_i = d$	date de fin souhaitée donnée	non-restrictive	FPTAS [221]	FPTAS [221]	Ouvert	**PTAS** *
		$\sum_i(\alpha_i E_i + \beta_i T_i)$	restrictive	Ouvert	**NPO** $\setminus$ **APX**	Ouvert	**NPO** $\setminus$ **APX**
		date de fin souhaitée contrôlable $\sum_i(\alpha_i E_i + \beta_i T_i + \gamma d)$		$\mathcal{PO}$ [40, 50]	$\mathcal{PO}$ [40, 50]	Ouvert	Ouvert
	$p_i = p$, $d_i \in D$, $\lvert D\rvert = k$,	dates de fin souhaitées données $\sum_i(\alpha_i E_i + \beta_i T_i)$		**PO**			
		dates de fin souhaitées contrôlables $\sum_i(\alpha_i E_i + \beta_i T_i + \gamma d)$		**PO**			

(PTAS *) : PTAS sous l'hypothèse $\alpha_{min} \leq \alpha_i \leq \alpha_{max}$ et $\alpha_{max}/\alpha_{min} \leq 2$

TAB. 5.1 – Bilan et perspectives

Deuxième partie

Ordonnancement de travaux interférant indépendants

Chapitre 6

Introduction à l'ordonnancement des travaux interférants

Mesurer la qualité d'un ordonnancement à l'aide d'un seul critère pour tous les travaux n'est pas toujours pertinent. En pratique, il arrive que le chef d'atelier impose un critère sur l'ensemble des travaux à réaliser au sein de son système de production tout en essayant de minimiser un autre critère imposé par un client vis-à-vis de sa demande, i.e. pour les travaux qui le concernent. Il s'agit donc de l'ordonnancement avec "travaux interférants". Dans ce chapitre, nous présentons le contexte de l'ordonnancement de travaux interférants et quelques exemples industriels. Nous présentons rapidement la définition d'un problème d'ordonnancement multi-critère. Ensuite, l'intérêt de cette étude par rapport aux problèmes étudiés dans la littérature est démontré.

6.1 Motivation et contexte de travail

Classiquement, la mesure de la qualité d'un ordonnancement se base sur un seul critère à optimiser pour l'ensemble des travaux. En effet, les modèles traditionnels d'ordonnancement considèrent que les travaux sont tous équivalents et que la qualité de l'ordonnancement global doit être mesurée selon un seul objectif, appliqué sur tous les

travaux sans aucune distinction. La présentation de distinctions entre les travaux se fait généralement par le moyen de coefficients de pondération. Toutefois, dans ce cas, la même mesure est appliquée à tous les travaux pour quantifier la qualité d'un ordonnancement. Par exemple, il peut s'agir de minimiser le temps de séjour moyen de l'ensemble des travaux ($\sum w_j C_j$) ou encore minimiser un critère lié au retard des travaux ($\sum w_j T_j$, $\sum w_j U_j$).

Cependant, dans un contexte réel, ces modèles d'ordonnancement ne sont pas toujours pertinents. Dans certaines situations concrètes, il peut être nécessaire d'examiner plusieurs aspects de l'ordonnancement, comme le temps de séjour moyen avec une autre mesure relative au respect des dates d'échéance (date de fin impérative), par exemple. Atteindre un bon compromis entre les critères est un challenge. On parle alors de l'ordonnancement multi-critères [209] et [95]. Une fois de plus, dans ces études, chaque fonction objectif est appliquée à l'ensemble des travaux.

Il arrive dans certains cas pratique que l'application de la même mesure pour tous les travaux ne soit pas pertinente. En effet, il est possible d'envisager un atelier où les travaux ont la particularité suivante : des travaux d'un premier sous-ensemble peuvent avoir une date de fin souhaitée avec une possibilité de retard (à réduire au minimum), ceux d'un deuxième sous-ensemble peuvent avoir des dates de fin impératives (à respecter) et d'autres travaux constituant un troisième sous-ensemble n'ont pas de date d'échéance (production pour le stock). Pour le premier sous-ensemble, le décideur veut réduire au maximum les retards ; pour le deuxième sous-ensemble, il ne tolère aucun retard ; pour le dernier type de travaux, il veut réduire le temps de traitement total ou le temps de séjour moyen. L'ordonnancement de chaque sous-ensemble est évalué en fonction de différents objectifs, mais les travaux sont tous en concurrence pour l'utilisation des machines. Il s'agit d'un problème d'ordonnancement multi-critères où un nouveau type de compromis doit être obtenu. Ces problèmes d'ordonnancement sont appelés dans la littérature "*ordonnancement multi-agent*" [5, 37] ou "*ordonnancement avec des agents concurrents*" [6]. Dans ces études, les auteurs considèrent des partitions disjointes de l'ensemble des travaux, et chaque partition a sa propre fonction objectif à optimiser.

Cette notion de l'ordonnancement multi-agent peut être différente dans certaines situations réelles où les sous-ensembles ne sont pas forcément disjoints. Nous considérons dans cette partie un autre type de problème où une fonction objectif porte sur

l'ensemble des travaux, et une autre fonction objectif porte sur un sous-ensemble des travaux.

Un tel problème apparaît dans des situations réelles. Par exemple, l'atelier de fabrication des roulements à billes SKF MDGBB (*Medium Deep Groove Ball Bearings*) est composés de machines parallèles [174, 175]). L'objectif principal (global) est la régularisation du flux de sortie de la ligne de production ($\sum C_i$). Toutefois, pour une partie de la production (les travaux qui définissant le sous-ensemble $\mathcal{N}_1$) une autre mesure de performance peut être appliquée, comme la réduction du nombre de travaux en retard (ou de toute autre mesure relative aux dates dues). Mais la mesure sur l'ensemble des travaux – y compris $\mathcal{N}_1$ – reste valable. La mesure concernant $\mathcal{N}$ est par exemple la minimisation du temps total d'achèvement, tandis que le nombre de travaux en retard parmi les travaux de $\mathcal{N}_1$ ne peut pas dépasser un seuil donné. On parle alors d'"ordonnancement de travaux interférants".

Un autre exemple peut être trouvé dans les industries d'emballage de shampoings [160]. Les shampoings sont livrés quotidiennement et stockés dans des cuves de capacité limitée. Le problème est de placer différents types de shampooing en bouteilles. Un objectif est de maximiser la production globale (réduction des temps de réglage). D'autre part, les livraisons à venir sont connues à l'avance ainsi que les demandes. Ainsi, chaque type de shampoing doit être produit de sorte que leur quantité en stock ne dépasse jamais la capacité de la cuve. Il s'agit d'un problème typique où l'objectif global concerne tous les produits et où chaque sous-ensemble de produits est évalué par un autre objectif.

6.2 Brève introduction de l'ordonnancement multi-critère [1]

6.2.1 Généralités

Nous présentons ici une brève introduction de l'ordonnancement multi-critère (cette partie est extraite de [209]).

Par nature, un problème d'ordonnancement dans le contexte de la production est très souvent multi-critère. Dans un contexte en dehors de la production, l'aspect mono ou multi-critère dépend du problème. En règle générale, et comme le souligne [188], la

[1] Nous remercions le Professeur Vincent T'kindt du Laboratoire d'Informatique de Tours pour son aide précieuse à la rédaction de cette section

prise en compte de plusieurs critères permet de fournir au Décideur une solution plus réaliste.

On trouve dans la littérature différents états de l'art sur les problèmes d'ordonnancement multi-critères (voir [58], [69], [96], [166], [209]). L'analyse de ces travaux fait ressortir :

- la nécessité de connaître les résultats du domaine de l'optimisation multi-critère pour bien appréhender les difficultés liées à la prise en compte de critères conflictuels,
- le besoin d'une typologie permettant de formaliser les différents types de problèmes possibles et d'homogénéiser la notation de ces problèmes.

Nous allons maintenant présenter une décomposition des problèmes d'ordonnancement multi-critères en mettant en évidence ce qui est plus du ressort d'une démarche de l'Aide à la Décision multi-critère, de l'Optimisation multi-critère et de l'Ordonnancement.

Définition 6.2.1 *On appelle* **problème d'ordonnancement multi-critère** *le problème qui consiste à déterminer un ordonnancement qui optimise plusieurs critères. Cet ordonnancement est un optimum de Pareto. Ce problème se décompose en trois sous-problèmes :*

1. *la* **modélisation du problème**, *où il s'agit de déterminer la nature du problème d'ordonnancement considéré ainsi que de définir les critères à prendre en compte,*
2. *la* **prise en compte des critères**, *où il s'agit d'indiquer le contexte de résolution et la façon dont on souhaite prendre en compte les critères. L'Homme d'Etude met au point un module d'aide à la décision pour le problème multi-critère (ou module de prise en compte des critères),*
3. *l'***ordonnancement** *où il s'agit de trouver une solution au problème qui optimise la fonction objectif construite. L'Homme d'Etude met au point un algorithme de résolution du problème d'ordonnancement (ou module de résolution du problème d'ordonnancement).*

La **modélisation du problème** ([188]) se fait avec le Décideur, et consiste à définir les critères pertinents à prendre en compte. On suppose que ces critères sont conflictuels c'est-à-dire que minimiser un critère n'est pas équivalent à en minimiser un autre. D'autre part, on définit lors de cette phase l'environnement dans lequel se situe le problème d'ordonnancement, c'est-à-dire l'ensemble des ressources disponibles pour réaliser les travaux (il peut s'agir de machines et d'hommes dans le cas d'un atelier, ou

d'un autre type de ressources), et de quelles façons ces ressources sont organisées. On identifie en quelque sorte la nature du problème d'ordonnancement. Enfin, on définit les contraintes particulières liées au problème : préemption autorisée ou non, dates de début au plus tôt, *etc.*

Une grande partie de la difficulté pour résoudre un problème multi-critère se situe dans la **prise en compte des critères**. Lors de cette phase, le Décideur donne des informations concernant sa perception des critères : il dit d'abord s'il autorise ou non des compensations entre les critères, c'est-à-dire s'il autorise des solutions de compromis. Si cela n'est pas le cas, il indique l'ordre dans lequel les critères doivent être optimisés. En revanche, si les compensations sont permises, il indique s'il est possible des pondérer les critères (si cela a un sens), et il donne éventuellement les poids, il indique les objectifs à atteindre pour chaque critère s'il les connaît, *etc.* Ensuite, il oriente le choix de la méthode en indiquant s'il souhaite un algorithme qui lui donne une solution unique, compte tenu des informations fournies, ou s'il préfère intervenir dans la procédure de résolution parce qu'il ne sait pas très bien répondre aux questions, ou encore s'il désire voir toutes les solutions possibles pour retenir celle qui l'intéresse. Ces choix orienteront la méthode respectivement vers un algorithme *a priori*, *interactif* ou bien *a posteriori*.
Les informations récoltées lors de cette phase sur le problème et sur la façon dont le Décideur peut et souhaite l'aborder vont permettre à l'Homme d'Etude de choisir une approche de résolution appropriée, que ce soit une combinaison linéaire des critères, une approche paramétrique ou une autre méthode de son choix. Il en résulte la forme de la fonction objectif du problème d'ordonnancement pour lequel il faut proposer un algorithme de résolution. Cette phase va généralement conduire l'Homme d'Etude à mettre au point un module de prise en compte des critères, c'est-à-dire un algorithme qui va réaliser les interactions demandées par le Décideur (mise en oeuvre d'une procédure interactive, a posteriori, *etc.*).

L'ordonnancement a pour objectif de fournir un ordonnancement qui optimise la fonction objectif définie à l'étape précédente. La solution obtenue est un optimum de Pareto pour le problème d'ordonnancement multi-critère. L'Homme d'Etude va donc devoir mettre au point un module d'ordonnancement qui va résoudre le problème d'ordonnancement résultant des étapes précédentes.

Il est important de souligner que [66] préconisait déjà de bien décomposer les travaux. Il précisait ce qui était de la compétence du chercheur opérationnel et ce qui n'en

était pas. Il conseillait éventuellement d'aborder les problèmes sous un angle multi-critère, avec l'aide de l'analyse multi-critère.

La figure 6.1 présente la décomposition des *problèmes d'ordonnancement multi-critères* telle qu'elle est développée dans la définition 6.2.1.

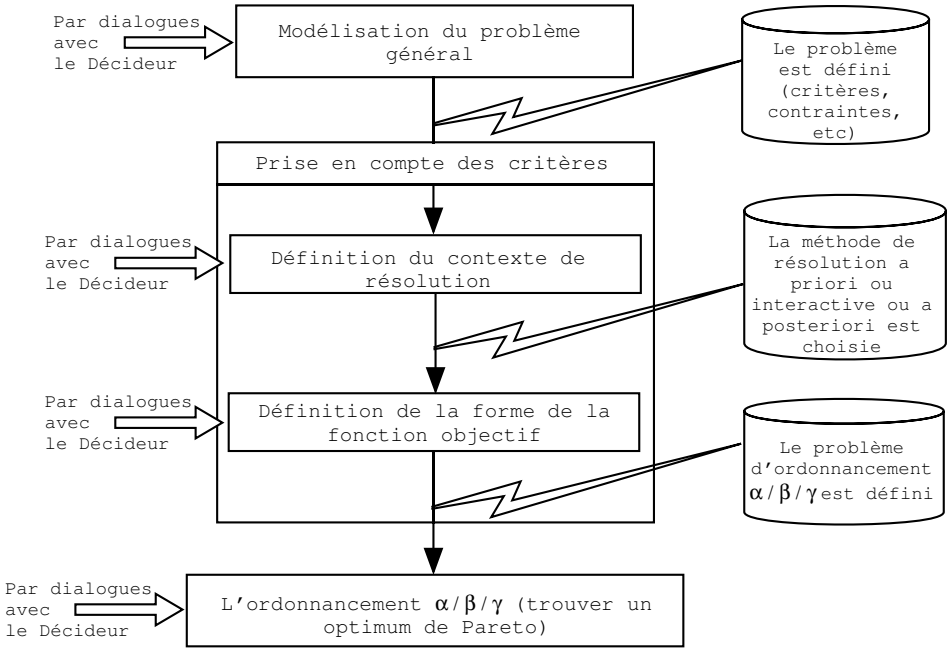

FIG. 6.1 – L'étude des problèmes d'ordonnancement multicritères [209]

Lors de la phase de *prise en compte des critères* et suite aux informations dont il dispose, l'Homme d'Etude choisit une approche de résolution du problème d'ordonnancement et définit ainsi un problème d'ordonnancement. Compte tenu de la diversité des méthodes pour déterminer des optima de Pareto, les fonctions à optimiser pour le problème d'ordonnancement peuvent prendre un nombre varié de formes en fonction des informations fournies par le Décideur.

Par rapport à la notation à trois champs de [80] $(\alpha|\beta|\gamma)$, seul le champ γ est étendu. Pour ce champ, on reprend ici les fonctions telles qu'elles sont présentées dans [208, 209] :

- Z si l'objectif est de minimiser l'unique critère Z (problème monocritère). Il s'agit du cas connu, *i.e.* Z peut valoir C_{max}, T_{max}, *etc.*
- $Z_1, Z_2, \ldots, Z_K$ indique le problème général de détermination de tous les optima de Pareto. On associe généralement à ce problème un algorithme de résolution a posteriori (énumération de l'ensemble E).
- $F_\ell(Z_1, \ldots, Z_K)$ si l'objectif est de minimiser une combinaison linéaire (convexe)

des K critères. Par exemple, ce cas convient particulièrement si le Décideur peut donner un poids pour chaque critère,

- $\varepsilon(Z_u / Z_1, \ldots, Z_{u-1}, Z_{u+1}, \ldots, Z_K)$, indique que seul le critère Z_u est minimisé, sachant que tous les autres critères sont bornés supérieurement par une valeur connue. Ce cas se distingue du cas précédent car la fonction à minimiser est Z_u et que ce critère n'est sujet à aucune contrainte de bornitude. L'Homme d'Etude se situe dans le cadre de l'approche ϵ-contrainte,
- $GP(Z_1, Z_2, \ldots, Z_K)$, pour la programmation par objectifs. Le problème n'est pas d'optimiser un critère, mais de trouver une solution qui satisfait les objectifs, même si cette solution ne correspond pas à un optimum de Pareto,
- $F_T(Z_1, \ldots, Z_K)$, $F_{Tp}(Z_1, \ldots, Z_K)$ et $F_{Tpa}(Z_1, \ldots, Z_K)$ indiquent une fonction objectif qui est l'expression d'une distance à une solution idéale connue (non atteignable), en utilisant pour le calcul de la distance la métrique de Tchebycheff, la métrique de Tchebycheff pondérée ou la métrique de Tchebycheff pondérée augmentée.
- $F_s(Z_1, \ldots, Z_K)$ pour l'approche de satisfaction de l'objectif,
- $Lex(Z_1, Z_2, \ldots, Z_K)$ si l'Homme d'Etude utilise l'ordre lexicographique,
- $P(Z_1, \ldots, Z_K)$ dans le cadre de l'analyse paramétrique.

6.2.2 Classes de méthodes de résolution

Nous avons vu (Figure 6.1) que l'Homme d'Etude doit résoudre trois problèmes pour proposer un algorithme de résolution au Décideur. Selon les réponses qu'il a eu, il va concevoir un algorithme de résolution plutôt qu'un autre. On distingue trois cas :

1. Le Décideur souhaite que l'outil lui fournisse une unique solution, autrement dit il choisit une méthode *à priori*. Dans ce cas, l'algorithme que réalise l'Homme d'Etude est composé d'un module qui saisit la valeur des paramètres (poids, *etc.*), permettant de donner une occurrence au problème d'ordonnancement. Les paramètres sont passés au second module, chargé de résoudre le problème d' ordonnancement, et la solution obtenue est retournée au Décideur.
2. Le Décideur souhaite intervenir dans le processus de résolution du problème d'ordonnancement, autrement dit il choisit une méthode *interactive*. Dans ce cas, le premier module de l'algorithme va dialoguer avec lui pour déterminer la nouvelle voie de recherche, et à chaque itération, le second module va résoudre le problème d'ordonnancement et retourner la solution calculée. Si celui-ci souhaite

poursuivre la recherche, une nouvelle direction de recherche est indiquée et le processus est itéré.

3. Enfin, si le Décideur préfère choisir la solution dans un ensemble d'optima de Pareto, il se place dans le cadre d'une méthode *a posteriori*. Le premier module va faire varier les paramètres de la fonction objectif pour que le second module puisse calculer l'ensemble des optima de Pareto. Cet ensemble est retourné au Décideur.

La figure 6.2 représente les trois méthodes de recherche d'une solution, avec les deux phases qui la constituent.

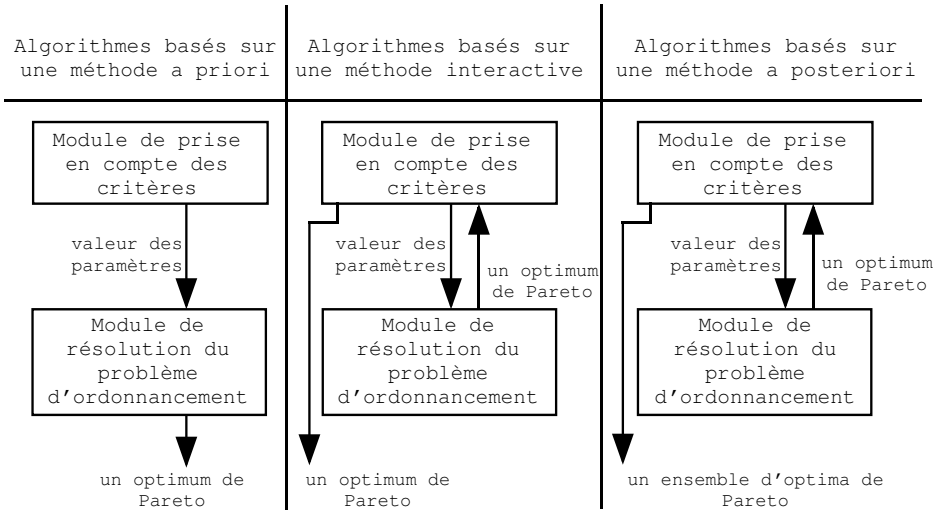

FIG. 6.2 – Les méthodes de résolution de problèmes multicritères [209]

Dans la cadre de cette thèse, nous ne considérons que les méthodes a priori et par la suite, on s'intéresse à un point de l'ensemble des solutions non-dominées (l'enveloppe convexe des optima Pareto strict).

6.3 Définition de l'ordonnancement avec des travaux interférants

Un ensemble $\mathcal{N}$ de n travaux doit être ordonnancé sur une seule machine ou m machines parallèles ($m \geq 2$). Tous les travaux sont disponibles à la date zéro ; aucune préemption n'est autorisée ; les durées opératoires des travaux sont données en nombres entiers, p_j est la durée opératoire de travail J_j, $1 \leq j \leq n$; les machines sont toujours disponibles et peuvent exécuter au plus un travail à un instant donné.

Soit $\mathcal{N}_1$ un sous-ensemble de $\mathcal{N}$. Notons n_1 le nombre de travaux de $\mathcal{N}_1$. Ces travaux sont numérotés de 1 à n_1. Les travaux restant de $\mathcal{N}$ sont numérotés de n_1+1 à n. Une fonction objectif est associée à $\mathcal{N}$ et une autre est associée à $\mathcal{N}_1$.

Nous notons $Z(\mathcal{S})$ la measure Z appliquée à l'ensemble $\mathcal{S}$ des travaux. Nous considérons par la suite deux types de fonction objectif :

- $\varepsilon\big(Z_1(\mathcal{S})/Z_2(\mathcal{S}')\big)$ désigne l'approche ε-contrainte, i.e. minimisation de $Z_1(\mathcal{S})$ sous réserve que $Z_2(\mathcal{S}') \leq \varepsilon$. Par la suite nous considérons que l'objectif est de minimiser une fonction objectif sur $\mathcal{N}$ sous la contrainte que la fonction objectif sur $\mathcal{N}_1$ est bornée : $\varepsilon\big(Z_1(\mathcal{N})/Z_2(\mathcal{N}_1)\big)$ (le cas inverse, $\varepsilon\big(Z_1(\mathcal{N}_1)/Z_2(\mathcal{N})\big)$, peut être déduit facilement). Notons que si deux fonctions objectif sont bornées, nous sommes dans le cas de l'approche "*goal programming*". Le problème de décision associé à l'approche ε-contrainte d'un problème d'optimisation et le problème de décision associé à l'approche "goal programming" sont les mêmes.
- $F_\ell\big(Z_1(\mathcal{S}), Z_2(\mathcal{S}')\big)$ désigne la combinaison linéaire entre $Z_1(\mathcal{S})$ et $Z_2(\mathcal{S}')$.

6.4 État de l'art

Dans la littérature, très peu de résultats sont consacrés à l'ordonnancement de travaux interférants. Dans [119], les auteurs s'intéressent au $F2||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ et $F2||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$. Les auteurs montrent que le problème est NP-difficile au sens ordinaire et un algorithme pseudo-polynomial de programmation dynamique est proposé. Il est à noter que le problème étudié dans [119] est le cas général du problème d'ordonnancement multi-agent présenté dans [6], qui peut être résolu par ce même algorithme. Le problème $P||\sum C_j(\mathcal{N}_1), \sum C_j(\mathcal{N})$ a été étudié dans [198, 197]. Les auteurs montrent que le problème est NP-difficile au sens ordinaire et proposent un algorithme pseudo-polynomial de programmation dynamique. Une méthode exacte de type procédure par séparation et évaluation et une heuristique efficace ont été développées.

Concernant les modèles multi-agents, dans [5], les auteurs considèrent le job-shop avec seulement deux travaux sur deux sous-ensembles de travaux disjoints $\mathcal{N}_1$ et $\mathcal{N}_2$ ($\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}$ et $\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ dans ce qui suit). Chaque agent se voit confier un sous-ensemble. Ils donnent un algorithme polynomial afin de trouver des ordonnancements de compromis pour la simplification des négociations entre agents.

Le cas d'une seule machine avec agents concurrents a été largement étudié. Citons

par exemple les travaux publiés dans [18], [6], [227], [37] et [38].

Dans [18], les auteurs cherchent à minimiser une combinaison linéaire de trois critères réguliers ($C_{\max}, \sum w_j C_j, L_{\max}$). Quelques résultats de complexité et des cas polynomiaux ont été illustrés. Par la suite Yuan et al. [227] proposent des résultats complémentaires aux travaux de Baker et Smith [18].

Dans [6], les auteurs s'intéressent également au flow shop et open shop avec deux sous-ensembles disjoints $\mathcal{N}_1$ et $\mathcal{N}_2$. Ils considèrent la minimisation d'une fonction objectif pour un sous-ensemble des travaux en respectant une borne sur une function objectif pour un autre sous-ensemble des travaux. Ils donnent des résultats de complexité et des algorithmes de programmation dynamique.

Dans [37] les auteurs s'intéressent au cas de m sous-ensembles disjoints de travaux $\mathcal{N}_1, ..., \mathcal{N}_m$ ($\cup_{i=1}^{m} \mathcal{N}_i = \mathcal{N}$). Chaque travail dispose d'une date d'échéance. Chaque sous-ensemble est mesuré par le nombre total de travaux en retard. Ce problème de "*goal programming*", se note $1||GP\left(\sum w_j U_j(\mathcal{N}_1), ..., \sum w_j U_j(\mathcal{N}_m)\right)$ ou $1|\sum w_j U_j(\mathcal{N}_1) \leq \varepsilon_1, ..., \sum w_j U_j(\mathcal{N}_m) \leq \varepsilon_m|-$. Le problème est montré NP-difficile au sens fort. Cependant, il peut être résolu en temps pseudo-polynomial lorsque le nombre d'agents (m) est fixé et ils proposent un schéma d'approximation totalement polynomial (FPTAS). Si les poids sont unitaires de plus, le problème est résolu en temps polynomial.

Le cas de multi-agent avec m fonctions objectif de type MinMax est étudié dans [38]. Les auteurs considèrent une approche de goal programming et montrent que le problème de faisabilité peut être résolu en temps polynomial même si les travaux sont soumis à des contraintes de précédence. Les problèmes $1||\sum_{i=1}^{m}\left(L_{\max}(\mathcal{N}_i)\right)$, $1||\sum_{i=1}^{m}\left(T_{\max}(\mathcal{N}_i)\right)$ et $1||\sum_{i=1}^{m}\left(\sum w_j C_j(\mathcal{N}_i)\right)$ sont montrés NP-difficiles. Quelques cas polynomiaux ont été identifiés.

Dans [7], les auteurs considèrent les problèmes d'ordonnancement sur une seule machine avec deux agents, indiqué dans la Figure 6.1. Deux approches sont considérées : (1) le problème de décision afin de trouver une solution réalisable telle que tous les critères sont bornés (goal programming approche dénotée par GP) et (2) le problème d'optimization Pareto où l'objectif est de trouver l'ensemble de toutes les solutions non-dominées (denoté par "#" dans [209]). Certains résultats sont également donnés pour certains problèmes d'ordonnancement multi-agents sur une seule machine.

Problem	Complexity	Reference
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}_2)\big)$	Polynomial	[18]
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), L_{max}(\mathcal{N}_2)\big)$	Polynomial	[18]
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}_2)\big)$	Polynomial	[18]
$1\|\|F_l\big(L_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2)\big)$	Polynomial	[18],[227]
$1\|\|F_l\big(\sum C_j(\mathcal{N}_1), L_{\max}(\mathcal{N}_2)\big)$	Polynomial	[227]
$1\|\|F_l\big(\sum w_j C_j(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}_2)\big)$	Polynomial	[18]
$1\|\|F_l\big(\sum w_j C_j(\mathcal{N}_1), L_{\max}(\mathcal{N}_2)\big)$	Strongly NP-hard	[18]
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2), \sum w_j C_j(\mathcal{N}_3)\big)$	Strongly NP-hard	[18]
$1\|\|\sum_{i=1}\big(L_{\max}(\mathcal{N}_i)\big)$	Binary NP-Hard	[38]
$1\|\|\sum_{i=1}\big(T_{\max}(\mathcal{N}_i)\big)$	Binary NP-Hard	[38]
$1\|\|\sum_{i=1}\big(\max w_j C_j(\mathcal{N}_i)\big)$	Strongly NP-Hard	[38]
$1\|\|\varepsilon\big(f_{\max}(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[6]
$1\|\|\varepsilon\big(\sum C_j(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[6]
$1\|\|\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N}_2)\big)$	Binary NP-Hard	[6]
$1\|\|\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Open*	
$1\|\|\varepsilon\big(\sum w_j C_j(\mathcal{N}_1)/C_{\max}(\mathcal{N}_2)\big)$	Binary NP-Hard	[6]
$1\|\|\varepsilon\big(\sum w_j C_j(\mathcal{N}_1)/L_{\max}(\mathcal{N}_2)\big)$	Strongly NP-Hard	[168]
$1\|\|\varepsilon\big(\sum w_j C_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Strongly NP-Hard	[168]
$1\|\|\varepsilon\big(\sum w_j C_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Binary NP-Hard	[6]
$1\|\|\varepsilon\big(\sum U_j(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[6]
$1\|\|\varepsilon\big(\sum U_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Polynomial	[6]
$F2\|\|\varepsilon\big(\mathrm{Cmax}(\mathcal{N}_1)/\mathrm{Cmax}(\mathcal{N}_2)\big)$	Binary NP-Hard	[6],[119]
$O2\|\|\varepsilon\big(\mathrm{Cmax}(\mathcal{N}_1)/\mathrm{Cmax}(\mathcal{N}_2)\big)$	Binary NP-Hard	[6]
$1\|\|GP\big(f_{\max}(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|GP\big(\sum C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|GP\big(\sum C_j(\mathcal{N}_1), \sum C_j(\mathcal{N}_2)\big)$	Binary NP-hard	[7]
$1\|\|GP\big(\sum C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Open*	
$1\|\|GP\big(\sum w_j C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Binary NP-hard	[7]
$1\|\|GP\big(\sum w_j C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Binary NP-hard	[7]
$1\|\|GP\big(\sum U_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|GP\big(\sum U_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|GP\big(\sum U_j(\mathcal{N}_1), ..., \sum U_j(\mathcal{N}_m)\big)$	Polynomial	[37]
$1\|\|GP\big(\sum w_j U_j(\mathcal{N}_1), ..., \sum w_j U_j(\mathcal{N}_m)\big)$	Binary NP-hard	[37]
$1\|\|GP\big(\sum w_j U_j(\mathcal{N}_1), ..., \sum w_j U_j(\mathcal{N}_i), ...\big)$	Strongly NP-hard	[37]
$1\|\|\#\big(f_{\max}(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|\#\big(\sum C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|\#\big(\sum C_j(\mathcal{N}_1), \sum C_j(\mathcal{N}_2)\big)$	Exponential	[7]
$1\|\|\#\big(\sum C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|\#\big(\sum w_j C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Exponential	[7]
$1\|\|\#\big(\sum w_j C_j(\mathcal{N}1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|\#\big(\sum U_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	[7]
$1\|\|\#\big(\sum U_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	[7]
$J\|n=2\|\#\big(g(\mathcal{N}_1), g(\mathcal{N}_2)\big)$	Polynomial	[5]

- with $f_{\max}$ a regular function of type $\max_j(f_j(C_j))$
- with g a non regular function
(*) The problem is NP-Hard under high multiplicity encoding [168]

TAB. 6.1 – Résultats de complexité de l'ordonnancement multi-agent [121]

6.5 Intérêt de l'étude

Nous montrons dans cette partie que le problème que nous abordons n'est pas équivalent au problème multi-agents.

Soit F le problème d'optimisation avec deux fonctions objectif $f_1(\mathcal{N}_1)$ et $f_2(\mathcal{N}_2)$ concernant deux sous-ensembles disjoints de travaux $\mathcal{N}_1$ et $\mathcal{N}_2$ ($\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ et $\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}$).

Soit G le problème d'optimisation avec deux fonctions objectif : $g_1(\mathcal{N})$ sur tout l'ensemble des travaux et $g_2(\mathcal{N}_1)$ sur $\mathcal{N}_1$. Si f_1 est de la forme de min-sum, elle est notée par sf_1 et par mf_1 si elle est de la forme de min-max (même notation pour f_2, g_1 et g_2). Nous avons $sf_1 \in \{\sum C_j, \sum w_j C_j, \sum T_j, \sum w_j T_j, \sum U_j, \sum w_j U_j\}$ et $mf_1 \in \{C_{\max}, L_{\max}, T_{\max}\}$ (de même pour f_2, g_1 et g_2).

Nous allons expliquer la différence entre les problèmes F et G. Nous distinguons dans cette section deux approches possibles : la minimisation d'une combinaison linéaire et le cas d'une approche "goal programming" (même problème de décision que pour l'approche ε-contrainte).

Notons que nous avons $sg_1(\mathcal{N}) = sg_1(\mathcal{N}_1) + sg_1(\mathcal{N}_2)$ et $mg_1(\mathcal{N}) = \max(mg_1(\mathcal{N}_1), mg_1(\mathcal{N}_2))$.

Nous avons les résultats préliminaires simples suivants : si $f_1 \not\equiv g_2$ ('f_1 n'est pas similaire à g_2', i.e. $sf_1 \neq sg_2$ et $mf_1 \neq mg_2$) ou $f_2 \not\equiv g_1$, alors les problèmes F et G ne sont pas comparables. De plus, si f_2 est sous la forme de mf_2, alors F et G ne sont pas comparables. Par la suite, nous supposons que $f_1 \equiv g_2$ ($sf_1 = sg_2$ or $mf_1 = mg_2$) et $sf_2 = sg_1$.

Il ne reste que deux cas à considérer :

1. $sf_1 = sg_2$ et $sf_2 = sg_1$
2. $mf_1 = mg_2$ et $sf_2 = sg_1$

Nous distinguons la combinaison linéaire des critères et l'approche "goal programming".

– Cas de l'approche de combinaison linéaire pour $sf_1 = sg_2$ et $sf_2 = sg_1$.
  La fonction objectif de F est Min $Z = \alpha \sum f_1(\mathcal{N}_1) + \beta \sum f_2(\mathcal{N}_2)$ et celle de G est Min $Z' = \alpha' \sum g_1(\mathcal{N}) + \beta' \sum g_2(\mathcal{N}_1) = \alpha' \sum f_2(\mathcal{N}) + \beta' \sum f_1(\mathcal{N}_1)$. Ainsi, nous avons $Z' = \alpha' \sum f_2(\mathcal{N}_2) + \alpha' \sum f_2(\mathcal{N}_1) + \beta' \sum f_1(\mathcal{N}_1)$. Si sf_1 et sf_2 ne sont pas identiques, il n'est pas possible de comparer ces fonctions objectif et les problèmes

ne sont pas comparables. Néanmoins, si $sf_1 = sf_2$ (et donc $= sg_2 = sg_1$), alors $Z' = \alpha' \sum f_1(\mathcal{N}_2) + (\alpha' + \beta') \sum f_1(\mathcal{N}_1)$. Dans ce cas particulier, les problèmes sont équivalents, une procédure pour résoudre F peut résoudre G et inversement.

- Cas de l'approche "goal programming" pour $sf_1 = sg_2$ et $sf_2 = sg_1$.
  Le problème F est de trouver une solution S telle que $\sum f_1(\mathcal{N}_1) \leq \varepsilon_1$ et $\sum f_2(\mathcal{N}_2) \leq \varepsilon_2$. Le problème G est de déterminer une solution S' telle que $\sum f_2(\mathcal{N}_1) + \sum f_2(\mathcal{N}_2) \leq \varepsilon'_1$ et $\sum f_1(\mathcal{N}_1) \leq \varepsilon'_2$. Nous montrons que ces problèmes ne sont jamais comparables. Si nous avons $sf_2 \neq sf_1$, alors $\sum f_2(\mathcal{N}_1) + \sum f_2(\mathcal{N}_2)$ est en rapport avec $\sum f_2(\mathcal{N}_1)$, qui n'est pas pris en considération dans le problème F.
  Par conséquent, F et G ne sont pas comparables. Nous considérons maintenant le cas où $sf2 = sf_1 = sg_2 = sg_1$. Il est clair que si $\varepsilon_1 \neq \varepsilon'_2$ ou $\varepsilon'_1 \neq \varepsilon_1 + \varepsilon_2$, alors F et G ne sont pas comparables. Si $\varepsilon_1 = \varepsilon'_2$ et $\varepsilon'_1 = \varepsilon_1 + \varepsilon_2$, supposons que S est une solution pour le problème F, S est ainsi une solution pour le problème G. Mais la réciproque n'est pas vraie :
  - si $sf = \sum_j C_j : \mathcal{N} = \{1,2,3\}$, $\mathcal{N}_1 = \{1,2\}$, $\mathcal{N}_2 = \{3\}$, $p_1 = 1$, $p_2 = 2$, $p_3 = 3$, $\varepsilon_1 = 8$, $\varepsilon_2 = 3$ ($\varepsilon_1 + \varepsilon_2 = 11$). G a une solution réalisable $S' = (1,3,2)$ avec $\sum_j C_j(\mathcal{N}_1) = 7$ et $\sum_j C_j(\mathcal{N}) = 11$. F n'a pourtant aucune solution réalisable. Par conséquent, l'inverse n'est pas vrai non plus pour tout $sf \in \{\sum w_j C_j, \sum T_j, \sum w_j T_j\}$.
  - si $sf = \sum_j U_j : \mathcal{N} = \{1,2,3\}$, $\mathcal{N}_1 = \{3\}$, $\mathcal{N}_2 = \{1,2\}$, $p_1 = 1$, $p_2 = 2$, $p_3 = 3$, $d_1 = d_2 = d_3 = 1$, $\varepsilon_1 = 2$, $\varepsilon_2 = 0$ ($\varepsilon_1 + \varepsilon_2 = 2$). G a une solution réalisable $S' = (1,3,2)$ mais F n'a aucune solution réalisable. Par conséquent, l'inverse n'est pas vraie pour $sf = \sum w_j U_j$.

  Donc, ces problèmes ne sont pas comparables.
- Cas de l'approche de combinaison linéaire pour $mf_1 = mg_2$ et $sf_2 = sg_1$.
  La fonction objectif du problème F est Min $Z = \alpha \max f_1(\mathcal{N}_1) + \beta \sum f_2(\mathcal{N}_2)$ et celle du problème G est Min $Z' = \alpha' \sum f_2(\mathcal{N}) + \beta' \max f_1(\mathcal{N}_1) = (\alpha' \sum f_2(\mathcal{N}_2) + \beta' \max f_1(\mathcal{N}_1)) + \alpha' \sum f_2(\mathcal{N}_1)$. Le terme $\sum f_2(\mathcal{N}_1)$ n'est pas considéré dans la fonction objectif du problème F et donc les problèmes ne sont pas comparables.
- Cas de l'approche "goal programming" pour $mf_1 = mg_2$ et $sf_2 = sg_1$.
  Problème F est de trouver une solution S telle que $\max f_1(\mathcal{N}_1) \leq \varepsilon_1$ et $\sum f_2(\mathcal{N}_2) \leq \varepsilon_2$. Problème G est de trouver une solution S' telle que $\sum f_2(\mathcal{N}_1) + \sum f_2(\mathcal{N}_2) \leq \varepsilon'_1$ et $\max f_1(\mathcal{N}_1) \leq \varepsilon'_2$. Pour la même raison, problèmes F et G ne sont pas compa-

rables.

Par conséquent, les problèmes F et G sont équivalents si et seulement Z_1, Z_2, Z_3 et Z_4 sont les mêmes et sous la forme de min-sum et si l'approche multi-critère est une combinaison linéaire.

Chapitre 7

Problème d'ordonnancement à une seule machine

Nous considérons dans ce chapitre les problèmes d'ordonnancement sur une seule machine avec des travaux interférants. Nous nous intéressons aux critères régulières simples ($C_{\max}$, $L_{\max}$, $\sum_j w_j C_j$, $\sum U_j$, $\sum T_j$, $\sum w_j T_j$) sur l'ensemble de tous les travaux $\mathcal{N}$ et sur un sous-ensemble des travaux $\mathcal{N}_1$. Il s'agit des problèmes d'ordonnancement multi-critères sous certaines approches – ε-contrainte, programmation par objectif et combinaison linéaire. Quelques problèmes sont montrés polynomiaux, d'autres sont montrés $\mathcal{NP}$-difficiles soit au sens ordinaire soit au sens fort. Des problèmes ouverts sont également présentés.

7.1 Introduction

Comme le contexte des travaux interférants n'a pas beaucoup été étudié dans la littérature, les preuves de complexité de certains problèmes d'ordonnancement seront abordées. Ces problèmes peuvent être identifiés par une des trois approches multi-critères ([209]) : ε-contrainte, programmation par objectif et combinaison linéaire de critères. En particulier, on s'appuie sur les critères réguliers tels que le makespan $C_{\max}$, le maximum de retard $L_{\max}$, la somme des dates de fin d'exécution $\sum C_j$ (pondérées -

$\sum w_j C_j$), le nombre des travaux en retard $\sum U_j$ et la somme des retards $\sum T_j$ (pondérées - $\sum w_j T_j$). La fonction objectif est définie par un critère sur l'ensemble global des travaux et un autre critère (ou sous contrainte d'un critère selon l'approche utilisée) sur un sous-ensembles des travaux.

La table 7.1 présente nos nouveaux résultats :

Fonctions objectifs	Approches étudiées			Références
	(F_ℓ)	(ε)	(GP)	
$C_{\max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.1
$C_{\max}(\mathcal{N}), L_{max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.1
$C_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.1
$C_{\max}(\mathcal{N}), \sum U_j(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.1
$L_{max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.2
$L_{\max}(\mathcal{N}), L_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.2
$L_{\max}(\mathcal{N}), \sum C_j(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.1
$L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)$	SNP	SNP	SNP	Proposition 7.4.1 et 7.4.2
$L_{\max}(\mathcal{N}), \sum U_j(\mathcal{N}_1)$	Ouvert	BNP	BNP	Proposition 7.5.1 et 7.3.1
$\sum C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Proposition 7.2.1
$\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1)$	Polynomial	BNP	BNP	Proposition 7.2.2 et 7.3.1
$\sum C_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)$	Ouvert	Ouvert	Ouvert	Proposition 7.5.1
$\sum w_j C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)$	Polynomial	BNP	BNP	Propositions 7.2.2 et 7.3.1
$\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)$	SNP	SNP	SNP	Propositions 7.4.1 et 7.4.2
$\sum w_j C_j(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)$	Polynomial	SNP	SNP	Propositions 7.2.1 et 7.4.1
$\sum w_j C_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)$	SNP	SNP	SNP	Propositions 7.4.1 et 7.4.2
$\sum U_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)$	Ouvert	Ouvert	Ouvert	Proposition 7.5.1
$\sum U_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)$	Ouvert	BNP	BNP	Proposition 7.5.1 et 7.3.1
$\sum U_j(\mathcal{N}), \sum C_j(\mathcal{N}_1)$	Ouvert	Ouvert	Ouvert	Proposition 7.5.1
$\sum U_j(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)$	SNP	SNP	SNP	Propositions 7.4.1 et 7.4.2
$\sum U_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)$	Ouvert	Ouvert	Ouvert	Proposition 7.5.1
$\sum T_j(\mathcal{N}), f(\mathcal{N}_1)$	BNP	BNP	BNP	Proposition 7.3.1
$\sum f(\mathcal{N}), \sum T_j(\mathcal{N}_1)$	BNP	BNP	BNP	Proposition 7.3.1
$\sum w_j T_j(\mathcal{N}), f(\mathcal{N}_1)$	SNP	SNP	SNP	Proposition 7.4.1
$\sum f(\mathcal{N}), \sum w_j T_j(\mathcal{N}_1)$	SNP	SNP	SNP	Proposition 7.4.1

(F_ℓ) : approche par combinaison linéaire
(ε) : approche ε-contrainte
(GP) : approche de 'goal programming'
(#) : approche d'énumération Pareto
(BNP) : $\mathcal{NP}$-difficile au sens ordinaire
(SNP) : $\mathcal{NP}$-difficile au sens fort
(f) : $f \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_j C_j, \sum U_j, \sum T_j, \sum w_j T_j\}$

TAB. 7.1 – Ordonnancement avec travaux interférants sur une seule machine [118]

L'organisation de ce chapitre est la suivante. Dans le paragraphe 7.2, nous présentons des problèmes polynomiaux et la complexité des algorithmes de résolution de ces problèmes. Dans le paragraphe 7.3, certains problèmes $\mathcal{NP}$-difficiles au sens faible sont identifiés. Dans le paragraphe 7.4, nous abordons les problèmes $\mathcal{NP}$-difficiles au sens fort. La dernière section 7.4 présente des problèmes qui restent ouverts.

7.2 Problèmes résolus en temps polynomial

Proposition 7.2.1 *Les problèmes d'ordonnancement suivants sont polynomiaux :*

1. *problèmes* $1||F_\ell\big(C_{\max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$,
2. *problèmes* $1||F_\ell\big(C_{\max}(\mathcal{N}), L_{max}(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(C_{\max}(\mathcal{N})/L_{max}(\mathcal{N}_1)\big)$,
3. *problèmes* $1||F_\ell\big(C_{\max}(\mathcal{N}), \sum w_jC_j(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(C_{\max}(\mathcal{N})/\sum w_jC_j(\mathcal{N}_1)\big)$,
4. *problèmes* $1||F_\ell\big(C_{\max}(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(C_{\max}(\mathcal{N})/\sum U_j(\mathcal{N}_1)\big)$,
5. *problèmes* $1||F_\ell\big(\sum C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(\sum C_j(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$,
6. *problèmes* $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum C_j(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(L_{\max}(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$,
7. *problèmes* $1||F_\ell\big(\sum w_jC_j(\mathcal{N}), \sum w'_jC_j(\mathcal{N}_1)\big)$.

Preuve. Les problèmes 1 sont triviaux car il est suffisant d'ordonnancer les travaux de $\mathcal{N}_1$ d'abord dans un ordre arbitraire et les travaux restant ensuite arbitrairement. De la même façon, les problèmes 2 sont triviaux car il est suffisant d'ordonnancer d'abord les travaux de $\mathcal{N}_1$ dans l'ordre EDD et les problèmes 3 sont triviaux car il est suffisant d'ordonnancer d'abord les travaux de $\mathcal{N}_1$ dans l'ordre WSPT. Les problèmes 4 peuvent être résolus en appliquant l'algorithme de Moore [162] sur l'ensemble de $\mathcal{N}_1$. Les problèmes 5 et 6 peuvent être transformés en un problème d'ordonnancement $1|\tilde{d}_j|\sum C_j(\mathcal{N})$ qui est polynomial en $O(n\log n)$ [99]. Problème 7 est trivial (voir [18]). □.

Selon les résultats présentés dans [18], [227] et [38], nous pouvons déduire la complexité des problèmes d'ordonnancement suivants.

Proposition 7.2.2 *Les problèmes d'ordonnancement suivants peuvent être résolus en temps polynomial :*

- *problèmes* $1||F_\ell\big(L_{max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(L_{max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$
- *problèmes* $1||F_\ell\big(L_{\max}(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(L_{\max}(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$,
- *problème* $1||F_\ell\big(\sum w_jC_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$.

Preuve.

- Les problèmes $1||F_\ell\big(L_{max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ et $1||\varepsilon\big(L_{max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ sont polynomiaux.

  De la même façon, les travaux sont ordonnés selon EDD et toutes les séquences $EDD(\mathcal{N}_1 \cup \{J_{n_1+1}, \ldots, J_j\})//EDD(\{J_{j+1}, J_{j+2}, \ldots, J_n\})$ pour tout $j \in \{n_1+1, n_1+2, \ldots, n\}$ sont vérifiées. La meilleure séquence donne une solution optimale. □

– Le problème $1||F_\ell\big(L_{\max}(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ est polynomial.

Il existe une solution optimale telle que les travaux de $\mathcal{N}_1$ sont exécutés dans l'ordre EDD et les travaux de $\mathcal{N} \setminus \mathcal{N}_1$ sont exécutés dans l'ordre EDD. Sans perte de généralité, les travaux de $\mathcal{N}_1$ sont numérotés de 1 à n_1 dans l'ordre EDD et les travaux de $\mathcal{N} \setminus \mathcal{N}_1$ sont également numérotés de $n_1 + 1$ à n dans l'ordre EDD. Par conséquent, la date de fin d'exécution d'un travail J_i de $\mathcal{N}_1$ a n_2 valeurs possibles données par $\sum_{k=1}^{i} p_k + \sum_{k=n_1+1}^{l} p_k$ avec $l \in \{0, n_1+1, n_1+2, \ldots, n\}$.

Nous introduisons les notations suivantes : $P_{(i,j)} = \sum_{1\leq k\leq i} p_k + \sum_{n_1+1\leq k\leq j} p_k$. Considérons que L correspond à $L_{\max}(\mathcal{N}_1)$. Nous supposons que les travaux sont numérotés selon la règle EDD : $d_1 \leq d_2 \leq \ldots \leq d_{n_1}$ d'une part et $d_{n_1+1} \leq \ldots \leq d_n$ d'autre part.

La fonction objectif à minimiser est $F(i, j, L) = L_{\max}(\mathcal{N})$. $\mathcal{L}$ représente l'ensemble des valeurs possibles de $L_{max}(\mathcal{N}_1)$ ($|\mathcal{L}| \leq n_1 \times (n_2 + 1) \leq n^2$, comme présenté dans [227]). On définit $F(0, n_1, 0) = 0, F(i, j, L) = +\infty, \forall (i, j, L) \neq (0, n_1, 0)$. Considérons maintenant le triplet $F(i, j, L)$:

– si le travail J_{i+1} est inséré, le triplet $F(i+1, j, \max(L, P_{(i,j)})$ est mis à jour : $F(i+1, j, \max(L, P_{(i,j)})) \leftarrow \min\big(F(i+1, j, \max(L, P_{(i,j)})); \max(F(i, j, L); P_{(i,j)} - d_i)\big)$

– si le travail J_{j+1} est inséré, le triplet $F(i, j+1, \max(L, P_{(i,j)})$ est mis à jour :

$$F(i+1, j, L)) \leftarrow \min\big(F(i, j+1, L); \max(F(i, j+1, L); P_{(i,j)} - d_j)\big)$$

Pour le problème de minimisation $\alpha L_{\max}(\mathcal{N}) + \beta L_{\max}(\mathcal{N}_1))$, la solution optimale correspond à $\min_{L\in\mathcal{L}} \alpha F(n_1, n, L) + \beta L$.

Pour le problème de minimisation $L_{\max}(\mathcal{N})$ with $L_{\max}(\mathcal{N}_1) \leq \varepsilon$, la solution optimale correspond à $\min_{L\in\mathcal{L}, L\leq\varepsilon} F(n_1, n, L)$.

Le temps de calcul au total pour déterminer une solution optimale est borné par $O(n^4)$. □

– Le problème $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ est polynomial.

− Pour le cas particulier où $w_j = w$: une solution optimale existe toujours avec les travaux dans $\mathcal{N}_1$ et les travaux dans $\mathcal{N} \setminus \mathcal{N}_1$ séquencés dans l'ordre WSPT. De plus, pour le critère de minimisation la date de fin d'exécution maximum, seule la date de fin d'exécution du dernier travail de $\mathcal{N}_1$ doit être considérée. Les travaux exécutés avant ce dernier travail de $\mathcal{N}_1$ sont donc ordonnancés dans l'ordre SPT, qu'ils appartiennent à $\mathcal{N}_1$ ou pas.

Nous évaluons les séquence $SPT(\mathcal{N}_1 \cup \{J_{n_1+1}, \ldots, J_j\})//SPT(\{J_{j+1}, J_{j+2}, \ldots, J_n\})$

pour tout $j \in \{n_1+1, n_1+2, \ldots, n\}$ ($a//b$ définit la concaténation entre a et b). La meilleure séquence est la solution optimale du problème. Cet algorithme peut être implémenté en $O(n\log(n))$.

$-$ Pour le cas général où les poids w_j sont définis aléatoirement. Soit J_i le dernier travail de $\mathcal{N}_1$ exécuté dans une solution optimale S^*. Nous savons que dans une solution optimale S^*, tous les travaux exécutés à une position avant celle du travail J_i respectent l'ordre WSPT ; tous les travaux exécutés après le travail J_i dans S^* respectent l'ordre WSPT (la preuve peut être donnée par la règle d'échange de deux travaux adjacents comme présentée dans [195]). Selon J_i, le dernier travail de $\mathcal{N}_1$, un travail J_j de $\mathcal{N} \setminus \mathcal{N}_1$ exécuté après J_i doit satisfaire l'inégalité : $\frac{p_i}{\alpha w_i + \beta} \leq \frac{p_j}{\alpha w_j}$. Par conséquent, les travaux de $\mathcal{N} \setminus \mathcal{N}_1$ qui ne respectent pas cette inégalité doivent être ordonnancés avant J_i dans S^* et donc la séquence S^* est déterminée. L'algorithme de résolution se déroule comme suit :

1. déterminer la solution initiale S^0 où tous les travaux sont rangés dans l'ordre WSPT,
2. selon le dernier travail de $\mathcal{N}_1$:
   - faire une recherche dichotomique pour déterminer le premier travail J_j de $\mathcal{N} \setminus \mathcal{N}_1$ satisfaisant l'inégalité ci-dessus,
   - déplacer tous les travaux $\mathcal{N} \setminus \mathcal{N}_1$ exécuté à partir de J_j dans S^0 après l'exécution de J_i,
3. nouvelle séquence obtenue S^* correspond à la solution optimale.

Le temps de calcul au total est borné par $O(n\log n)$. □

7.3 Problèmes NP-difficiles au sens ordinaire

Proposition 7.3.1 *Les problèmes d'ordonnancement suivants sont $\mathcal{NP}$-difficiles au sens ordinaire :*

1. *problèmes* $1||\varepsilon\big(L_{\max}(\mathcal{N}) / \sum U_j(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(\sum U_j(\mathcal{N}) / L_{\max}(\mathcal{N}_1)\big)$,
2. *problème* $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / C_{\max}(\mathcal{N}_1)\big)$,
3. *problème* $1||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$,
4. *problèmes* $1||\varepsilon\big(\sum T_j(\mathcal{N}) / f(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(f(\mathcal{N}) / \sum T_j(\mathcal{N}_1)\big)$ *avec* $f \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_j C_j, \sum U_j, \sum T_j, \sum w_j T_j\}$.

Preuve.

1. Les problèmes $1||\varepsilon\big(L_{\max}(\mathcal{N})/\sum U_j(\mathcal{N}_1)\big)$ et $1||\varepsilon\big(\sum U_j(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$ sont $\mathcal{NP}$-difficiles au sens ordinaire.

   Comme le problème $1|\tilde{d}_j|\sum U_j$ est $\mathcal{NP}$-difficile [149], les problèmes considérés sont également $\mathcal{NP}$-difficiles. À partir d'une instance du problème $1|\tilde{d}_j|\sum U_j$, la preuve est donnée en insérant un nouveau travail qui définit le sous-ensemble de $\mathcal{N}\setminus\mathcal{N}_1$ avec durée opératoire et date de fin très grande telles qu'il existe une solution optimale où ce travail est ordonnancé à la dernière position. □

2. Problème $1||\varepsilon\big(\sum w_jC_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ est $\mathcal{NP}$-difficile au sens ordinaire.

   Soit EWCCM le problème de décision associée à $1||\varepsilon\big(\sum w_jC_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$. Ce problème est défini par :

   EWCCM

   Données : Un ensemble $\mathcal{N}$ de n travaux, un sous-ensemble $\mathcal{N}_1\subset\mathcal{N}$, durée opératoire p_j et un poids w_j pour chaque travail J_j, $1\leq j\leq n$, deux valeurs entières Y et Y_1.

   Question : Existe-t-il un ordonnancement σ pour $\mathcal{N}$ tel que

   $$\textstyle\sum_{J_j\in\mathcal{N}} w_jC_j\leq Y \text{ et } \max_{J_j\in\mathcal{N}_1} C_j\leq Y_1\,?$$

   Nous montrons que PARTITION $\propto$ EWCCM. Rappelons que le problème de PARTITION [71] est défini comme suit :

   PARTITION [71]

   Données : Un ensemble $\mathcal{A}$ de r éléments $a_1,a_2,\ldots,a_r$, avec des valeurs entières notées $s(a_i)$, $\forall i, 1\leq i\leq r$, $\sum_{i=1}^{r} s(a_i)=2B$.

   Question : Existe-t-il un sous-ensemble $\mathcal{A}_1$ des indices tel que

   $$\textstyle\sum_{i\in\mathcal{A}_1} s(a_i)=\sum_{i\in\{1,2,\ldots,r\}\setminus\mathcal{A}_1} s(a_i)=B\,?$$

   À partir de l'instance de PARTITION, on définit une instance du problème EWCCM comme suit :

   – $\mathcal{N}=\{1,2,\ldots,r+1\}, \mathcal{N}_1=\{r+1\}$,

   – pour le travail $J_j, j\in\{1,2,\ldots,r\}$: $p_j=w_j=s(a_j)$;

   – pour le travail J_{r+1} : $p_{r+1}=2B$, $w_{r+1}=1$,

   – $Y=6B^2+3B$ and $Y_1=3B$,

   (⇒) Supposons que la réponse au problème PARTITION est 'oui', on définit une solution pour EWCCM en ordonnançant un sous-ensemble des travaux corres-

pondant à $\mathcal{A}_1$ avant la travail $r+1$, les travaux qui correspondent à $\mathcal{A} \setminus \mathcal{A}_1$ sont ordonnancés après travail $r+1$. Cet ordonnancement est une solution réalisable et la réponse au problème de EWCCM es donc 'oui'.

($\Leftarrow$) S'il existe une solution réalisable pour le problème EWCCM, alors :

(a) $\max_{J_j \in \mathcal{N}_1} C_j \leq Y_1 \Leftrightarrow \sum_{J_j \in \mathcal{A}_1} p_j + p_{r+1} \leq Y_1 \Leftrightarrow \sum_{J_j \in \mathcal{A}_1} s(a_j) \leq B$

(b) $\sum_{J_j \in \mathcal{N}} w_j C_j \leq Y$

$\Leftrightarrow (\sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} p_j)(\sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} w_j) + w_{r+1}(\sum_{J_j \in \mathcal{A}_1} p_j + p_{r+1}) + p_{r+1}(\sum_{J_j \in \mathcal{N} \setminus (\mathcal{N}_1 \cup \mathcal{A}_1)} w_j) \leq Y$

$\Leftrightarrow 4B^2 + w_{r+1}(p_{r+1} + \sum_{J_j \in \mathcal{A}_1} p_j) + p_{r+1}(\sum_{J_j \in \mathcal{N} \setminus (\mathcal{N}_1 \cup \mathcal{A}_1)} w_j) \leq Y$

$\Leftrightarrow 4B^2 + w_{r+1}(p_{r+1} + \sum_{J_j \in \mathcal{A}_1} p_j) + p_{r+1}(\sum_{J_j \in \mathcal{N} \setminus (\mathcal{N}_1 \cup \mathcal{A}_1)} p_j) \leq Y$

$\Leftrightarrow 4B^2 + w_{r+1}(p_{r+1} + \sum_{J_j \in \mathcal{A}_1} p_j) + p_{r+1}(2B - \sum_{J_j \in \mathcal{A}_1)} p_j) \leq Y$

$\Leftrightarrow (w_{r+1} - p_{r+1}))(\sum_{J_j \in \mathcal{A}_1} s(a_j)) \leq Y - 4B^2 - p_{r+1}w_{r+1} - 2Bp_{r+1}$

$\Leftrightarrow (w_{r+1} - p_{r+1}))(\sum_{J_j \in \mathcal{A}_1} s(a_j)) \leq -2B^2 + B$

$\Leftrightarrow (p_{r+1} - w_{r+1}))(\sum_{J_j \in \mathcal{A}_1} s(a_j)) \geq B(2B-1)$

$\Leftrightarrow \sum_{J_j \in \mathcal{A}_1} s(a_j) \geq B$

Nous déduisons que $\sum_{j \in \mathcal{A}_1} s(a_j) = B$ et $\sum_{j \in \mathcal{A} \setminus \mathcal{A}_1} s(a_j) = B$ et donc la réponse à PARTITION est 'oui'. □

3. Problème $1||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$ est $\mathcal{N}P$-difficile au sens ordinaire.

Nous étudions d'abord les propriétés du problème $1||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$.

Propriété 7.3.2 *Il existe une solution optimale qui respecte les propriétés suivantes :*

(a) il n'y a pas de temps mort intermédiaire entre les travaux adjacents ni au début de l'ordonnancement ;

(b) l'ordonnancement des travaux du sous-ensemble N_1 respecte l'ordre SPT : $p_i \leq p_j \Leftrightarrow i <_{prec} j; Ji, Jj \in N_1$;

(c) l'ordonnancement des travaux du sous-ensemble $(N - N_1)$ respecte l'ordre SPT : $p_i \leq p_j \Leftrightarrow i <_{prec} j; Ji, Jj \notin N_1$;

(d) si $p_i \leq p_j$ alors travail J_i doit être ordonnancé avant J_j, $\forall (i,j) \in \mathcal{N}_1 \times (\mathcal{N} \setminus \mathcal{N}_1)$.

Le premier point est vrai car nous considérons des critères réguliers. Les deux points suivants sont vrais parce qu'un échange des travaux qui ne respectent pas l'ordre SPT ne peut pas diminuer la qualité de la solution. Le dernier point est

vrai parce que la permutation de J_i et de J_j améliore tant $\sum_{J_j \in \mathcal{N}} C_j$ que $\sum_{J_j \in \mathcal{N}_1} C_j$. Notez que le point 4 n'est pas vrai si $(i,j) \in (\mathcal{N} \setminus \mathcal{N}_1) \times \mathcal{N}_1$.

Afin de montrer la complexité du problème $1||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$, nous allons faire un réduction au problème de partition avec éléments distincts :

PAED

Données : Un ensemble $\mathcal{B}$ de t éléments $b_1, b_2, \ldots, b_t$, d'entiers notés ($s(b_i) \neq s(b_j), \forall i,j$), $\sum_{i=1}^{t} s(b_i) = 2C$.

Question : Existe-t-il un sous-ensemble $\mathcal{B}_1$ des indices tel que

$$\sum_{i \in \mathcal{B}_1} s(b_i) = \sum_{i \in \{1,2,\ldots,t\} \setminus \mathcal{B}_1} s(b_i) = C \,?$$

Le problème PAED est $\mathcal{N}P$-difficile ([120]). Soit ESCSC le problème de décision associé au problème $1||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$. Ce problème est défini par :

ESCSC

Données : Un ensemble $\mathcal{N}$ de n travaux, un sous-ensemble $\mathcal{N}_1 \subset \mathcal{N}$, durée opératoire p_j pour chaque travail J_j, $1 \leq j \leq n$, deux valeurs entières Y et Y_1.

Question : Existe-t-il un ordonnancement σ pour $\mathcal{N}$ tel que

$$\sum_{J_j \in \mathcal{N}} C_j \leq Y \text{ et } \sum_{J_j \in \mathcal{N}_1} C_j \leq Y_1 \,?$$

Nous montrons que PAED $\propto$ ESCSC.

Considérons une instance de PAED et supposons sans perte de généralité que $s(a_1) < s(a_2) < \ldots < s(a_t)$. Nous avons $\min_{i=1}^{t-1} \frac{a_{i+1}}{a_i} > 1$. Il est toujours possible de trouver α et K tels que $1 < \alpha < \min_{i=1}^{t-1} \frac{a_{i+1}}{a_i}$ et $\alpha K \in N$ (si $\frac{a_{\ell+1}}{a_\ell} = \min_{i=1}^{t-1} \frac{a_{i+1}}{a_i}$; prenons par exemple $\alpha = \frac{a_{\ell+1}}{a_\ell+1}$ et $K = a_\ell + 1$ si $a_{\ell+1} \neq a_\ell + 1$ ou prenons par exemple $\alpha = \frac{10 \times a_{\ell+1}}{10 \times a_\ell + 1}$ et $K = 10 \times a_\ell + 1$ sinon).

Suite à la définition de α et de K, nous avons : $Ks(a_i) < \alpha Ks(a_i) < Ks(a_{i+1}) < \alpha Ks(a_{i+1})$.

Soient $\beta = \alpha - 1$, $\beta > 0$ et $X = K \sum_{i=1}^{t} \big(2(t-i+1) + (2t-2i+1)\alpha\big) \times s(a_i)$.

Nous construisons une instance du problème ESCSC comme suit : $n = 2t$ et

- $p_{(2i-1)} = K \times s(a_i), \forall i = 1,2,\ldots,t,$
- $p_{(2i)} = \alpha K \times s(a_i), \forall i = 1,2,\ldots,t,$
- $Y_1 = K(1+\alpha)\big(\sum_{i=1}^{t}(t-i+1) \times s(a_i)\big) - KC,$
- $Y = X + \beta KC,$
- $\mathcal{N}_1 = \{2,4,6,\ldots,2t\}.$

Soit S^0 une solution initiale définie comme suit : $S^0 = \{1, 2, 3, \ldots, 2t-1, 2t\}$, c-à-d. la séquence où les travaux sont ordonnancés selon la règle SPT (voir Figure 7.1).

Nous avons :

$\sum_{j=1}^{n} C_j(S^0) = Ks(a_1) + (Ks(a_1) + \alpha Ks(a_1)) + (Ks(a_1) + \alpha Ks(a_1) + Ks(a_2)) + (Ks(a_1) + \alpha Ks(a_1) + Ks(a_2) + \alpha Ks(a_2)) + \ldots$

$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = 2tKs(a_1) + (2t-1)\alpha Ks(a_1) + (2t-2)Ks(a_2) + (2t-3)\alpha Ks(a_2) + \ldots$

$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = Ks(a_1)(2t + (2t-1)\alpha) + Ks(a_2)((2t-2) + (2t-3)\alpha) + \ldots$

$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = K\sum_{i=1}^{t} \big(2(t-i+1) + (2t-2i+1)\alpha\big) \times s(a_i) = X.$

De la même façon, nous obtenons :

$\sum_{J_j \in \mathcal{N}_1} C_j(S^0) = (K \times s(a_1) + \alpha K \times s(a_1)) + (K \times s(a_1) + \alpha K \times s(a_1) + K \times s(a_2) + \alpha K \times s(a_2)) + \ldots$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^0) = (K \times s(a_1) + \alpha K \times s(a_1)) + (K \times s(a_1) + \alpha K \times s(a_1) + K \times s(a_2) + \alpha K \times s(a_2)) + \ldots$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^0) = t \times (K \times s(a_1) + \alpha K \times s(a_1)) + (t-1) \times (K \times s(a_2) + \alpha K \times s(a_2)) + \ldots$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^0) = t \times ((1+\alpha)K \times s(a_1)) + (t-1) \times ((1+\alpha)K \times s(a_2)) + \ldots$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^0) = K(1+\alpha)(t \times s(a_1) + (t-1) \times s(a_2) + \ldots$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^0) = K(1+\alpha)\sum_{i=1}^{t}(t-i+1) \times s(a_i) = Y_1 + KC$

Par conséquent, cette solution n'est pas faisable pour le problème ESCSC car $\sum_{J_j \in \mathcal{N}} C_j(S^0) \leq Y$ mais $\sum_{J_j \in \mathcal{N}_1} C_j(S^0) > Y_1$.

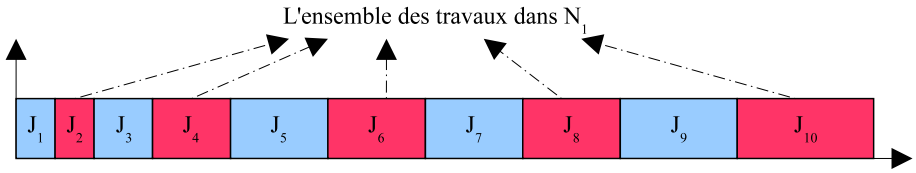

FIG. 7.1 – Une séquence initiale avec 10 travaux

Supposons qu'on sache résoudre le PAED. Nous allons illustrer une méthode basée sur la permutation de certains travaux adjacents dans S^0 afin de diminuer $\sum_{J_j \in \mathcal{N}_1} C_j$ ce qui augmentera de façon raisonnable $\sum_{J_j \in \mathcal{N}} C_j$ et qui donnera la séquence optimale.

Considérons l'ensemble des travaux $\mathcal{G} = \{J_j \in \mathcal{N} / j = 2i \wedge i \in \mathcal{B}_1\}$. Notons que $\mathcal{G} \subseteq \mathcal{N}_1$. Nous construisons la séquence S^1 à partir de S^0 par la permutation d'un

travail d'indice dans $\mathcal{G}$ avec son prédécesseur : $S^1[j] = S^0[j-1]$, $S^1[j-1] = S^0[j]$ pour $J_j \in \mathcal{G}$ et $S^1[j] = S^0[j]$ pour les autres travaux.

Ainsi, il faut calculer les deux valeurs suivantes $\sum_{J_j \in \mathcal{N}} C_j(S^1)$ et $\sum_{J_j \in \mathcal{N}_1} C_j(S^1)$. Pour cela, nous analysons d'abord la séquence S' construite à partir de S^0 par la permutation de deux travaux adjacents (l'indice du successeur est dans $\mathcal{G}$).

$\sum_{J_j \in \mathcal{N}} C_j(S') = \sum_{J_j \in \mathcal{N}} C_j(S^0) + (p_j - p_{j-1})$.

Donc, $\sum_{J_j \in \mathcal{N}} C_j(S^1) = \sum_{J_j \in \mathcal{N}} C_j(S^0) + \sum_{J_j \in \mathcal{G}} (p_j - p_{j-1}) = \sum_{J_j \in \mathcal{N}} C_j(S^0) + \sum_{J_j \in \mathcal{G}} (\alpha K \times s(a_{j/2}) - K \times s(a_{j/2}))$.

$\Rightarrow \sum_{J_j \in \mathcal{N}} C_j(S^1) = \sum_{J_j \in \mathcal{N}} C_j(S^0) + \sum_{J_j \in \mathcal{G}} (\beta K \times s(a_{j/2})) = \sum_{J_j \in \mathcal{N}} C_j(S^0) + \beta K \times \sum_{J_j \in \mathcal{G}} (s(a_{j/2}))$

$\Rightarrow \sum_{J_j \in \mathcal{N}} C_j(S^1) = \sum_{J_j \in \mathcal{N}} C_j(S^0) + \beta K \times C = X + \beta K C = Y$

De même $\sum_{J_j \in \mathcal{N}_1} C_j(S') = \sum_{J_j \in \mathcal{N}_1} C_j(S^0) - p_{j-1}$

Par conséquent, $\sum_{J_j \in \mathcal{N}_1} C_j(S^1) = \sum_{J_j \in \mathcal{N}_1} C_j(S^0) - \sum_{J_j \in \mathcal{G}} p_{j-1}$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^1) = \sum_{J_j \in \mathcal{N}_1} C_j(S^0) - \sum_{J_j \in \mathcal{G}} K \times s(a_{j/2})$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(S^1) = \sum_{J_j \in \mathcal{N}_1} C_j(S^0) - KC = Y_1 + KC - KC = Y_1$

Donc, S^1 est une solution réalisable et optimale pour le problème ESCSC.

Supposons maintenant qu'on sache résoudre ESCSC. Soit σ la séquence. Si σ ne respecte pas les conditions de la proposition 7.3.2, alors nous déplaçons tous les travaux vers la gauche, appliquons par la suite la règle SPT pour les travaux de $\mathcal{N}_1$, appliquons l'ordre SPT pour les travaux de $\mathcal{N} \setminus \mathcal{N}_1$. Par la suite, si deux travaux J_i et J_j ne respectent pas de dernier point de la proposition 7.3.2, ils doivent être permutés. Nous obtenons une nouvelle séquence σ' telle que :

- $\sum_{J_j \in \mathcal{N}} C_j(\sigma') \leq \sum_{J_j \in \mathcal{N}} C_j(\sigma) \leq Y$ **(1)**
- $\sum_{J_j \in \mathcal{N}_1} C_j(\sigma') \leq \sum_{J_j \in \mathcal{N}_1} C_j(\sigma) \leq Y_1$ **(2)**
- et σ' satisfait les conditions de la proposition 7.3.2.

Nous allons maintenant comparer σ' et S^0.

Considérons le travail numéro $2i$. Ce travail est ordonnancé à la position $2i$ dans S^0 et à la position k dans σ'. Supposons que $k > 2i$. Dans ce cas, il existe au moins un travail ordonnancé avant la position $2i$ dans σ' avec une durée opératoire supérieure à p_{2i}. Ce travail ne peut pas appartenir à $\mathcal{N}_1$ car les travaux de $\mathcal{N}_1$ dans σ' sont ordonnancés selon l'ordre SPT. Ce travail appartient donc à $\mathcal{N} \setminus \mathcal{N}_1$. Ce cas

n'est pas possible à cause du dernier point de la Proposition 7.3.2. Par conséquent, $k \leq 2i$. De même, nous pouvons montrer que le travail $2i-1$ est à la position $2i-1$ dans S^0 et à la position l dans σ' avec $l \geq 2i-1$. Ce cas est illustré par la Figure 7.2.

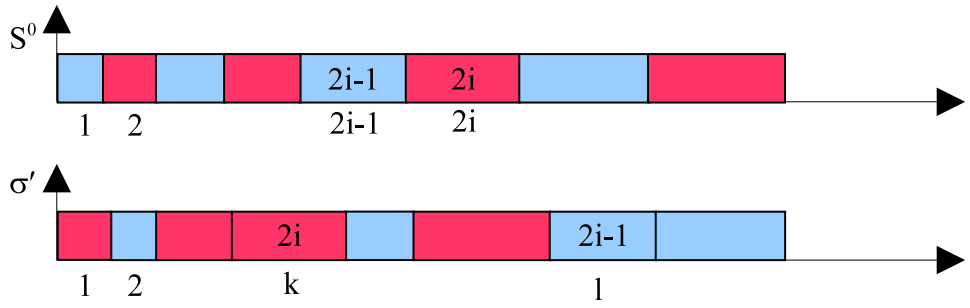

FIG. 7.2 – Séquences S^0 et σ' et position des travaux $2i-1$ et $2i$

Soit l'ensemble des travaux $\mathcal{H}_{2i} = \{J_j / (j \succ_{\sigma'} 2i) \wedge (p_j < p_{2i}) \wedge (j \in \mathcal{N} \setminus \mathcal{N}_1)\}$. Par exemple selon la figure précédente, le travail $2i-1$ appartient à $\mathcal{H}_{2i}$. Ces travaux sont dans $\mathcal{N} \setminus \mathcal{N}_1$ et précèdent le travail $2i$ dans S^0.

Nous avons $C_{2i}(S^0) = C_{2i}(\sigma') + \sum_{J_k \in \mathcal{H}_{2i}} p_k$ selon la définition de $\mathcal{H}_{2i}$.

$\Rightarrow C_{2i}(\sigma') = C_{2i}(S^0) - \sum_{J_k \in \mathcal{H}_{2i}} p_k$

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(\sigma') = \sum_{J_j \in \mathcal{N}_1} C_j(S^0) - \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} p_k$ **(3)**

$\Rightarrow \sum_{J_j \in \mathcal{N}_1} C_j(\sigma') = Y_1 + KC - \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} p_k$

Selon l'inégalité (2) $\sum_{J_j \in \mathcal{N}_1} C_j(\sigma') \leq Y_1$, nous avons :

$Y_1 + KC - \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} p_k \leq Y_1$

$\Rightarrow KC \leq \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} p_k$

$\Rightarrow KC \leq \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} Ks(a_{(k+1)/2})$

$\Rightarrow C \leq \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} s(a_{(k+1)/2})$ **(4)**

Par rapport à la solution initiale S^0 et selon le dernier point dans la proposition 7.3.2, la position du travail $J_j \in \mathcal{N}_1$ dans σ' sera inchangée ou déplacée vers la gauche. De même, la position du travail $J_j \in \mathcal{N} \setminus \mathcal{N}_1$ dans σ' sera inchangée ou déplacée vers la droite. La différence de date de fin d'exécution d'un travail $J_j \in \mathcal{N} \setminus \mathcal{N}_1$ entre deux séquences σ' et S^0 est déterminée par la somme des durées opératoires des travaux de $\mathcal{N}_1$ ordonnancés après J_j dans S^0 et ordonnancés avant J_j dans σ'. Par exemple, selon la Figure 7.2, la déviation de la date de fin

d'exécution du travail $2i-1$ entre deux séquences σ' et S^0 est au moins égale à p_{2i}. Plus généralement, nous avons :

$C_{2i-1}(\sigma') = C_{2i-1}(S^0) + \sum_{J_k \in \mathcal{N}_1 | 2i-1 \in \mathcal{H}_k} p_k$

$\Rightarrow \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} C_j(\sigma') - \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} C_j(S^0) = \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} \sum_{J_k \in \mathcal{N}_1 | J_j \in \mathcal{H}_k} p_k$

$\Rightarrow \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} C_j(\sigma') - \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} C_j(S^0) = \sum_{J_k \in \mathcal{N}_1 | J_j \in \mathcal{H}_k} \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} p_k = \sum_{J_k \in \mathcal{N}_1} \sum_{J_j \in \mathcal{H}_k} p_k$

Donc, la déviation de la somme des dates de fin d'exécution entre deux séquences σ' et S^0 est définie comme suit.

$\sum_j C_j(\sigma') - \sum_j C_j(S^0)$

$= \left(\sum_{J_j \in \mathcal{N}_1} C_j(\sigma') - \sum_{J_j \in \mathcal{N}_1} C_j(S^0) \right) + \left(\sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} C_j(\sigma') - \sum_{J_j \in \mathcal{N} \setminus \mathcal{N}_1} C_j(S^0) \right).$

Selon (3), nous avons : $\sum_{J_j \in \mathcal{N}_1} C_j(\sigma') - \sum_{J_j \in \mathcal{N}_1} C_j(S^0) = \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} p_k$

$\Rightarrow \sum_j C_j(\sigma') - \sum_j C_j(S^0) = \sum_{J_k \in \mathcal{N}_1} \sum_{J_j \in \mathcal{H}_k} p_k - \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} p_k.$

$\Rightarrow \sum_j C_j(\sigma') - \sum_j C_j(S^0) = \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} (p_j - p_k)$

Depuis $p_j > p_k$ où $J_j \in \mathcal{N}_1, J_k \in \mathcal{H}_j$, nous avons $p_j \geq p_{k+1}$ avec $J_j, J_{k+1} \in \mathcal{N}_1$ et $J_k \in \mathcal{H}_j$ **(5)**

$\Rightarrow \sum_{J_j \in \mathcal{N}} C_j(\sigma') - \sum_{J_j \in \mathcal{N}} C_j(S^0) \geq \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} (p_{k+1} - p_k) = \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} \left(\alpha K s(a_{(k+1)/2}) - K s(a_{(k+1)/2}) \right)$

$\Rightarrow \sum_{jJ_\in \mathcal{N}} C_j(\sigma') - \sum_{J_j \in \mathcal{N}} C_j(S^0) \geq \beta K \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} s(a_{(k+1)/2})$

$\Rightarrow \sum_{J_j \in \mathcal{N}} C_j(\sigma') \geq \sum_{J_j \in \mathcal{N}} C_j(S^0) + \beta K \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} s(a_{(k+1)/2})$

Selon l'inégalité (2) que $\sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} s(a_{(k+1)/2}) \geq C$, nous avons alors :

$\sum_{J_j \in \mathcal{N}} C_j(\sigma') \geq \sum_{J_j \in \mathcal{N}} C_j(S^0) + \beta K C = Y$ **(6)**

Par conséquent, grâce aux (1) et (6), nous déduisons que $\sum_{J_j \in \mathcal{N}} C_j(\sigma') = Y$.

Autrement dit, toutes inégalités (4) et (5) doivent être devenues égalités :

$$\begin{cases} \sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} s(a_{(k+1)/2}) = C \\ p_j = p_{k+1} \text{ où } J_j \in \mathcal{N}_1, J_k \in \mathcal{H}_j \end{cases}$$

Rappelons que les durées opératoires des travaux sont toutes différentes. Ainsi, soit $p_j = p_{k+1}$ (c.a.d. $|\mathcal{H}_j| = 1$) ou $|\mathcal{H}_j| = 0$ où $J_j \in \mathcal{N}_1, J_k \in \mathcal{H}_j$.

$\Rightarrow |\mathcal{H}_j| \leq 1, \forall J_j \in \mathcal{N}_1$

$\Rightarrow$ L'égalité $\sum_{J_j \in \mathcal{N}_1} \sum_{J_k \in \mathcal{H}_j} s(a_{(k+1)/2}) = C$ définit le sous-ensemble B_1 de PAED.

Par conséquent, les travaux J_j avec $|\mathcal{H}_j| = 1$ définissent la solution B_1 du problème PAED. □

4. Les problèmes $1||\varepsilon\big(\sum T_j(\mathcal{N})/f(\mathcal{N}_1)\big)$ et $1||\varepsilon\big(f(\mathcal{N})/\sum T_j(\mathcal{N}_1)\big)$ avec $f \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_jC_j, \sum U_j, \sum T_j, \sum w_jT_j\}$ sur $\mathcal{N}$ sont $\mathcal{NP}$-difficiles au sens ordinaire. Comme le problème $1||\sum T_j$ est $\mathcal{NP}$-difficile [60], les problèmes considérés sont également $\mathcal{NP}$-difficiles. □.

Les programmes dynamiques pour ces problèmes d'ordonnancement $\mathcal{NP}$-difficiles au sens ordinaire sont présentés dans le chapitre suivant.

7.4 Problèmes NP-difficiles au sens fort

Proposition 7.4.1 *Les problèmes suivants sont $\mathcal{NP}$-difficiles au sens fort :*

1. *problème* $1||\varepsilon\big(\sum w_jC_j(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$,
2. *problème* $1||\varepsilon\big(L_{\max}(\mathcal{N})/\sum w_jC_j(\mathcal{N}_1)\big)$,
3. *problème* $1||\varepsilon\big(\sum w_jC_j(\mathcal{N})/\sum U_j(\mathcal{N}_1)\big)$,
4. *problème* $1||\varepsilon\big(\sum U_j(\mathcal{N})/\sum w_jC_j(\mathcal{N}_1)\big)$,
5. *problèmes* $1||\varepsilon\big(\sum w_jT_j(\mathcal{N})/f(\mathcal{N}_1)\big)$ *et* $1||\varepsilon\big(f(\mathcal{N})/\sum w_jT_j(\mathcal{N}_1)\big)$ *avec* $f \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_jC_j, \sum U_j, \sum T_j, \sum w_jT_j\}$ *sur* $\mathcal{N}$.

Preuve. Le résultats peut être obtenus en utilisant l'instance définie par Lawler [146] pour le problème $1||\sum w_jT_j$.

Proposition 7.4.2 *Les problèmes suivants sont $\mathcal{NP}$-difficiles au sens fort :*

1. *Problème* $1||F_\ell\big(\sum w_jC_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$
2. *Problème* $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum w_jC_j(\mathcal{N}_1)\big)$
3. *Problème* $1||F_\ell\big(\sum w_jC_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$
4. *Problème* $1||F_\ell\big(\sum U_j(\mathcal{N}), \sum w_jC_j(\mathcal{N}_1)\big)$
5. *Problème* $1||F_\ell\big(\sum w_jC_j(\mathcal{N}), \sum w_jC_j(\mathcal{N}_1)\big)$

Preuve.

1. Problème $1||F_\ell\big(\sum w_jC_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ est $\mathcal{NP}$-difficile au sens fort.

   Le problème de décision associé est le suivant, noté FWCLM :

   FWCLM

   Données : un ensemble $\mathcal{N}$ de n travaux ; un sous-ensemble $\mathcal{N}_1 \subset \mathcal{N}$, durée d'exécution p_j pour chaque travail J_j, $1 \leq j \leq n$ et date de fin souhaitée d_j si $J_j \in \mathcal{N}_1$; a,

b et y réels.

Question : Existe-t-il un ordonnancement σ tel que

$$a\sum w_jC_j(\mathcal{N}) + bL_{\max}(\mathcal{N}_1) \leq y\,?$$

Il est clair que le problème FWCLM est dans $\mathcal{NP}$. Nous montrons par la suite que FWCLM est $\mathcal{NP}$-complet au sens fort par la réduction de 3-PARTITION, qui est connu comme $\mathcal{NP}$-complet au sens fort ([71]).

3-PARTITION

Données : un entier B et un ensemble $A = \{a_1, \ldots, a_{3r}\}$ $3r$ entiers positifs avec $B/4 < a_k < B/2$ $(k = 1, \ldots, 3r)$ et $\sum_{k=1}^{3r} a_k = rB$.

Question : Existe-t-il une partition de A en r sous-ensembles disjoints mutuellement $A_1, \ldots, A_r$ tels que la somme des éléments dans A_k soit égal à B, $\forall k = 1, \ldots, r$?

Étant donnée une instance de 3-PARTITION, nous construisons une instance de FWCLM comme suit :

- $\mathcal{N} = \{1, 2, \ldots, 4r\}$,
- $\mathcal{N}_1 = \{1, 2, \ldots, r\}$,
- pour J_j, $j \in \{1, 2, \ldots, r\}$: $p_j = w_j = 1$ et $d_j = 1 + (j-1)(B+1)$, soit $D = \sum_{i=1}^{r} d_j$,
- pour J_j, $j \in \{r+1, r+2, \ldots, 4r\}$: $p_j = a_{j-r}$, $w_j = D^2 a_{j-r}$ (tous les travaux ont le même ratio $\frac{p_j}{w_j} = 1/D^2$),
- $a = 1$ et $b = y = D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + D^2 Br(r+1)/2 + D$,

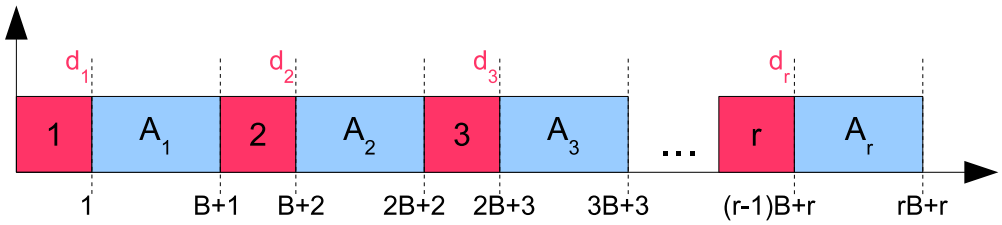

FIG. 7.3 – Une séquence optimale σ du $1||F_\ell\left(\sum w_jC_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\right)$

$(\Rightarrow)$ Supposons que 3-PARTITION a une réponse 'oui'. Alors, nous construisons une solution du problème FWCLM. Nous formons r blocks, où le jth block contient le travail J_j suivi des travaux correspondant aux éléments de A_j, que nous traitons dans cet ordre. Donc, $bL_{\max}(\mathcal{N}_1) = 0$. Il est facile de vérifier que la pondération de la date de fin d'exécution des travaux de $\mathcal{N}$ est égale à $D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + Br(r+1)/2 + D$. Ainsi, FWCLM a aussi une réponse 'oui'.

($\Leftarrow$) Supposons maintenant que FWCLM a une réponse 'oui', et soit σ une solution réalisable. Parce que le retard du travail J_1 est supérieur ou égal à 0, $L_{\max}(\mathcal{N}_1) \geq 0$. Parce que $\sum w_j C_j(\mathcal{N}) > 0$ et que $y = b$, nous avons $L_{\max}(\mathcal{N}_1) < 1$ et donc $L_{\max}(\mathcal{N}_1) = 0$ et le travail J_1 commence à l'instant 0. La somme pondérée des dates de fin d'exécution des travaux de $\mathcal{N}_1$ est donc inférieure ou égale à D.

Soit H_j ($j = 1, \ldots, r-1$) l'ensemble des travaux de $\mathcal{N} \setminus \mathcal{N}_1$ qui sont exécutés entre les travaux J_j et J_{j+1}, et soit H_r l'ensemble des travaux de $\mathcal{N}$ exécutés après le travail r dans la séquence σ. Nous utilisons $p(H_j)$ et $w(H_j)$ pour désigner la somme des durées opératoires et la somme des poids des travaux appartenant à H_j respectivement. La somme pondérée des dates de fin d'exécution des travaux dans $\mathcal{N} \setminus \mathcal{N}_1$ selon σ est donc égale à $D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} j w(H_j)$ (la première partie est le total des dates de fin d'exécution pondérées de $\mathcal{N} \setminus \mathcal{N}_1$ dans les travaux de $\mathcal{N}_1$). Car le travail J_2 est achevé avant sa date de fin souhaitée, nous savons que $p(H_1) \leq B$. De même, car le travail J_3 est achevé avant sa date de fin souhaitée, nous savons que $p(H_1) + p(H_2) \leq 2B$. En étendant ce raisonnement, nous constatons que

$rp(H_1) + (r-1)p(H_2) + \ldots + p(H_r) \leq \sum_{i=1}^{r} iB = Br(r+1)/2$

$\Leftrightarrow (r+1)[p(H_1) + p(H_2) + \ldots + p(H_r)] - \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2$

$\Leftrightarrow (r+1)[rB] - \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2$

$\Leftrightarrow Br(r+1) - Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j)$

$\Leftrightarrow Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j)$

D'autre part, nous avons

$\sum w_j C_j(\mathcal{N} \setminus \mathcal{N}_1) + \sum w_j C_j(\mathcal{N}_1) \leq D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + D^2 Br(r+1)/2 + D$

$\Leftrightarrow D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} j w(H_j) + \sum w_j C_j(\mathcal{N}_1) \leq D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + D^2 Br(r+1)/2 + D$

$\Leftrightarrow \sum_{j=1}^{r} j w(H_j) + \sum w_j C_j(\mathcal{N}_1) \leq D^2 Br(r+1)/2 + D$

$\Leftrightarrow D^2 \sum_{j=1}^{r} jp(H_j) + \sum w_j C_j(\mathcal{N}_1) \leq D^2 Br(r+1)/2 + D$

$\Leftrightarrow \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2 + (D - \sum w_j C_j(\mathcal{N}_1))/D^2$

Ainsi, nous avons :

$$Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2 + (D - \sum w_j C_j(\mathcal{N}_1))/D^2$$

Car $0 < \sum w_j C_j(\mathcal{N}_1) \leq D$, $0 < (D - \sum w_j C_j(\mathcal{N}_1))/D^2 < 1$.

Par conséquent, $\sum_{j=1}^{r} jp(H_j) = Br(r+1)/2$

Puisque $\forall j \in \{1,\ldots,r\}$, $p(H_1)+\ldots+p(H_j) \leq jB$ nous pouvons déduire que $\forall j \in \{1,\ldots,r\}$, $p(H_j) = B$. Par conséquent, la partition de A en $H_1,\ldots,H_r$ donne une instance de 3-PARTITION à réponse 'oui'. □

2. Problème $1||F_\ell\left(L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\right)$ est $\mathcal{NP}$-difficile au sens fort.

Le problème de décision associé est le problème suivant, noté FLMWC :

FLMWC

Données : Un ensemble $\mathcal{N}$ de n travaux ; un sous-ensemble $\mathcal{N}_1 \subset \mathcal{N}$; durée opératoire p_j pour chaque travail J_j, $1 \leq j \leq n$ et date de fin souhaitée d_j si $J_j \in \mathcal{N}_1$; a, b et y sont les valeurs réelles.

Question : Existe-t-il un ordonnancement σ tel que

$$aL_{\max}(\mathcal{N}) + b\sum w_j C_j(\mathcal{N}_1) \leq y\,?$$

Il est clair que le problème FLMWC est dans $\mathcal{NP}$. Nous montrons par la suite que FLMWC est $\mathcal{NP}$-complet au sens fort par la réduction de 3-PARTITION, connu comme $\mathcal{NP}$-complet au sens fort ([71]).

Étant donnée une instance de 3-PARTITION, nous construisons une instance du problème FLMWC comme suit :

- $\mathcal{N} = \{1,2,\ldots,4r+1\}$, $\mathcal{N}_1 = \{1,2,\ldots,3r\}$, $P_n = \sum_{j=1}^{4r+1} p_j$
- pour J_j, $j \in \{1,2,\ldots,3r\}$: $p_j = w_j = a_j$ et $d_j = P_n$,
- pour J_j, $j \in \{3r+1, 3r+2,\ldots,4r+1\}$: $p_j = B$ et $d_j = (2(j-3r)-1)B$,
- $y = \frac{1}{2}B^2 r(r+1) + \sum_{i=1}^{3n}\sum_{j=1}^{i} a_i a_j$,
- $a = 2y$ and $b = 1$.

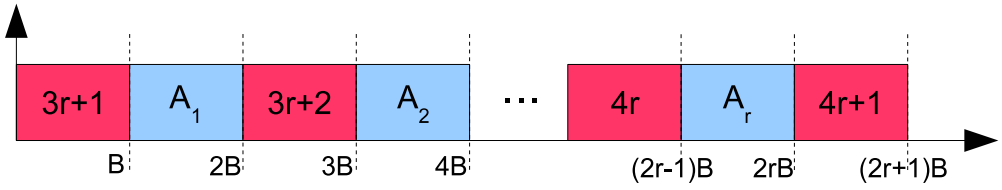

FIG. 7.4 – Une séquence optimale σ du $1||F_\ell\left(L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\right)$

Comme la définition de $a = 2y$, $L_{\max}(\mathcal{N})$ doit être inférieur ou égal à 0. Puisque le travail J_{3r+1} a une durée et une date de fin souhaitée égales à B, ce travail doit commencer à la date zéro. Par conséquent, $L_{\max}(\mathcal{N}) = 0$ et les travaux J_j, $j \in \{3r+1, 3r+2,\ldots,4r+1\}$ ne peuvent pas être en retard.

($\Rightarrow$) Supposons que le problème 3-PARTITION a une réponse 'oui' et qu'il partitionne l'ensemble A en r sous-ensembles disjoints A_j ($1 \leq j \leq r$). Nous construisons la solution suivante pour le problème FLMWC. Nous formons r blocs, où le bloc j contient le travail J_{3r+j} suivi des travaux correspondants aux éléments de A_j, que nous traitonsdans cet ordre. Le dernier travail de la séquence est le travail J_{4r+1}, qui se termine à la date P_n.

Il est facile de vérifier que le maximum du retard est égal à zéro et la somme des dates de fin d'exécution pondérées des travaux de $\mathcal{N}_1$ est égale à $\sum_{k=0}^{r-1} B^2(r-k) + \sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j = y$. Ainsi, $aL_{\max}(\mathcal{N}) + b\sum w_j C_j(\mathcal{N}_1) = y$ qui est signifié que LMWC a aussi un réponse 'oui'.

($\Leftarrow$) Supposons maintenant que FLMWC a une réponse 'oui', soit σ une solution réalisable. Le maximum retard de séquence σ est 0. Chaque travail $J_j \in \mathcal{N} \setminus \mathcal{N}_1$ est en avance, nous allons montrer que tous les travaux se terminent à (ou avant) leur date de fin souhaitée.

Soit H_j ($j = 1, \ldots, r-1$) l'ensemble des travaux de $\mathcal{N}$ qui sont exécutés entre les travaux $3r+j$ et $3r+j+1$. On utilise $p(H_j)$ et $w(H_j)$ pour indiquer la somme des durées opératoires et la somme des poids des travaux appartenant à H_j, respectivement. La somme des dates de fin d'exécution pondérées des travaux dans $\mathcal{N}_1$ selon σ, $\sum w_j C_j(\mathcal{N}_1)$, est donc égale à $\sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + B\sum_{j=1}^{r} jw(H_j)$.

Comme le travail J_{3r+2} doit être terminé avant sa date de fin souhaitée, $p(H_1) \leq B$. De la même façon, comme le travail J_{3r+3} doit être complété avant sa date de fin souhaitée, nous avons $p(H_1) + p(H_2) \leq 2B$. En étendant le même raisonnement, nous constatons que :

$rp(H_1) + (r-1)p(H_2) + \ldots + p(H_r) \leq \sum_{i=1}^{r} iB = Br(r+1)/2$

$\Leftrightarrow (r+1)[p(H_1) + p(H_2) + \ldots + p(H_r)] - \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2$

$\Leftrightarrow (r+1)[rB] - \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2$

$\Leftrightarrow Br(r+1) - Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j)$

$\Leftrightarrow Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j)$

$\Leftrightarrow Br(r+1)/2 \leq \sum_{j=1}^{r} jw(H_j)$ (car $w(H_j) = p(H_j)$).

D'autre part, nous avons :

$aL_{\max}(\mathcal{N}) + b\sum w_j C_j(\mathcal{N}_1) \leq y$

$\Leftrightarrow \sum w_j C_j(\mathcal{N}_1) \leq \frac{1}{2}B^2 r(r+1) + \sum_{i=1}^{3n}\sum_{j=1}^{i} a_i a_j$

$\Leftrightarrow \sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + B\sum_{j=2}^{r} jw(H_j) \leq \sum_{i=1}^{3n}\sum_{j=1}^{i} a_i a_j + \frac{1}{2}B^2 r(r+1)$

$\Leftrightarrow \sum_{j=1}^{r} jw(H_j) \leq Br(r+1)/2.$

Par conséquent, on déduit que : $\sum_{j=1}^{r} jp(H_j) = Br(r+1)/2$. Comme $\forall j \in \{1,\ldots,r\}$, $p(H_1)+\ldots+p(H_j) \leq jB$, nous pouvons déduire que $\forall j \in \{1,\ldots,r\}$, $p(H_j) = B$. Par conséquent, la partition de A en $H_1,\ldots,H_r$ donne une solution à 3-PARTITION à réponse 'oui'. □

Remarque 7.4.1 *Par conséquent, en utilisant le même raisonnement, le problème d'ordonnancement classique bi-critère* $1||F_\ell(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}))$ *est également* $\mathcal{N}P$*-difficile au sens fort.*

3. Problème $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$ est $\mathcal{N}P$-difficile au sens fort.

   En utilisant la même instance que pour le problème $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$, nous pouvons montrer que le problème $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$ est $\mathcal{N}P$-difficile au sens fort.

4. Problème $1||F_\ell\big(\sum U_j(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$ est $\mathcal{N}P$-difficile au sens fort.

   Considérons l'instance choisie pour la réduction du $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$. Avec les modifications suivantes : $d_{3r+1} = r(B+1)$, $d_j = (j-2r-1)(B+1)$ pour le travail J_j, $j \in \{3r+2, 3r+3, \ldots, 4r\}$ et $y = \sum_{i=1}^{3n}\sum_{j=1}^{i} a_i a_j + Br^2(B+1) + Br(r-1)/2$, nous montrons que le problème $1||F_\ell\big(\sum U_j(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$ est $\mathcal{N}P$-difficile au sens fort. □

5. Problème $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / \sum w'_j C_j(\mathcal{N}_1)\big)$ est $\mathcal{N}P$-difficile au sens fort.

   Soit EWCWC the decision problème associated to $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / \sum w'_j C_j(\mathcal{N}_1)\big)$. Ce problème est défini par :

   EWCWC

   Données : Un ensemble $\mathcal{N}$ de n travaux, un sous-ensemble $\mathcal{N}_1 \subset \mathcal{N}$, durée opératoire p_j et les poids w_j, w'_j pour chaque travail J_j, $1 \leq j \leq n$, deux valeurs entières Y et Y_1.

   Question : Existe-t-elle une séquence σ de $\mathcal{N}$ telle que

   $$\sum_{J_j \in \mathcal{N}} w_j C_j \leq Y \text{ et } \sum_{J_j \in \mathcal{N}_1} w'_j C_j \leq Y_1 \,?$$

   Nous montrons que la réponse au problème 3-PARTITION est "oui" si et seulement si la réponse au problème EWCWC est "oui".

   Soit une instance de 3-PARTITION, nous construisons une instance de EWCWC comme suit :

– $\mathcal{N} = \{1, 2, \ldots, 4r\}$,
– $\mathcal{N}_1 = \{1, 2, \ldots, r\}$,
– pour le travail $J_j, j \in \{1, 2, \ldots, r\} : p_j = B, w_j = 1$ et $w'_j = B^{3(r-j)}$,
– pour le travail $J_j, j \in \{r+1, r+2, \ldots, 4r\} : p_j = a_{j-r}, w_j = a_{j-r}$ (tous les travaux ont le même ratio $\frac{p_j}{w_j} = 1$),
– $Y = \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + B^2 r(r+1)/2 + r^2 B$ et $Y_1 = \sum_{i=1}^{r} (2i-1) B^{3r-3i+1}$.

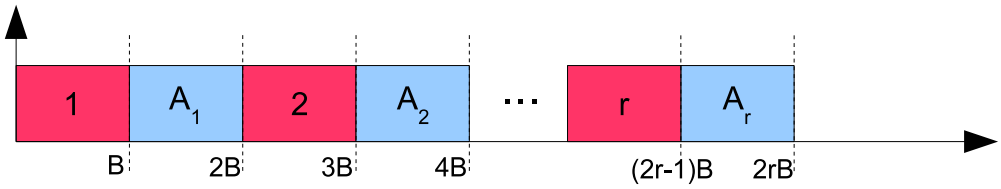

FIG. 7.5 – Une séquence optimale σ du $1||\varepsilon(\sum w_j C_j(\mathcal{N}) / \sum w'_j C_j(\mathcal{N}_1))$

Remarque que les durées opératoires et les poids w_j des travaux de $\mathcal{N}_1$ sont identiques. L'ordonnancement des travaux de $\mathcal{N}_1$ dans l'ordre non-croissant de w'_j (également dans l'ordre numérique) est donc dominant.

($\Rightarrow$) Supposons que 3-PARTITION a une réponse 'oui'. Alors, nous construisons une solution du problème EWCWC. Nous formons r blocks, où le $j$ème block contient le travail J_j suivi par les travaux correspondant aux éléments de A_j, que nous traitons dans cet ordre.

Il est facile de vérifier que :

– la pondération de la date de fin d'exécution des travaux de $\mathcal{N}_1$ est égale à :

$$\begin{aligned} \sum w'_j C_j(\mathcal{N}_1) &= B^{3r-3} \times B + B^{3r-6} \times 3B + B^{3r-9} \times 5B + \ldots + 1 \times (2r-1)B \\ &= \sum_{j=1}^{r} (2j-1) B^{3r-3j+1} = Y_1, \end{aligned}$$

– la pondération de la date de fin d'exécution des travaux de $\mathcal{N}$ est égale à :

$$\begin{aligned} \sum w_j C_j(\mathcal{N}) &= \sum w_j C_j(\mathcal{N} \setminus \mathcal{N}_1) + \sum w_j C_j(\mathcal{N}_1) \\ &= (\sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} jB^2) + (B \sum_{j=1}^{r} (2j-1)) \\ &= (\sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + B^2 r(r+1)/2) + (B \sum_{j=1}^{r} (2j-1)) \\ &= \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + B^2 r(r+1)/2 + r^2 B = Y. \end{aligned}$$

Ainsi, EWCWC a également une réponse 'oui'.

($\Leftarrow$) Supposons maintenant que EWCWC a une réponse 'oui', soit σ une solution réalisable. Nous pouvons supposer que σ est une solution dominante, dont les travaux de $\mathcal{N}_1$ sont ordonnancés dans l'ordre numérique.

Soit H_j ($j = 1, \ldots, r-1$) l'ensemble des travaux de $\mathcal{N} \setminus \mathcal{N}_1$ qui sont exécutés entre les travaux J_j et J_{j+1}, et soit H_r l'ensemble des travaux de $\mathcal{N}$ exécutés après le travail r dans la séquence σ. Nous utilisons $p(H_j)$ et $w(H_j)$ pour désigner la somme des durées opératoires et la somme des poids des travaux appartenant à H_j respectivement.

Nous avons l'inégalité suivante pour tout $k \in \{1, 2, \ldots, r\}$:

$$\begin{aligned}
\sum_{i=k+1}^{r} (2i-1)B^{3r-3i+1} &< B^{3r-3(k+1)+1} \sum_{i=k+1}^{r} (2i-1) \\
&< B^{3r-3(k+1)+1} \sum_{i=1}^{r} (2i-1) < r^2 B^{3r-3(k+1)+1} \\
&< B^2 B^{3r-3(k+1)+1} = B^{3(r-k)}.
\end{aligned}$$

Par conséquent, la date de fin d'exécution du travail J_1 ne peut pas dépasser B (car dans le cas contraire, $C_1 \geq (B+1) \Rightarrow w'_1 C_1 \geq B^{3r-3}(B+1) = B^{3r-2} + B^{3r-3} > B^{3r-2} + \sum_{i=2}^{r}(2i-1)B^{3r-3i+1} = Y_1$). Néanmoins, comme $p_1 = B$, la date de fin d'exécution du travail J_1 doit être égale à B. De plus, la date de fin d'exécution du travail J_2 ne peut pas dépasser $3B$ car sinon $w'_1 C_1 + w'_2 C_2 \geq B^{3r-2} + B^{3r-6}(3B+1) = B^{3r-2} + B^{3r-6}(3B+1) = B^{3r-2} + 3BB^{3r-6} + B^{3r-6} > B^{3r-2} + 3BB^{3r-6} + \sum_{i=3}^{r}(2i-1)B^{3r-3i+1} = Y_1$. Ainsi, on peut déduire que $p(H_1) \leq B$. De la même façon, nous pouvons montrer que la date de fin d'exécution du travail J_3 ne peut pas dépasser à $5B$. Cela veut dire que $p(H_1) + p(H_2) \leq 2B$. En étendant ce raisonnement, nous constatons que :

$(r-1)p(H_1) + (r-2)p(H_2) + \ldots + p(H_{r-1}) \leq \sum_{i=1}^{r-1} iB = Br(r-1)/2$

$\Leftrightarrow \sum_{j=1}^{r-1}(r-j)p(H_j) + \sum_{j=1}^{r} p(H_j) \leq Br(r-1)/2 + rB$

$\Leftrightarrow \sum_{j=1}^{r}(r-j+1)p(H_j) \leq Br(r+1)/2$

$\Leftrightarrow (r+1)[p(H_1) + p(H_2) + \ldots + p(H_r)] - \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2$

$\Leftrightarrow (r+1)[rB] - \sum_{j=1}^{r} jp(H_j) \leq Br(r+1)/2$

$\Leftrightarrow Br(r+1) - Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j)$

$\Leftrightarrow Br(r+1)/2 \leq \sum_{j=1}^{r} jp(H_j)$

D'autre part, nous avons

– $\sum w_j C_j(\mathcal{N}_1) = \sum_{j=1}^{r}(jB + \sum_{i=1}^{j-1} p(H_j)) = Br(r+1)/2 + \sum_{j=1}^{r-1}(r-j)p(H_j)$

$$\begin{aligned}\Rightarrow \sum w_j C_j(\mathcal{N}_1) &= Br(r+1)/2 + \sum_{j=1}^{r-1}(r-j)p(H_j) + \sum_{j=1}^{r} p(H_j) - rB \\ &= Br(r+1)/2(r+1)[rB] - \sum_{j=1}^{r} jp(H_j) - rB \\ &= Br(r+1)/2 + Br^2 - \sum_{j=1}^{r} jp(H_j).\end{aligned}$$

– $\sum w_j C_j(\mathcal{N} \setminus \mathcal{N}_1) = \sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} jBw(H_j)$

Comme $\sum w_j C_j(\mathcal{N}) \leq Y$ et $p(H_j) = w(H_j)$ pour tout $j \in \{1,\ldots,r\}$, nous avons

$\sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} jBp(H_j) + Br(r+1)/2 + Br^2 - \sum_{j=1}^{r} jp(H_j) \leq \sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + B^2 r(r+1)/2 + Br^2$

$\Leftrightarrow (B-1)\sum_{j=1}^{r} jp(H_j)) + Br(r+1)/2 \leq B^2 r(r+1)/2$

$\Leftrightarrow (B-1)\sum_{j=1}^{r} jp(H_j)) \leq (B-1)Br(r+1)/2$

$\Leftrightarrow \sum_{j=1}^{r} jp(H_j)) \leq Br(r+1)/2$

Par conséquent, $\sum_{j=1}^{r} jp(H_j) = Br(r+1)/2$

Puisque $\forall j \in \{1,\ldots,r\}$, $p(H_1) + \ldots + p(H_j) \leq jB$ nous pouvons déduire que $\forall j \in \{1,\ldots,r\}, p(H_j) = B$. Par conséquent, la partition de A en $H_1,\ldots,H_r$ montre que le problème 3-PARTITION est à réponse 'oui'. □

7.5 Problèmes ouverts

Proposition 7.5.1 *Les problèmes suivants restent ouverts :*

1. *Problème* $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$,
2. *Problème* $1||F_\ell\big(\sum C_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$ *et problème* $1||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum U_j(\mathcal{N}_1)\big)$,
3. *Problème* $1||F_\ell\big(\sum U_j(\mathcal{N}), \sum C_j(\mathcal{N}_1)\big)$ *et problème* $1||\varepsilon\big(\sum U_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$,
4. *Problème* $1||F_\ell\big(\sum U_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$,
5. *Problème* $1||F_\ell\big(\sum U_j(\mathcal{N}), \sum U_j(\mathcal{N}_1)\big)$ *et problème* $1||\varepsilon\big(\sum U_j(\mathcal{N}) / \sum U_j(\mathcal{N}_1)\big)$.

Conclusion

Dans ce chapitre, nous avons élaboré des preuves de complexité pour les problèmes d'ordonnancement à une seule machine. Les approches les plus utilisées en pratique telles que ε-contrainte, programmation par objectif et combinaison linéaire sont abordées. L'objectif est de minimiser à la fois un critère régulier pour tous les travaux parmi l'ensemble des critères $\{C_{\max}(\mathcal{N}), L_{\max}(\mathcal{N}), \sum C_j(\mathcal{N}), \sum w_j C_j(\mathcal{N}), \sum U_j(\mathcal{N}, \sum T_j(\mathcal{N}), \sum w_j T_j(\mathcal{N}))\}$ et un autre critère pour un sous-ensemble $\mathcal{N}_1$ des travaux choisi parmi $\{C_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_1), \sum C_j(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_1), \sum T_j(\mathcal{N}_1), \sum w_j T_j(\mathcal{N}_1)\}$. Tous les combinaisons possibles de ces deux critères ont été étudiées et la synthèse des résultats est présentée dans la Figure 7.1. Dans une future recherche, nous étudierions les problèmes qui restent ouverts.

Chapitre 8

Problème d'ordonnancement à machines parallèles

Les problèmes d'ordonnancement des travaux interférants sur m machines parallèles identiques sont considérés dans ce chapitre. Nous nous intéressons à deux critères réguliers : minimiser la date de fin d'exécution du dernier travail de $\mathcal{N}$ (resp. de $\mathcal{N}_1$) notée $C_{max}(\mathcal{N})$ (resp. $C_{max}(\mathcal{N}_1)$), la somme des dates de fin des travaux $\sum C_j(\mathcal{N})$ (resp. $\sum C_j(\mathcal{N}_1)$) ou la somme pondérée $\sum w_j C_j(\mathcal{N})$ (resp. $\sum w_j C_j(\mathcal{N})$). Toutes les combinaisons possibles des critères sur plusieurs approches de résolution pour l'optimisation multicritère sont étudiées. Ces problèmes seront montrés $\mathcal{NP}$-difficiles. Pour résoudre ceux qui sont $\mathcal{NP}$-difficiles au sens ordinaire, les méthodes exactes basées sur la programmation dynamique sont développées.

8.1 Introduction

Nous considérons un ensemble $\mathcal{N}$ de travaux à ordonnancer sur m machines parallèles identiques, où m est fixé. Soit $\mathcal{N}_1$ un sous-ensemble de $\mathcal{N}$. L'objectif est de déterminer un ordonnancement qui minimise à la fois un critère régulier global sur tous les travaux et un autre critère régulier local sur l'ensemble $\mathcal{N}_1$. Plus particulièrement, nous nous intéressons à deux critères réguliers : minimiser la date de fin d'exécution

du dernier travail, la somme des dates de fin des travaux $\sum C_j$ ou la somme pondérée des dates de fin $\sum w_j C_j$. Il s'agit ici d'étudier plusieurs approches de résolution pour l'optimisation multicritère telles que l'approche ε-contrainte, la combinaison linéaire et le goal programming. Toutes les combinaisons possibles des critères sont considérées. En se basant sur les résultats de complexité montrés dans le cas d'un seule machine ou dans le cas monocritère, nous déterminons les statuts de $\mathcal{NP}$-complétude. Les problèmes étudiés sont $\mathcal{NP}$-difficiles. Pour résoudre ceux qui sont $\mathcal{NP}$-difficiles au sens ordinaire, les méthodes exactes basées sur la programmation dynamique seront développées.

La table 8.1 présente nos nouveaux résultats :

Fonction objectif	Approche étudiée			Références
	(F_ℓ)	(ε)	(GP)	
$C_{\max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)$	BNP	BNP	BNP	Prop. 8.2.1
$C_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)$	BNP	BNP	SNP	Prop. 8.2.1
$\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1)$	Open	BNP	BNP	Prop. 8.3.1, 8.2.1
$\sum w_j C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)$	BNP	BNP	BNP	Prop. 8.2.1
$\sum w_j C_j(\mathcal{N}), w_j C_j(\mathcal{N}_1)$	BNP	SNP	SNP	Prop. 8.2.1, 8.2.2

(F_ℓ) : l'approche par combinaison linéaire
(ε) : l'approche ε-contrainte
(GP) : l'approche de goal programming

TAB. 8.1 – Ordonnancement avec travaux interférants sur machines parallèles

L'organisation de ce chapitre est la suivante. Les preuves de complexité pour ces problèmes d'ordonnancements sont présentés dans la section 8.2. Dans la section 8.4, une formulation récursive générale est proposée pour résoudre les problèmes d'ordonnancement sur m machines parallèles. Nous résolvons par la suite les problèmes $\mathcal{NP}$-difficiles au sens ordinaire.

8.2 Résultats de complexité

Proposition 8.2.1 *Les problèmes suivants sont $\mathcal{NP}$-difficiles au sens ordinaire :*

1. $Pm||\varepsilon\big(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1)\big)$
2. $Pm||\varepsilon\big(C_{\max}(\mathcal{N}) / \sum w_j C_j(\mathcal{N}_1)\big)$ *et* $Pm||F_\ell\big(\sum w_j C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$
3. $Pm||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / C_{\max}(\mathcal{N}_1)\big)$ *et* $Pm||F_\ell\big(C_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$
4. $Pm||\varepsilon\big(C_{\max}(\mathcal{N}) / C_{\max}(\mathcal{N}_1)\big)$ *et* $Pm||F_\ell\big(C_{\max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$
5. $Pm||F_\ell\big(\sum w_j C_j(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$

Preuve. Problème 1 est $\mathcal{NP}$-difficiles puisque $1||\varepsilon\big(\sum C_j(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$ est $\mathcal{NP}$-difficiles (voir proposition 7.3.1 du chapitre précédent). Problèmes 2-6 sont $\mathcal{NP}$-difficiles puisque les problèmes de $Pm||C_{\max}$ et $Pm||\sum w_jC_j$ sont $\mathcal{NP}$-difficiles (voir [153] et [30]). □

Proposition 8.2.2 *Le problème $Pm||\varepsilon\big(\sum w_jC_j(\mathcal{N})/\sum w_jC_j(\mathcal{N}_1)\big)$ est $\mathcal{NP}$-difficile au sens fort.*

Preuve. Vrai car le problème $1||\varepsilon\big(\sum w_jC_j(\mathcal{N})/\sum w_jC_j(\mathcal{N}_1)\big)$ est $\mathcal{NP}$-difficile au sens fort (cf. proposition 7.4.1 du chapitre précédent). □

8.3 Problèmes ouverts

Proposition 8.3.1 *Le problème $Pm||F_\ell\big(\sum C_j(\mathcal{N}),\sum C_j(\mathcal{N}_1)\big)$ reste ouvert.*

8.4 Algorithme de programmation dynamique

8.4.1 Formulation générale de programmation dynamique

Nous considérons d'abord le cas de deux machines parallèles. Le problème est dénoté par $P||\varepsilon(Z_1(\mathcal{A})/Z_2(\mathcal{B}))$. Dans ce qui suit, les deux cas sont possibles : $\mathcal{A} = \mathcal{N} \wedge \mathcal{B} = \mathcal{N}_1$ ou $\mathcal{A} = \mathcal{N}_1 \wedge \mathcal{B} = \mathcal{N}$ et Z_1 et Z_2 appartiennent à $\{C_{\max}, \sum C_j, \sum C_j, \sum w_jC_j\}$.

Comme précédemment, on suppose que les travaux dans $\mathcal{N}_1$ sont numérotés de 1 à $n_1 = |\mathcal{N}_1|$ et que les travaux dans $\mathcal{N} \setminus \mathcal{N}_1$ sont numérotés de $n_1 + 1$ à n.

Soit $F(i, j, P_1, Q_1, Q_2)$ le coût minimal pour ordonnancer les travaux de $\{1, 2, ..., i\} \in \mathcal{N}_1$ et les travaux de $\{n_1 + 1, n_1 + 2, ..., j\} \in \mathcal{N} \setminus \mathcal{N}_1$ de sorte que la somme des durées d'exécution des travaux sur $M1$ est égale à P_1. Q_1 et Q_2 dépendent des fonctions objectif Z_1 et Z_2. Clairement, la somme des durées d'exécution de tous les travaux notée $P = \sum_{j=1}^{n} p_j$ est une borne supérieure de P_1. Soit Q'_1 (resp. Q'_2) une borne supérieure de Q_1 (resp. Q_2).

La décision consiste à affecter un travail de $\mathcal{N}_1$ ou de $\mathcal{N} \setminus \mathcal{N}_1$ sur $M1$ ou sur $M2$, ce qui fait quatre cas possibles. Nous donnons d'abord une formulation générale de l'algorithme ProgDyn et présentons par le suite son application pour résoudre plusieurs problèmes d'ordonnancement.

Les décisions sont comme suit (dans le cas de deux machines) :

- affecter le travail i appartenant à $\mathcal{N}_1$ sur $M1$
- affecter le travail i appartenant à $\mathcal{N}_1$ sur $M2$
- affecter le travail j appartenant à $\mathcal{N} \setminus \mathcal{N}_1$ sur $M1$
- affecter le travail j appartenant à $\mathcal{N} \setminus \mathcal{N}_1$ sur $M2$

Ces décisions peuvent être facilement étendues au cas de plus de deux machines (ce qui conduit à $F(i, j, P_1, P_2, \ldots, P_{m-1}, Q_1, Q_2, \ldots, Q_m)$). La relation récursive générale (dans le cas de deux machines) est donnée par $F(i, j, P_1, Q_1, Q_2)$:

$$F(0, n_1, 0, 0, 0) = \quad 0$$

$$F(i, j, P_1, Q_1, Q_2) = \quad +\infty, \left(\begin{array}{l} \forall i > n_1, \\ \forall j \leq n_1, \\ \forall (P_1, Q_1, Q_2) \end{array} \right)$$

$$F(i, j, P_1, Q_1, Q_2) = \quad +\infty, \left(\begin{array}{l} \forall i \in \{0, 1, \ldots, n_1\}, \\ \forall j \in \{n_1, n_1 + 1, \ldots, n\}, \\ \forall ((P_1, Q_1, Q_2) < (0, 0, 0) \vee (P_1, Q_1, Q_2) > (P, Q'_1, Q'_2)) \end{array} \right)$$

$$F(i, j, P_1, Q_1, Q_2) = \min \quad \left\{ \begin{array}{l} F(i-1, j, P_1 - p_i, Q_{11}, Q_{21}) + F_1, \\ F(i-1, j, P_1, Q_{12}, Q_{22}) + F_2, \\ F(i, j-1, P_1 - p_j, Q_{13}, Q_{23}) + F_3, \\ F(i, j-1, P_1, Q_{14}, Q_{24}) + F_4 \end{array} \right\}, \left(\begin{array}{l} \forall i \in \{1, \ldots, n_1\} \\ \forall j \in \{n_1 + 1, \ldots, n\} \\ \forall 0 \leq P_1 \leq P \\ \forall 0 \leq Q_1 \leq Q'_1 \\ \forall 0 \leq Q_2 \leq Q'_2 \end{array} \right)$$

F_k ($1 \leq k \leq 4$) est lié à décision k et lié à la fonction objectif. Q_{1k}, Q_{2k} sont fonction de Q_1 et Q_2 et de la décision k.

Dans la suite, nous présentons quelques implémentations de cette formulation récursive.

8.4.2 Une application de la formulation générale

Ce algorithme de ProgDyn peut être appliqué au problème d'ordonnancement classique $P2||C_{max}$ comme suit (la formulation générale est simplifiée) :

$$F(0,0) = \quad 0$$

$$F(i,P_1) = \quad +\infty, \left(\begin{array}{l} \forall i \in \{0,1,\ldots,n\}, \\ \forall (P_1 < 0 \vee P_1 > P) \end{array} \right)$$

$$F(i,P_1) = \min \quad \left\{ \begin{array}{l} F(i-1,P_1-p_i)+p_i, \\ F(i-1,P_1), \end{array} \right\}, \left(\begin{array}{l} \forall i \in \{1,\ldots,n\} \\ \forall 0 \leq P_1 \leq P \end{array} \right)$$

La valeur de la solution optimale est égale à $min_{P/2 \leq P_1 \leq P} \max(F(n,P_1), P-P_1)$. Le solution correspondante est obtenue en utilisant un algorithme classique de backtracking. La complexité de cet algorithme de ProgDyn est en $O(nP)$ (même complexité que dans [187]).

On introduit les notations suivantes : $P_{(i,j)} = \sum_{1 \leq k \leq i} p_k + \sum_{n_1+1 \leq k \leq j} p_k$. Cette quantité $P_{(i,j)} - P_1$ correspond à la date de fin d'exécution des travaux sur $M2$. De plus, supposons que les travaux sont numérotés selon règle WSPT : $p_1/w_1 \leq p_2/w_2 \leq \ldots \leq p_{n_1}/w_{n_1}$ et $p_{n_1+1}/w_{n_1+1} \leq p_{n_1+2}/w_{n_1+2} \leq \ldots \leq p_n/w_n$ (nous considérons $w_j = 1$, $j = 1..n$, pour le critère $\sum C_j$).

Nous présentons par la suite l'application de la formulation de ProgDyn générale aux problèmes concernant les fonctions objectif suivantes :

- $\sum C_j(\mathcal{N}_1)$ et $\sum C_j(\mathcal{N})$,
- $\sum w_j C_j(\mathcal{N}_1)$ et $C_{\max}(\mathcal{N})$,
- $\sum w_j C_j(\mathcal{N})$ et $C_{\max}(\mathcal{N}_1)$,
- $C_{\max}(\mathcal{N})$ et $C_{\max}(\mathcal{N}_1)$.

8.4.3 Problèmes avec les fonction objectifs $\sum C_j(\mathcal{N}_1)$ et $\sum C_j(\mathcal{N})$

Considérons par exemple le problème d'ordonnancement $P2||\varepsilon(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N}))$ [114]. It faut minimiser $\sum C_j(\mathcal{N}_1)$ et respecter la contrainte de $\sum C_j(\mathcal{N}) \leq \varepsilon$. Nous considérons que Q_1 correspond à $\sum C_j(\mathcal{N})$; $Q_2 = 0$ (est omis dans la relation récursive). La fonction objectif est de minimiser $F(i,j,P_1,Q_1) = \sum C_j(\mathcal{N}_1)$. La relation récursive générale devient :

$$F(0, n_1, 0, 0) = \quad 0$$

$$F(i, j, P_1, Q_1) = \quad +\infty, \left(\begin{array}{l} \forall i > n_1, \\ \forall j \leq n_1, \\ \forall (P_1, Q_1) \end{array} \right)$$

$$F(i, j, P_1, Q_1) = \quad +\infty, \left(\begin{array}{l} \forall i \in \{0, 1, \ldots, n_1\}, \\ \forall j \in \{n_1, n_1 + 1, \ldots, n\}, \\ \forall ((P_1, Q_1) < (0, 0) \vee (P_1, Q_1) > (P, \varepsilon)) \end{array} \right)$$

$$F(i, j, P_1, Q_1) = \min \left\{ \begin{array}{l} F(i-1, j, P_1 - p_i, Q_1 - P_1) + P_1, \\ F(i-1, j, P_1, Q_1 - w_i(P_{(i,j)} - P_1)) + P_{(i,j)} - P_1, \\ F(i, j-1, P_1 - p_j, Q_1 - P_1), \\ F(i, j-1, P_1, Q_1 - (P_{(i,j)} - P_1)) \end{array} \right\}, \left(\begin{array}{l} \forall i \in \{1, \ldots, n_1\} \\ \forall j \in \{n_1 + 1, \ldots, n\} \\ \forall 0 \leq P_1 \leq P \\ \forall 0 \leq Q_1 \leq \varepsilon \end{array} \right)$$

La solution optimale est donnée par $min_{(0 \leq P_1 \leq P \wedge 0 \leq Q_1 \leq \varepsilon)} F(n_1, n, P_1, Q_1)$. Le temps de calcul de cet algorithme est en $O(n^2 P\varepsilon)$. Cette méthode peut être généralisée pour le cas de m machines et nous obtenons la proposition suivante.

Proposition 8.4.1 *Une solution optimale pour le problème* $Pm||\varepsilon\big(\sum C_j(\mathcal{N}_1) / \sum C_j(\mathcal{N})\big)$ *peut être déterminée en* $O(n^2 P^{m-1}\varepsilon)$.

Considérons maintenant le problème $P2||GP(\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1))$, qui est équivalent à déterminer une solution qui respecte $\sum C_j(\mathcal{N}) \leq \varepsilon$ et $\sum C_j(\mathcal{N}_1) \leq \varepsilon_1$. Une solution réalisable est donnée par $F(n_1, n, P_1, Q_1) \leq \varepsilon_1$, ($0 \leq P_1 \leq P$ et $0 \leq Q_1 \leq \varepsilon$). Le temps de calcul est en $O(n^2 P\varepsilon)$.

Considérons le problème $P2||\varepsilon(\sum C_j(\mathcal{N}) / \sum C_j(\mathcal{N}_1))$. On cherche la valeur la plus petite de Q_1 telle qu'il existe une valeur de P_1 avec $F(n_1, n, P_1, Q_1) \leq \varepsilon$. On énumère toutes les valeurs possibles de (Q_1, P_1) à partir de $(0, 0)$ à $(n \times w_{max} \times P, P)$ où $w_{max} = \max_{j \in \mathcal{N}} w_j$. Avec un couple de valeur (Q_1, P_1) tel que $F(n_1, P_1, Q_1) \leq \varepsilon$ est trouvée, l'algorithme s'arrête et retourne la valeur courante de Q_1 qui définit la valeur minimum de $\sum C_j(\mathcal{N}_1)$. Dans le cas contraire, si aucune couple (Q_1, P_1) n'est trouvé, l'algorithme retourne '*aucune solution réalisable*'. Le temps de calcul est borné par $O(n^3 P^2 w_{max})$. Notons qu'une légère modification de la relation récursive pourrait mener à une complexité plus intéressante en $O(n^2 P\varepsilon)$.

De la même façon, pour résoudre le problème $P2||F_\ell(\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1))$, nous consi-

dérons la formulation de ProgDyn générale avec ε une borne supérieure de $\sum C_j(\mathcal{N})$. On définit par exemple $F_\ell(\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1)) = a\sum C_j(\mathcal{N}) + b\sum C_j(\mathcal{N}_1)$. Nous énumérons toutes les valeurs possibles de (P_1, Q_1) de $(0,0)$ jusqu'à $(P, nw_{max}P)$ et la solution optimale est retournée avec la valeur minimum de $a \times Q_1 + b \times F(n_1, n, P_1, Q_1)$. Le temps de calcul est en $O(n^3P^2w_{max})$. Notons qu'un autre algorithm peut être donné avec une meilleure complexité en $(O(n^2P))$ en reformulant $P2||a\sum C_j(\mathcal{N}) + b\sum C_j(\mathcal{N}_1)$ comme $P2||\sum w_jC_j(\mathcal{N})$ ($w_j = a$ si $j \in \mathcal{N} \setminus \mathcal{N}_1$, et $w_j = a + b$ si $j \in \mathcal{N}_1$).

8.4.4 Problèmes avec les fonction objectifs $\sum w_jC_j(\mathcal{N}_1)$ and $C_{\max}(\mathcal{N})$

Nous considérons maintenant le problème d'ordonnancement à deux machines parallèles noté $P2||\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$. Nous avons $C_{\max}(\mathcal{N}) \leq \varepsilon$ et supposons que $\varepsilon < P$. Nous considérons que Q_1 correspond à la somme des durées opératoires des travaux de $\mathcal{N}_1$ ordonnancés sur $M1$; et Q_2 est égal à 0 (omis dans la relation récursive). Nous proposons donc $Q'_1 = \sum_{j\in_1}$ comme une borne supérieure de Q. Il est clair que $Q'_1 \leq P$. La fonction objectif est de minimiser $F(i,j,P_1) = \sum w_jC_j(\mathcal{N}_1)$. Le makespan est donné par $\max(P_1, P - P_1)$. La relation récursive générale devient :

$$F(0, n_1, 0, 0) = \quad 0$$

$$F(i,j,P_1,Q_1) = \quad +\infty, \left(\begin{array}{l} \forall i > n_1, \\ \forall j \leq n_1, \\ \forall (P_1, Q_1) \end{array}\right)$$

$$F(i,j,P_1,Q_1) = \quad +\infty, \left(\begin{array}{l} \forall i \in \{0,1,\ldots,n_1\}, \\ \forall j \in \{n_1, n_1+1, \ldots, n\}, \\ \forall ((P_1,Q_1) < (0,0) \vee (P_1,Q_1) > (\varepsilon, Q'_1)) \end{array}\right)$$

$$F(i,j,P_1,Q_1) = \min \left\{\begin{array}{l} F(i-1,j,P_1-p_i,Q_1-p_i) + w_iQ_1, \\ F(i-1,j,P_1,Q_1) + w_i(P_{(i,0)} - Q_1), \\ F(i,j-1,P_1-p_j,Q_1), \\ F(i,j-1,P_1,Q_1) \end{array}\right\}, \left(\begin{array}{l} \forall i \in \{1,\ldots,n_1\}; \\ \forall j \in \{n_1+1,\ldots,n\}; \\ \forall \max(0, P-\varepsilon) \leq P_1 \leq \varepsilon \\ \forall 0 \leq Q_1 \leq Q'_1 \end{array}\right)$$

La solution optimale est donnée par la valeur minimum de $F(n_1, n, P_1, Q_1)$ avec $\max(P_1, P - P_1) \leq \varepsilon$. Le temps de calcul de cet algorithme est en $O(n^2P\varepsilon)$. Cet algorithme peut être généralisé au cas de m machines, nous obtenons proposition suivante.

Proposition 8.4.2 *Une solution optimale pour le problème* $Pm||\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$ *peut être déterminée en* $O(n^2P^m\varepsilon^{m-1})$.

En suivant le même raisonnement que pour les critères $\sum C_j(\mathcal{N}_1))$ et $\sum C_j(\mathcal{N})$, nous pouvons déduire les résultats suivants :

- le problème $P2||GP(C_{\max}(\mathcal{N}), \sum w_jC_j(\mathcal{N}_1))$ peut être résolu en $O(n^2P\varepsilon)$,
- le problème $Pm||\varepsilon\big(C_{\max}(\mathcal{N})/\sum w_jC_j(\mathcal{N}_1)\big)$ peut être résolu en $O(n^2P\varepsilon)$,
- le problème $P2||F_l(C_{\max}(\mathcal{N}), \sum w_jC_j(\mathcal{N}_1))$ peut être résolu en $O(n^2P^2)$.

8.4.5 Problèmes avec les fonction objectifs $\sum w_jC_j(\mathcal{N})$ and $C_{\max}(\mathcal{N}_1)$

Nous considérons maintenant le problème d'ordonnancement à deux machines parallèles noté $P2||\varepsilon\big(\sum w_jC_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$. Q_1 et Q_2 correspondent à $C_{\max}(\mathcal{N}_1)$ sur $M1$ et $M2$ respectivement (rappelons que P_1 est le makespan sur $M1$). La fonction objectif est de minimiser $F(i,j,P_1,Q_1,Q_2) = \sum w_jC_j(\mathcal{N})$. La relation récursive générale devient :

$$F(0,n_1,0,0,0) = \quad 0$$

$$F(i,j,P_1,Q_1,Q_2) = \quad +\infty, \left(\begin{array}{l} \forall i > n_1, \\ \forall j < n_1, \\ \forall (P_1,Q_1,Q_2) \end{array}\right)$$

$$F(i,j,P_1,Q_1,Q_2) = \quad +\infty, \left(\begin{array}{l} \forall i \in \{0,1,\ldots,n_1\}, \\ \forall j \in \{n_1,n_1+1,\ldots,n\}, \\ \forall ((P_1,Q_1,Q_2) < (0,0,0) \vee (P_1,Q_1,Q_2) > (P,\varepsilon,\varepsilon)) \end{array}\right)$$

$$F(i,j,P_1,Q_1,Q_2) = \min \left\{\begin{array}{l} F(i-1,j,P_1-p_i,P_1-p_i,Q_2)+w_iP_1, \\ F(i-1,j,P_1,Q_1,P_{(i,j)}-P_1-p_i)+w_i(P_{(i,j)}-P_1), \\ F(i,j-1,P_1-p_j,Q_1,Q_2)+w_jP_1, \\ F(i,j-1,P_1,Q_1,Q_2)+w_j(P_{(i,j)}-P_1) \end{array}\right\}, \left(\begin{array}{l} \forall i \in \{1,\ldots,n_1\} \\ \forall j \in \{n_1+1,\ldots,n\} \\ \forall 0 \le P_1 \le P \\ \forall 0 \le Q_1 \le \varepsilon \\ \forall 0 \le Q_2 \le \varepsilon \end{array}\right)$$

La solution optimale est donnée par $min_{(0\le P_1\le P \wedge 0\le Q_1\le\varepsilon\wedge 0\le Q_2\le\varepsilon)}F(n_1,n,P_1,Q)$. Le temps de calcul de l'algorithme est en $O(n^2P\varepsilon^2)$. En généralisant pour le cas de m machines, nous obtenons la proposition suivante.

Proposition 8.4.3 *Une solution optimale pour le problème* $Pm||\varepsilon\big(\sum w_jC_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ *peut être déterminée en* $O(n^2P^{m-1}\varepsilon^m)$.

En suivant le même raisonnement que pour les critères $\sum C_j(\mathcal{N}_1))$ et $\sum C_j(\mathcal{N})$, nous pouvons déduire les résultats suivants :

- problème $P2||GP(C_{\max}(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}))$ peut être résolu en $O(n^2 P\varepsilon^2)$,
- problème $Pm||\varepsilon\big(C_{\max}(\mathcal{N}_1) / \sum w_j C_j(\mathcal{N})\big)$ peut être résolu en $O(n^2 P\varepsilon^2)$,
- problème $P2||F_l(C_{\max}(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}))$ peut être résolu en $O(n^2 P^3)$.

8.4.6 Problèmes avec les fonction objectifs $C_{\max}(\mathcal{N})$ et $C_{\max}(\mathcal{N}_1)$

Nous considérons maintenant le problème d'ordonnancement à deux machines parallèles noté $P2||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$. Q_1 et Q_2 correspondent à $C_{\max}(\mathcal{N}_1)$ sur $M1$ et $M2$ respectivement (rappelons que P_1 est le makespan sur $M1$). La fonction objectif est de minimiser $F(i, j, P_1, Q_1, Q_2)$ qui correspond à makespan sur $M2$. La relation récursive générale devient :

$$F(0, n_1, 0, 0, 0) = \quad 0$$

$$F(i, j, P_1, Q_1, Q_2) = \quad +\infty, \left(\begin{array}{l} \forall i > n_1, \\ \forall j < n_1, \\ \forall (P_1, Q_1, Q_2) \end{array} \right)$$

$$F(i, j, P_1, Q_1, Q_2) = \quad +\infty, \left(\begin{array}{l} \forall i \in \{0, 1, \ldots, n_1\}, \\ \forall j \in \{n_1, n_1 + 1, \ldots, n\}, \\ \forall ((P_1, Q_1, Q_2) < (0, 0, 0) \vee (P_1, Q_1, Q_2) > (P, \varepsilon, \varepsilon)) \end{array} \right)$$

$$F(i, j, P_1, Q_1, Q_2) = \min \quad \left\{ \begin{array}{l} F(i-1, j, P_1 - p_i, P_1 - p_i, Q_2), \\ F(i-1, j, P_1, Q_1, P_{(i,j)} - P_1 - p_i) + p_i, \\ F(i, j-1, P_1 - p_j, Q_1, Q_2), \\ F(i, j-1, P_1, Q_1, Q_2) + p_j \end{array} \right\}, \left(\begin{array}{l} \forall i \in \{1, \ldots, n_1\} \\ \forall j \in \{n_1 + 1, \ldots, n\} \\ \forall 0 \leq P_1 \leq P \\ \forall 0 \leq Q_1 \leq \varepsilon \\ \forall 0 \leq Q_2 \leq \varepsilon \end{array} \right)$$

La solution optimale est donnée par $min_{(0 \leq P_1 \leq P \wedge 0 \leq Q_1 \leq \varepsilon \wedge 0 \leq Q_2 \leq \varepsilon)} F(n_1, n, P_1, Q)$. Le temps de calcul de l'algorithme est en $O(n^2 P\varepsilon^2)$. En généralisant pour le cas de m machines, nous obtenons la proposition suivante.

Proposition 8.4.4 *Une solution optimale pour le problème $Pm||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$ peut être déterminée en $O(n^2 P^{m-1} \varepsilon^{2m})$.*

En suivant le même raisonnement que pour les critères $\sum C_j(\mathcal{N}_1))$ et $\sum C_j(\mathcal{N})$, nous

pouvons déduire les résultats suivants :

- le problème $P2||GP(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}))$ peut être résolu en $O(n^2P\varepsilon^2)$,
- le problème $Pm||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ peut être résolu en $O(n^2P\varepsilon^2)$,
- le problème $P2||F_l(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}))$ peut être résolu en $O(n^2P^3)$.

8.5 Conclusion

Dans ce chapitre, les problèmes d'ordonnancement sur machines parallèles avec des travaux interférants sont étudiés. Leur $\mathcal{N}P$-complétude est mise en évidence. Afin de résoudre les problèmes $\mathcal{N}P$-difficiles au sens ordinaire, nous proposons une formulation de ProgDyn générale qui permet de résoudre les problèmes d'ordonnancements sur m machines parallèles identiques. De plus, cette formulation peut être appliquée pour résoudre le cas de machines uniformes. Dans une future recherche, l'existence des schémas d'approximation sur ces problèmes d'ordonnancement abordés sera à étudier.

Chapitre 9

Conclusion et perspectives de la seconde partie

Nous avons abordé dans cette partie une nouvelle classe de problèmes d'ordonnancement avec travaux interférants. Le contexte de cette étude est justifié par des contextes industriels, où à la fois un objectif doit être minimisé pour l'ensemble des travaux, et un objectif doit être satisfait pour un sous-ensemble de travaux. Les problèmes de base à une machine et à machines parallèles identiques sont abordés : des cas polynomiaux sont identifiés, des cas sont montrés NP-complets et quelques problèmes restent ouverts. Des algorithmes de programmation dynamiques sont proposés, dont une formulation générale pour les problèmes à machines parallèles.

D'autres techniques de résolution comme les (meta-)heuristiques et les méthodes exactes comme les procédures par séparation et évaluation sont à étudier [198, 197]. Des schémas d'approximation peuvent aussi être proposés pour les problèmes à machines parallèles. Les problèmes d'atelier (flowshop, jobshop) peuvent également être considérés [119, 123, 112, 113, 196].

Conclusion générale et perspectives

Le travail présenté se situe dans le cadre de l'étude des problèmes d'ordonnancement qui se posent dans un environnement à une seule machine et à machines parallèles. Nous avons abordé deux grandes familles de problèmes d'ordonnancement multicritère de travaux indépendants : les problèmes de type JàT qui font l'objet des chapitres 1 à 5 et les problèmes avec travaux interférants qui font l'objet des chapitres 6 à 9.

Conclusion générale

Nous faisons par la suite une synthèse des principaux résultats obtenus, et précisons les références aux articles soumis dans des revues.

Le chapitre 1 a introduit de façon succinte l'approche JàT et ses objectifs ainsi que les différents niveaux de décision qui apparaissent dans un système de production.

Au chapitre 2, nous avons étudié le problème $1|d_i = d|\sum w_i T_i$. Nous avons montré que l'algorithme proposé par Lawler et Moore [150] ne peut pas être généralisé au cas de machines parallèles. Nous avons alors proposé une nouvelle formulation de ce problème par la programmation dynamique, qui permet de résoudre le cas d'une seule machine et qui peut s'étendre au cas de machines parallèles. Ces résultats ont fait l'objet d'un article en deuxième lecture dans le journal "*European Journal of Operational Research*" [117].

Au chapitre 3, nous avons étudié le problème $Q|p_i = p, d_i \in D, |D| = \ell|\sum(\alpha_i E_i + \beta_i T_i)$ avec le cas particulier à 1 machine et le cas particulier à 1 machine avec une seule famille ($|D| = 1$). Pour ces problèmes, que nous avons montré polynomiaux, il s'agit de déterminer les matrices de coûts associés à l'affectation d'un travail à une position

et de résoudre le problème d'affectation en utilisant les méthodes classiques [111]. Ensuite, nous avons étudié le problème $Pm|d_i = d, non - restrictive|\sum(\alpha_i E_i + \beta_i T_i)$ ainsi que le cas à une machine. Nous avons proposé un schéma d'approximation PTAS pour résoudre chaque problème. Ce résultat est en cours de relecture dans le journal "*Theoretical Computer Science*" [106].

Au chapitre 4, nous avons tenu compte en plus du coût associé à la décision de la date de fin souhaitée commune. Les problèmes abordés sont notés $P|p_i = p, d_i \in D, |D| = \ell|\sum(\alpha_i E_i + \beta_i T_i + \gamma_i d_i)$, avec le cas particulier à une machine, et à une machine avec $|D| = 1$. En montrant de nouvelles propriétés et avec le même raisonnement que celui présenté dans le chapitre 3, le problème d'ordonnancement a été montré polynomial. Ce résultat a fait l'objet d'un article en cours de relecture dans le journal "*European Journal of Operational Research*" [108]. Ensuite, pour le cas d'une seule machine avec une date de fin souhaitée commune à déterminer et durées opératoires quelconques, nous avons montré que le problème est équivalent au problème JàT classique avec l'objectif de l'avance-retard pondéré et une date de fin souhaitée commune non-restrictive. Le problème considéré accepte également un PTAS qui nécessite une autre transformation de l'instance. A la fin du chapitre, nous avons proposé une borne inférieure basée sur la relaxation Lagrangienne pour le problème à machines parallèles. Cette borne est évaluée en comparaison avec une borne supérieure obtenue par une recherche tabou. Le résultat montre que la déviation est très petite si le nombre de travaux ne dépasse pas $n = 40$.

Le chapitre 5 a présenté une conclusion de la première partie.

Le chapitre 6 a introduit le contexte et la motivation pratique de l'ordonnancement des travaux interférants. Une petite comparaison avec l'ordonnancement multi-agent a permis de déduire rapidement certains résultats connus et de vérifier la différence entre ces deux types de problèmes.

Au chapitre 7, nous avons considéré les problèmes d'ordonnancement dans le cas d'une seule machine. L'approche de combinaison linéaire et l'approche ε-contrainte ont été étudiées. Toutes les combinaisons possible des critères $f \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_j C_j, \sum U_j, \sum T_j, \sum w_j T_j\}$ sur $\mathcal{N}$ et $\mathcal{N}_1$ ont été pris en compte. Pour l'approche par combinaison linéaire, onze problèmes polynomiaux ont été identifiés, sept problèmes NP-difficiles au sens fort et 2 NP-difficiles au sens faible. Six problèmes restent ouverts. Certains de ces résultats sont étendus à l'approche ε-contrainte, d'autres ne s'étendent

pas et les classes de complexité peuvent changer selon l'approche utilisée.

Le chapitre 8 est dédié au cas à machines parallèles et un algorithme général de programmation dynamique est proposé, ainsi que son application à quelques cas particuliers NP-difficiles au sens ordinaire.

Une partie des résultats des chapitres 7 et 8 fait l'objet d'un article en cours de relecture dans le journal "Operations Research" [121].

Le chapitre 9 a présenté une conclusion de la seconde partie.

Perspectives

Les perspectives de recherche qui découlent de ce travail de thèse sont très nombreuses.

Tout d'abord, il serait intéressant de mettre en oeuvre des méthodes de résolution pour les problèmes avec travaux interférants dans un contexte industriel réel, comme par exemple pour la production de roulement à billes (entreprise SKF MDGBB) ou pour la production de shampoings. Ces contextes réels permettraient de valider l'intérêt de l'étude sur un plan applicatif.

Ensuite, les perspectives de recherche pour un travail plus académique sont également très nombreuses.

- Tout d'abord, il y a une grande quantité de problèmes qui restent ouverts, que ce soit dans la partie dédiée aux problèmes JàT ou dans la partie dédiée aux travaux interférants. Certains de ces problèmes – qui n'ont pas été abordés profondément dans le cadre de cette thèse par faute de temps – peuvent être étudiés et certainement fermés.
- D'autre part, pour les problèmes difficiles, très peu de bornes inférieures et supérieures ont été développées. Il serait tout à fait intéressant d'étudier le comportement des solveurs de programmation mathématique, d'élaborer des procédures par séparation et évaluation, des bornes inférieures performantes, etc., donc d'appliquer tous les outils de la recherche opérationnelle pour résoudre ces problèmes le plus efficacement possible. Et ceci, à la fois pour les problèmes de type JàT et pour les problèmes avec travaux interférants.
- Ensuite, le travail présenté ici s'est focalisé sur les problèmes d'ordonnancement de travaux indépendants. Il serait tout à fait pertinent de voir ce que deviennent

tous ces problèmes, à la fois les problèmes JàT et les problèmes avec travaux interférants, soit dans un contexte d'atelier de production (flowshop, jobshop, etc.), soit en considérant des contraintes de précédence entre travaux, en ajoutant des contraintes de dates de début au plus tôt, etc. Les modèles de base ont été abordés, tous les modèles plus compliqués qui en découlent peuvent aussi faire l'objet d'études fort intéressantes.

– Enfin, il serait tout à fait pertinent d'essayer de combiner les deux parties de cette thèse en étudiant des travaux interférants, dont certains doivent être exécutés JàT.

Globalement, il nous semble que l'étude des problèmes d'ordonnancement de travaux interférants généralise très bien les études sur les problèmes mono-critères et multi-critères, et il serait fort intéressant de voir ce que donnent tous les résultats de la littérature ([95, 209]) dans ce contexte particulier.

Bibliographie

[1] T.S. Abdul-Razaq, C.N. Potts, and L.N. VanWassenhove. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, 26 :235–253, 1990.

[2] G.I. Adamopoulos and C.P. Pappis. Scheduling under a common due date on parallel unrelated machines. *European Journal of Operational Research*, 105 :494–501, 1998.

[3] F. Afrati, A. Bampis, C. Kenyon, and I. Milis. A PTAS for the average weighted completion time problem on unrelated machines. *Journal of Scheduling*, 3 :323–332, 2000.

[4] F. Afrati and I. Milis. Designing PTASs for MIN-SUM scheduling problems. *Discrete Applied Mathematics*, 154 :622–639, 2006.

[5] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, and A. Pacifici. Nondominated schedules for a job-shop with two competing users. *Computational & Mathematical Organization Theory*, 6 :191–217, 2000.

[6] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, and A. Pacifici. Scheduling problems with two competing users. *Operations Research*, 52 :229–242, 2004.

[7] A. Agnetis, D. Pacciarelli, and A. Pacifici. Multi-agent single machine scheduling. *Annals of Operations Research*, 150 :3–15, 2007.

[8] M.S. Akturk and F. Erhun. An overview of design and operational issues of Kanban systems. *International Journal of Production Research*, 37 :3859–3881, 1999.

[9] B. Alidaee and S.S. Panwalkar. Single stage minimum absolute lateness problem with a common due date on non-identical machines. *Journal of the Operational Research Society*, 44(1) :29–36, 1993.

[10] M.-T. Almeida and M. Centeno. A composite heuristic for the single machine early/tardy job scheduling problem. *Computers & Operations Research*, 25(7/8) :625–635, 1998.

[11] E. Angel and E. Bampis. A multi-start dynasearch algorithm for the time dependent single-machine total weighted tardiness scheduling problem. *European Journal of Operational Research*, 162(1) :281–289, 2005.

[12] E.M. Arkin and R.O. Roundy. Weighted-tardiness scheduling on parallel machines with proportional weights. *Operations Reasearch*, 39 :64–81, 1991.

[13] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation. Combinatorial optimization problems and their approximability properties*. 2nd edition, Springer, berlin edition, 2003.

[14] G. Ausiello, P. Crescenzi, and M. Protasi. Approximate solution of $\mathcal{NP}$ optimization problems. *Theoretical Computer Science*, 150(1) :1–55, 1995.

[15] G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi. Toward a unified approach for the classification of $\mathcal{NP}$-complete optimization problems. *Theoretical Computer Science*, 12(1) :83–96, 1980.

[16] U. Bagchi, R.S. Sullivan, and Y.L. Chang. Minimizing mean absolute deviation of completion times about a common due date. *Naval Research Logistics*, 33(2) :227–240, 1986.

[17] U. Bagchi, R.S. Sullivan, and Y.L. Chang. Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties about a common due-date. *Naval Research Logistics*, 34(5) :739–751, 1987.

[18] K. Baker and J.C. Smith. A multiple-criterion model for machine scheduling. *Journal of Scheduling*, 6 :7–16, 2003.

[19] K.R. Baker and G.D. Scudder. On the assignment of optimal due dates. *Journal of the Operational Research Society*, 40(1) :93–95, 1989.

[20] K.R. Baker and G.D. Scudder. Sequencing with earliness and tardiness penalties : a review. *Operations Research*, 38 :22–36, 1990.

[21] Ph. Baptise. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103 :21–32, 2000.

[22] Ph. Baptiste, M. Flamini, and F. Sourd. Lagrangian bounds for just-in-time jobshop scheduling. *Computers & Operations Research*, 35(3) :906–915, 2008.

[23] M. Den Basten, T. Stüzle, and M. Dorigo. Ant colony optimisation for the total weighted tardiness problem. volume 1917, pages 611–620, 2000.

[24] J. Bauman and J. Jozefowska. Minimizing the earliness-tardiness costs on a single machine. *Computers & Operations Research*, 33(11) :3219–3230, 2006.

[25] U. Bilge, M. Kurtulan, and F. Kiraç. A tabu search algorithm for the single machine total weighted tardiness problem. *European Journal of Operational Research*, 176 :1423–1435, 2007.

[26] D. Biskup and M. Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Operations Research*, 28 :787–801, 2001.

[27] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Handbook on scheduling : from theory to applications*. Springer, 2007.

[28] W. Bozejko, J. Grabowski, and M. Wodecki. Block approach - tabu search algorithm for single machine total weighted tardiness problem. *Computers & Industrial Engineering*, 50 :1–14, 2006.

[29] P. Brucker. *Scheduling algorithms*. 4th edition, Springer-Verlag, Berlin, Germany, 2004.

[30] J. Bruno, Jr.E.G. Coffman, and R. Sethi. Scheduling Independent Tasks to Reduce Mean Finishing Time. *Communicaiton of the ACM*, 17 :382–387, 1974.

[31] J. Calier and Ph. Chrétienne. *Problème d'ordonnancement : modélisation, complexité, algorithmes*. Masson, Collection Etudes et Recherches en Informatique, 1988.

[32] G. Casanova. Qualité et organisation. 2000.

[33] T.C.E. Cheng. An algorithm for the CON due-date determination and sequencing problem. *Computer & Operations Research*, 14 :537–542, 1987.

[34] T.C.E. Cheng. An alternative proof of optimality for the common due-date assignment problem. *European Journal of Operational and Research*, 37 :250–253, 1989.

[35] T.C.E. Cheng. A note on a partial search algorithm for the single-machine optimal common due-date assignment and sequencing problem. *Computer & Operations Research*, 17 :321–324, 1990.

[36] T.C.E. Cheng and M.C. Gupta. Survey of scheduling research involving due date determination decisions. *European Journal of Operational and Research*, 38 :156–166, 1989.

[37] T.C.E. Cheng, C.T. Ng, and J.J. Yuan. Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical Computer Science*, 362 :273–281, 2006.

[38] T.C.E. Cheng, C.T. Ng, and J.J. Yuan. Multi-agent scheduling on a single machine with max-form criteria. *European Journal of Operational Research*, 188 :603–609, 2008.

[39] T.C.E. Cheng, C.T. Ng, J.J. Yuan, and Z.H. Liu. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165 :423–443, 2005.

[40] T.C.E Cheng and Z.-L.Chen. Parallel-machine scheduling problems with earliness and tardiness penalties. *Journal of the Operational Research Society*, 45 :685–695, 1994.

[41] Ph. Chrétienne and F. Sourd. PERT scheduling with convex cost functions. *Theoretical Computer Science*, 292(1) :145–164, 2003.

[42] R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21 :5–12, 2001.

[43] R.K. Congram, C.N. Potts, and S.L. Van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14 :52–67, 2002.

[44] H.A.J. Crauwels, C.N. Potts, and L.N. Van Wassenhove. Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 10(3) :341–350, 1998.

[45] F. Della Croce, M. Ghirardi, and R. Tadei. Recovering Beam Search : Enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10 :89–104, 2004.

[46] F. Della Croce and V. T'kindt. A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem. *Journal of Operational Research Society*, 53 :1275–1280, 2002.

[47] J.S. Davis and J.J. Kanet. Single-machine scheduling with early and tardy completion costs. *Naval Research Logistics*, 40(1) :85–101, 1993.

[48] P. De, J.B. Ghosh, and C.E. Wells. CON due-date determination and sequencing. *Computer & Operations Research*, 17 :333–344, 1990.

[49] P. De, J.B. Ghosh, and C.E. Wells. On the minimization of completion time variance with a bicriteria extension. *Operations Research*, 40(6) :1148–1155, 1992.

[50] P. De, J.B. Ghosh, and C.E. Wells. Due-date assignment and early/tardy scheduling on identical parallel machines. *Naval Research Logistics*, 41(1) :17–32, 1994.

[51] P. De, J.B. Ghosh, and C.E. Wells. Solving a generalized model for CON due date assignment and sequencing. *International Journal of Production Economics*, 34(2) :179–185, 1994.

[52] C. Del'homme. Ordonnancement de type Juste-à-temps avec date de fin commune contrôlable. Rapport de projet de Fin d'Étude sous l'encadrement de N. Huynh Tuong et A. Soukhal, École Polytechnique de l'Université François Rabelais de Tours, 2008.

[53] M. Demange and V. Paschos. Autour de nouvelles notions pour l'analyse des algorithmes d'approximation : formalisme unifié et classes d'approximation. *RAIRO - Operations Research - Recherche Opérationnelle*, 36(3) :237–277, 2002.

[54] M. Van den Akker, J.A. Hoogeveen, and S. Van de Velde. Combining column generation and lagrangean relaxation to solve a single-machine common due date problem. *INFORMS Journal on Computing*, 14(1) :37–51, 2002.

[55] B. Dickman, Y. Wilamowsky, and S. Epstein. Multiple common due dates. *Naval Research Logistics*, 48(4) :293–298, 2001.

[56] G. Diepen. *Column generation algorithms for machine scheduling and intergrated airport planning*. Thesis report, Utrecht Institute for ICT Research, Utrecht, Neitherland, 2008.

[57] P. Dileepan. Common due date scheduling problem with separate earliness and tardiness penalties. *Computers & Operations Research*, 20 :179–181, 1993.

[58] P. Dileepan and T. Sen. Bicriterion static scheduling research for a single machine. *Omega*, 16(1) :53–59, 1988.

[59] P. Dileepan and T. Sen. Bicriterion jobshop scheduling with total flowtime and sum of squared lateness. *Engineering Costs and Production Economics*, 21(3) :295–299, 1991.

[60] J. Du and J.Y.-T. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15 :483–495, 1990.

[61] H. Emmons. Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics*, 34(6) :803–810, 1987.

[62] B. Esteve. *Des problèmes d'ordonnancement multicritères de type juste-à-temps : modélisation et résolution*. Rapport de thèse, LI Tours - Université François Rabelais, Tours, France, 2005.

[63] B. Esteve, C. Aubijoux, A. Chartier, and V. T'kindt. A recovering beam search algorithm for the single machine Just-in-Time scheduling problem. *European Journal of Operational Research*, 172(3) :798–813, 2006.

[64] S. Kedad-Sidhoum F. Sourd. A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. *Journal of Scheduling*, 11(1) :49–58, 2008.

[65] Y. Fathi and H.W.L. Nuttle. Heuristics for the common due date weighted tardiness problem. *IIE Transaction*, 22 :215–225, 1990.

[66] R. Faure. Précis de recherche opérationnelle. *Dunod*, 1979.

[67] M. Feldmann and D. Biskup. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers and Industrial Engineering*, 44 :307–323, 2003.

[68] M.L. Fisher. A dual algorithm for the one machine scheduling problem. *Mathematical Programming*, 11 :229–252, 1976.

[69] T.D. Fry, R.D. Armstrong, and H. Lewis. A framework for single machine multiple objective sequencing research. *Omega*, 17(6) :595–607, 1989.

[70] T.D. Fry and G.K. Leong. Single machine scheduling : A comparison of two solution procedures. *Omega*, 15(4) :277–282, 1987.

[71] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco, 1979.

[72] M.R. Garey, R.E. Tarjan, and G.T Wilfong. One-processor scheduling with asymmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13 :330–348, 1988.

[73] E.S. Gee and C.H. Smith. Selecting allowance policies for improved job shop performance. *International Journal of Production Research*, 31(8) :1839–1852, 1993.

[74] M. Ghirardi and C.N. Potts. Makespan minimization for scheduling unrelated parallel machines : a recovering beam search approach. *European Journal of Operational Research*, 165 :457–467, 2005.

[75] F. Glover. Tabu search – part I. *ORSA Journal on Computing*, 3 :190–206, 1989.

[76] F. Glover. Tabu search – part I. *ORSA Journal on Computing*, 2 :4–32, 1990.

[77] V. Gordon, J.M. Proth, and C. Chu. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139 :1–25, 2002.

[78] V. S. Gordon, J-M Proth, and V. A. Strusevich. *Scheduling with due date assignment*, chapter 20. Handbook of scheduling : algorithms, models and performance analysis, J.Y.T Leung eds., CRC Press, Boca Raton, FL, USA (2004), 2004.

[79] Groupe GOThA. *Modèles et algorithmes en ordonnancement*. Ellipses, Sous la coordination de P. Baptiste, E. N´eron et F. Sourd, San Francisco, 2004.

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

[81] A. Grosso, F. Della Croce, and R. Tadei. An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters*, 32(1) :68–72, 2004.

[82] M. Guignard. Lagrangean relaxation. *Top*, 11(2) :151–228, 2003.

[83] Y.P. Gupta, C.R. Bector, and M.C. Gupta. Optimal schedule on a single machine using various due date determination methods. *Computers in Industry*, 15 :245–253, 1990.

[84] N. Hall. Scheduling problems with generalized due dates. *IIE Transaction*, 18 :220–222, 1986.

[85] N. Hall. Single and multiple processor models for minimizing completion time variance. *Naval Research Logistics*, 33 :49–54, 1986.

[86] N.G. Hall, W. Kubiak, and S.P. Sethi. Earliness-tardiness scheduling problem, II : Deviation of completion times about a restrictive common due date. *Operations Research*, 39(5) :847–856, 1991.

[87] N.G. Hall and M.E. Posner. Earliness-tardiness scheduling problem, I : Weighted deviation of completion times about a common due date. *Operations Research*, 39(5) :835–846, 1991.

[88] N.G. Hall, S.P. Sethi, and C. Sriskandarajah. On the complexity of generalized due date scheduling problems. *European Journal of Operational Research*, 51 :100–109, 1991.

[89] Q. Hao, Z. Yang, D. Wang, and Z. Li. Common due date determination and sequencing using tabu search. *Computers & Operations Research*, 23 :409–417, 1996.

[90] Y. Hendel. *Contribution à l'ordonnancement juste-à-temps*. Rapport de thèse, LIP6 - Université Pierre et Marie Curie, Paris 6, Paris, 2005.

[91] Y. Hendel and F. Sourd. Efficient neighborhood search for the one-machine earliness-tardiness scheduling problem. *European Journal of Operational Research*, 173(1) :108–119, 2006.

[92] Y. Hendel and F. Sourd. An improved earliness-tardiness timing algorithm. *Computers & Operations Research*, 34(10) :2931–2938, 2007.

[93] C.M. Hino, D.P. Ronconi, and A.B. Mendes. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, 160 :190–201, 2005.

[94] H. Hoogeveen. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2 :225–231, 1973.

[95] H. Hoogeveen. Multicriteria scheduling. *European Journal of Operational Research*, 167(3) :592–623, 2005.

[96] J.A. Hoogeveen. Single machine scheduling to minimize a function of k maximum cost criteria. *Dans J.A. Hoogeven, Single-machine bicriteria scheduling, Thèse de doctorat, CWI Amsterdam (Pays Bas)*, pages 67–77, 1992.

[97] J.A. Hoogeveen. Minimizing maximum promptness and maximum lateness on a single machine. *Mathematics of Operations Research*, 21(1) :100–114, 1996.

[98] J.A. Hoogeveen and S.L. Van de Velde. Scheduling around a small common due date. *European Journal of Operational Research*, 55(2) :237–242, 1991.

[99] J.A. Hoogeveen and S.L. Van de Velde. Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. *Operations Research Letters*, 17 :205–208, 1995.

[100] J.A. Hoogeveen and S.L. Van de Velde. A branch-and-bound algorithm for single machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing*, 8(4) :402–412, 1996.

[101] J.A. Hoogeveen and S.L. Van de Velde. Earliness-tardiness scheduling around almost equal due dates. *INFORMS Journal on Computing*, 9(1) :92–99, 1997.

[102] J.A. Hoogeveen, J.K. Lenstra, and S.L. Van de Velde. Sequencing and scheduling : an annotated bibliography. Memorandum COSOR 97-02, Eindhoven University of Technology, 1997.

[103] J.A. Hoogeveen, H. Oosterhout, and S.L. Van de Velde. New lower and upper bounds for scheduling around a small common due date. *Operations research*, 42(1) :102–110, 1994.

[104] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2) :317–327, 1976.

[105] C.-C. Huang and A. Kusiak. Overview of Kanban systems. *International Journal of Computer Integrated Manufacturing*, 9 :169–189, 1996.

[106] N. Huynh-Tuong and A. Soukhal. A PTAS for just-in-time scheduling around a common due date. *Theoretical Computer Science, submitted in May*, 2008.

[107] N. Huynh-Tuong and A. Soukhal. Due date assignment and just-in-time scheduling on uniform parallel machines. In *The 21st Conference of the European Chapter on Combinatorial Optimization (ECCO XXI'08)*, Dubrovnik (Croatia), May 29-31 2008.

[108] N. Huynh-Tuong and A. Soukhal. Due dates assignment and JIT scheduling with equal-size jobs. *European Journal of Operational Research, submitted in August 2008*, 2008.

[109] N. Huynh-Tuong and A. Soukhal. Ordonnancement juste-à-temps sur une seule machine avec date de fin souhaitée commune. In *9ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'08)*, pages 146–148, Clermont-Ferrand (France), Febuary 25-27 2008.

[110] N. Huynh-Tuong and A. Soukhal. Polynomial cases and PTAS for just-in-time scheduling on parallel machines around a common due date. In *11th International Workshop on Project Management and Scheduling (PMS'08)*, pages 152–155, Istanbul (Turkey), April 28-30 2008.

[111] N. Huynh-Tuong and A. Soukhal. Some new polynomial cases in just-in-time scheduling problems with multiple due dates. In *6th International conference on Research, Innovation & Vision for the Future in Computing & Communications Tech-*

nologies (RIVF'08), ISBN 978-1-4244-2379-8, pages 36–41, Ho Chi Minh (Vietnam), Juillet 13-17 2008.

[112] N. Huynh-Tuong and A. Soukhal. Interfering job set scheduling on three stage flowshop with a common stage machine. Rapport technique, Université de François Rabelais de Tours, France, 2009.

[113] N. Huynh-Tuong and A. Soukhal. Interfering job set scheduling on two-operation three-machine flowshop. In *The 2009 IEEE - RIVF International Conference on Computing and Communication Technologies (RIVF'09)*, Da Nang (Vietnam), Juillet 13-17 2009.

[114] N. Huynh-Tuong and A. Soukhal. Ordonnancement des travaux interférants sur machines parallèles. In *10ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'09), ISBN 2-905267-64-X*, pages 49–61, Nancy (France), February 10-12 2009.

[115] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. Single machine and parallel machine scheduling problems with a common due date to minimize total weighted tardiness. In *3rd Multidisciplinary International Conference on Scheduling : Theory and Application (MISTA'07)*, pages 498–505, Paris (France), August 28-31 2007.

[116] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. Uniform parallel machine scheduling problem with a common due date to minimize total weighted tardiness. In *8th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP'07)*, pages 146–148, Istanbul (Turkey), Juillet 02-06 2007.

[117] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. A new dynamic programming formulation for scheduling independent tasks with common due date on parallel machines. *European Journal of Operational Research, submitted in August*, 2008.

[118] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. Exact algorithms for scheduling interfering independent jobs. Rapport technique, Université de François Rabelais de Tours, France, 2008.

[119] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. Ordonnancement des travaux interférant dans un flowshop à deux machines. In *7ème Conférence Internationale de Modélisation et Simulation (MOSIM'08), ISBN 978-2-7430-1057-7*, pages 582–591, Paris (France), March 31 - April 02 2008.

[120] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. Complexity of Partition problem with distinct elements. Rapport technique, Université de François Rabelais de Tours, France, 2009.

[121] N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. New scheduling problems with interfering and independent jobs. *Operations Research, submitted*, 2009.

[122] N. Huynh-Tuong, A. Soukhal, C. Del'homme, and D. Shao. Due date assignment and JIT scheduling on single machine. In *Modelling, Computation and Optimization in Information Systems and Management Sciences (MCO'08)*, Metz (France-Luxembourg), Septembre 08-10 2008.

[123] N. Huynh-Tuong, A. Soukhal, and L. Miscopein. Interfering job set scheduling on three-stage flowshop with a common stage machine. In *4th Multidisciplinary International Conference on Scheduling : Theory and Application (MISTA'09)*, Dublin (Ireland), Août 10-12 2008.

[124] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4) :463–468, 1975.

[125] R.J.W. James. Using tabu search to solve the common due date early/tardy machine scheduling problem. *Computers & Operations Research*, 24(3) :199–208, 1997.

[126] J. Józefowska. *Just-in-time scheduling : models and algorithms for computer and manufacturing systems*. Springer, 2007.

[127] B. Jurisch, W. Kubiak, and J. Jozefowska. Algorithms for minclique scheduling problems. *Discrete Applied Mathematics*, 72 :115–139, 1997.

[128] Ph. Kaminsky and D. Hochbaum. *Due date quotation models and algorithms*, chapter 19. Handbook of scheduling : algorithms, models and performance analysis, J.Y.T Leung eds., CRC Press, Boca Raton, FL, USA (2004), 2004.

[129] J.J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics*, 28 :643–651, 1981.

[130] V. Kann. Polynomially bounded minimization problems that are hard to approximate. *Nordic Journal of Computing*, 1(3) :317–331, 1994.

[131] M.-Y. Kao, T.-W. Lam, W.-K. Sung, and H.-F. Ting. A decomposition theorem for maximum weight bipartite matchings. *SIAM Journal on Computing*, 31 :18–26, 2002.

[132] S. Kedad-Sidhoum, Y.A. Rios-Solis, and F. Sourd. Lower bound for the earliness-tardiness scheduling problem on parallel machines with distinct due dates. *European Journal of Operational Research*, 189(3) :1305–1316, 2008.

[133] H. Kellerer and Vitaly A. Strusevich. A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. *Theoretical Computer Science*, 369 :230–238, 2006.

[134] Y.-D. Kim and C.A. Yano. Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. *Naval Research Logistics*, 41(7) :913–933, 1994.

[135] S.G. Kolliopoulos and G. Steiner. Approximation algorithms for minimizing total weighted tardiness on a single machine. *Theoretical Computer Science*, 355 :261–273, 2006.

[136] S.G. Kolliopoulos and G. Steiner. Approximation algorithms for scheduling problems with a modified total weighted tardiness objective. *Operations Research Letters*, 35 :685–692, 2007.

[137] C. Koulamas. A faster fully polynomial approximation scheme for the singe-machine total tardiness problem. *European Journal of Operational Research*, 193 :637–638, 2009.

[138] M.Y. Kovalyov. Improving the complexities of approximation algorithms for optimization problems. *Operations Research Letters*, 17 :85–87, 1995.

[139] M.Y. Kovalyov. Soft and negotiable release and due dates in supply chain scheduling. 2006.

[140] M.Y. Kovalyov and W. Kubiak. A fully polynomial approximation scheme for the weighted earliness-tardiness problem. *Operations Research*, 47 :757–761, 1999.

[141] M.Y. Kovalyov and F. Werner. Approximation schemes for scheduling jobs with common due date on parallel machines to minimize total tardiness. *Journal of Heuristics*, 8 :415–428, 2002.

[142] W. Kubiak, S. Lou, and S. Sethi. Equivalence of mean flow time problems and mean absolute deviation problems. *Operations Research Letters*, 9 :371–374, 1990.

[143] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2 :83–97, 1955.

[144] S. Lakshminarayan, R. Lakshmanan, R. Papinou, and R Rochette. Optimum single machine scheduling with earliness and tardiness penalties. *Mathematical and Computer Modelling*, 26(6) :1079–1082, 1978.

[145] V. Lauff and F. Werner. Scheduling with common due date, earliness and tardiness penalties for multi-machine problems : a survey. *Mathematical and Computer Modelling*, 40 :637–655, 2004.

[146] E.L. Lawler. A 'pseudopolynomial algorithm' for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1 :331–342, 1977.

[147] E.L. Lawler. Efficient implementation of dynamic programming algorithms for sequencing problems. 1979.

[148] E.L. Lawler. A fully polynomial approximation scheme for the total tardiness problem. *Operations Research Letters*, 1 :207–208, 1982.

[149] E.L. Lawler. Scheduling a single machine to minimize the number of late jobs. Technical report, University of California at Bekerley, 1983.

[150] E.L. Lawler and J.M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16 :77–84, 1969.

[151] S.R. Lawrence. Estimating flowtimes and setting due-dates in complex production. *IIE Transaction*, 27 :657–668, 1995.

[152] C.-Y. Lee and S.J. Kim. Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights. *Computers and Industrial Engineering*, 28 :231–248, 1995.

[153] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26 :22–35, 1978.

[154] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1 :343–362, 1977.

[155] J.Y-T. Leung. *Handbook of scheduling : algorithms, models, and performance analysis*. Computer and information science series, Chapman and Hall/CRC (ed.), Boca Raton, Florida, 2004.

[156] C.-L. Li and T.C.E. Cheng. The parallel machine min-max weighted absolute lateness scheduling problem. *Naval Research Logistics*, 41(1) :33–46, 1994.

[157] C.-F. Liaw. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers & Operations Research*, 26(7) :679–693, 1999.

[158] C. Marchwinski and J. Shook. *Lexique Lean : Un glossaire illustré à l'intention des adeptes de la Pensée Lean*. 3ème édition, Institut Lean France, France, 2006.

[159] H. Matsuo, S.W. Otto, and R.S. Sullivan. A controlled search simulated annealing method for the single machine weighted tardiness problem. *Annals of Operations Research*, 21(1) :85–108, 1987.

[160] C. Mocquillon, C. Lenté, and V. T'Kindt. Solution of a multicriteria shampoo production problem. *IEEE International Conference on Service Systems and Service Management (IEEE-SSSM'06)*, pages 907–911, 2006.

[161] Y. Monden. *Toyota production systems*. Industrial Engineering and Management, Norcross, GA., 1983.

[162] J.M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15 :102–109, 1968.

[163] T.E. Morton, R.M. Rachamadugu, and A. Vepsalainen. Accurate myoptic heuristic for tardiness scheduling. 1984.

[164] G. Mosheiov and A. Sarig. Due-date assignment on uniform machines. *European Journal of Operations Research*, 191(1) :49–58, 2009.

[165] G. Mosheiov and U. Yovel. Minimizing weighted earliness-tardiness and due date cost with unit processing-time jobs. *European Journal of Operations Research*, 172 :528–544, 2006.

[166] A. Nagar, J. Haddock, and S.S. Heragu. Multiple and bicriteria scheduling : A literature survey. *European Journal of Operational Research*, 81 :88–104, 1995.

[167] A.C. Nearchou. A differential evolution approach for the common due date early/tardy job scheduling problem. *Computers & Operations Research*, 35(4) :1329–1343, 2008.

[168] C.T. Ng, T.C.E. Cheng, and J.J. Yuan. A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization*, 12 :387–394, 2006.

[169] S.S. Panwalkar, M.L. Smith, and A. Seidmann. Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research*, 30 :391–399, 1982.

[170] V.Th. Paschos. Tutorial sur l'approximation polynomiale. Présentation de Journée Franciliennes de Recherche Opérationnelle, 2003.

[171] V.Th. Paschos. *Complexité et approximation polynomale*. Hermès - Lavoisier, 2004.

[172] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science*, 15(3) :251–277, 19815.

[173] S.O. Peng and T.E. Morton. Filtered beam search in scheduling. *International Journal of Production Research*, 26(1) :35–62, 1988.

[174] C. Pessan, J-L. Bouquard, and E. Neron. An unrelated parallel machines model for an industrial production resetting problem. *European Journal of Industrial Engineering*, 2(2) :153–171, 2008.

[175] C. Pessan, J-L. Bouquard, and E. Neron. Genetic branch-and-bound or exact genetic algorithm ? *Lecture Notes in Computer Science, Springer Berlin/Heidelberg*, 4926 :136–147, 2008.

[176] P.R. Philipoom, L. Wiegmann, and L.P. Rees. Cost-based due-date assignment with the use of classical and neural-network approaches. *Naval research logistics*, 44(1) :21–46, 1997.

[177] M. Pinedo. *Scheduling : theory, algorithms, and systems*. 2nd edition, Precentice Hall, Upper Saddle River, New Jork, USA, 2002.

[178] C.N. Potts and L.N. Van Wassenhove. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, 33(2) :363–377, 1985.

[179] C.N. Potts and L.N. Van Wassenhove. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, 33 :363–377, 1985.

[180] C.N. Potts and L.N. Van Wassenhove. Single machine tardiness sequencing heuristics. *IIE Transactions*, 23 :346–354, 1991.

[181] M.A. Quaddus. A generalized model of Optimal due-date assignment by linear. *Journal of the Operational Research Society*, 38(4) :353–359, 1987.

[182] G. Rabadi, M. Mollaghasemi, and G.C. Anagnostopoulos. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-

date and sequence-dependent setup time. *Computers & Operations Research*, 31(10) :1727–1751, 2004.

[183] R. Ramasesh. Dynamic job shop scheduling : a survey of simulation research. *Omega*, 18 :43–57, 1990.

[184] Y.A. Rios-Solis. *Ordonnancement avance-retard sur machines parallèles*. Rapport de thèse, LIP6 - Université Pierre et Marie Curie, Paris 6, Paris, 2007.

[185] D.B. Roman and A.G. Dell Valle. Dynamic assignation of due-dates in an assembly shop based in simulation. *International Journal of Production Research*, 34(6) :1539–1554, 1996.

[186] Y. Rossier. *Systèmes de gestion industrielle : Une étude cas*. Presses Polytechniques et Universitaires Romandes (PPUR), ISBN 978-2880742270, 1991.

[187] M.H. Rothkopf. Scheduling independant tasks on parallel processors. *Management Science*, 12 :438–447, 1966.

[188] B. Roy. *Méthodologie multicritère d'aide à la décision*. Economica edition, 1985.

[189] N. Runge. *Problème d'ordonnancement d'avance-retard avec ou sans préemption*. Rapport de thèse, LIP6 - Université Pierre et Marie Curie, Paris 6, Paris, 2008.

[190] S.K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1) :116–127, 1976.

[191] A. Seidmann, S.S. Panwalkar, and M.L. Smith. Optimal assignment of due dates for a single processor scheduling problem. *International Journal of Production Research*, 19 :393–399, 1981.

[192] T. Sen, J.M. Sulek, and P. Dileepan. Static scheduling research to minimize weighted and unweighted tardiness : a state-of-the-art survey. *International Journal of Production Economics*, 83 :1–12, 2003.

[193] J. Sidney. Optimum single-machine scheduling with earliness and tardiness penalties. *Operations Research*, 25(1) :62–69, 1977.

[194] C.H. Smith, E.D. Minor, and H.J. Wen. Regression-based due date assignment rules for improved assembly shop performance. *International Journal of Production Research*, 33 :2375–2385, 1995.

[195] W.E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2) :59–66, 1956.

[196] A. Soukhal, N. Huynh-Tuong, and L. Miscopein. Deux agents concurents pour l'ordonnancement des travaux dans un flowshop avec étage commun. In *10ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'09), ISBN 2-905267-65-8*, pages 125–127, Nancy (France), February 10-12 2009.

[197] A. Soukhal, N. Huynh Tuong, and Z. Dao. Méthodes exactes et approchées pour l'ordonnancement des travaux interférant. In *International Symposium On Operational Research (ISOR'08), ISBN 1112-7104*, pages 269–277, Alger (Algeria), 11 2008.

[198] A. Soukhal, N. Huynh Tuong, and Z. Dao. Parallel machine scheduling with interfering jobs. In *8th International Conference on Multiple Objective and Goal Programming (MOPGP'08))*, Portsmouth (UK), 10 2008.

[199] F. Sourd. Ordonnancer juste-à-temps. *Bulletin de la société française de recherche opérationnelle et d'aide à la décision*, 12, 2004.

[200] F. Sourd. Punctuality and idleness in just-in-time scheduling. *European Journal of Operational Research*, 16(3) :739–751, 2005.

[201] F. Sourd. Dynasearch for the earliness-tardiness scheduling problem with release dates and setup constraints. *Operations Research Letters*, 34(5) :591–598, 2006.

[202] F. Sourd. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal of Computing*, 2009.

[203] F. Sourd and S. Kedad-Sidhoum. The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, 6(6) :533–549, 2003.

[204] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal Global Optimization*, 11 :341–354, 1997.

[205] H. Sun and G. Wang. Parallel machine earliness and tardiness scheduling with proportional weights. *Computers & Operations Research*, 30(5) :801–808, 2003.

[206] W. Szwarc and S.K. Mukhopadhyay. Minimizing a quadratic cost function of waiting times in single-machine scheduling. *Journal of the Operational Research Society*, 46 :753–761, 1995.

[207] S. Tanaka and S.Fujikuma. An efficient exact algorithm for general single-machine scheduling with machine idle time. In *IEEE International Conference on*

Automation Science and Engineering (CASE 2008), ISBN 978-1-4244-2022-3, pages 371–376, Washington DC (USA), August 23-26 2008.

[208] V. T'kindt and J-C. Billaut. *L'ordonnancement multicritère*. Presses de l'Université de Tours, 2000.

[209] V. T'kindt and J-C. Billaut. *Multicriteria scheduling : theory, models and algorithms*. 2nd edition, Springer, 2006.

[210] C.-H. Tsai, G.-T. Chang, and R.-K. Li. Integrating order release control with due-date assignment rules. *International journal of production research*, 35(12) :3379–3392, 1997.

[211] J.M.S. Valente and R.A.F.S. Alves. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. *Computers & Industrial Engineering*, 48 :363–375, 2005.

[212] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, Heidelberg, 2003.

[213] M.M. Vig and K.J. Dooley. Dynamic rules for due date assignments. *International Journal of Production Research*, 29(7) :1361–1377, 1991.

[214] M.M. Vig and K.J. Dooley. Mixing static and dynamic flowtime estimates for due date assignment. *Journal of Operations Management*, 11(1) :67–79, 1993.

[215] B.P.-C. Yen Wan. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142(2) :271–281, 2002.

[216] S. Webster. The complexity of scheduling job families about a common due date. *Operations Research Letters*, 20(2) :65–74, 1997.

[217] S. Webster, P.D. Jog, and A. Gupta. A genetic algorithm for scheduling job families on a single machine with arbitrary earliness/tardiness penalties and an unrestricted common due date. *International Journal of Production Research*, 43(4) :2543–2552, 1998.

[218] G.J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS) ? *INFORMS Journal on Computing*, 12 :57–74, 2000.

[219] L.A. Wolsey and G.L. Nemhauser. *Integer and combinatorial optimization*. Wiley-Interscience, United States of America, 1999.

[220] J. Womack and D. Jones. *Le Lean au service du client : Ce que le client veut - Quand il veut - Où il veut*. Vuibert, France, 2006.

[221] W.-Q. Xiao and C.-L. Li. Approximation algorithms for common due date assignment and job scheduling on parallel machines. *IIE Transactions*, 34(5) :467–477, 2002.

[222] F. Sourd Y.A. Rios-Solis. Exponential neighborhood search for a parallel machine scheduling problem. *Computers & Operations Research*, 35(5) :1697–1712, 2008.

[223] C.A. Yano and Y.-D. Kim. Algorithms for a class of single-machine weighted tardiness and earliness problems. *European Journal of Operational Research*, 52(2) :167–178, 1991.

[224] K.-C. Ying. Minimizing earliness–tardiness penalties for common due date single–machine scheduling problems by a recovering beam search algorithm. *Computers & Industrial Engineering*, 55(2) :494–502, 2008.

[225] S. Yoshida. Le lean et la surproduction. *LEAN Training Newsletter*, 8 :1–2, 2009.

[226] J. Yuan. The $\mathcal{NP}$-hardness of the single machine common due date weighted tardiness problem. *Systems Science and Mathematical Sciences*, 5 :328–333, 1992.

[227] J.J. Yuan, W.P. Shang, and Q. Feng. A note on the scheduling with two families of jobs. *Journal of Scheduling*, 8 :537–542, 2005.

[228] S.H. Zegordi, K. Itoh, and T. Enkawa. A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem. *International Journal of Production Research*, 33(5) :1449–1466, 1995.

Index

- Nguyen HUYNH TUONG -

COMPLEXITÉ ET ALGORITHMES POUR L'ORDONNANCEMENT MULTICRITERE DE TRAVAUX INDÉPENDANTS : PROBLÈMES JUSTE-À-TEMPS ET TRAVAUX INTERFÉRANTS

Résumé

Nous abordons dans cette thèse deux types de problèmes d'ordonnancement sur une machine ou sur des machines parallèles :

1. les problèmes d'ordonnancement de type juste-à-temps : il s'agit de déterminer un ordonnancement de sorte que les travaux se terminent le plus près possible de leur date de fin souhaitée. On considère le cas où la date de fin souhaitée commune est connue et le cas où elle est à déterminer. De nouveaux algorithmes exacts sont proposés. Des schémas d'approximation sont élaborés.
2. les problèmes d'ordonnancements de travaux interférants : il s'agit de déterminer un ordonnancement qui permet d'optimiser un critère pour tous les travaux, sachant que la solution trouvée doit permettre également l'optimisation d'un autre critère défini uniquement sur un sous-ensemble des travaux. Il s'agit ici d'un nouveau problème d'ordonnancement multicritère, différent de la notion classique. Les approches considérées pour trouver une solution non dominée sont l'approche ε-contrainte, la combinaison linéaire de critères et le *goal programming*. De nouveaux résultats de complexité sont montrés et des algorithmes exacts sont développés.

Abstract

In this thesis, we consider two kinds of scheduling problems on a single machine or on parallel machines :

1. just-in-time scheduling problems : it aims to determine a schedule so that a job completes as close as possible to its due date. We consider the case where the common due date is known and the case where the common due date has to be fixed. New exact algorithms based on greedy algorithms and dynamic programming are proposed. Approximation schemes are given.
2. scheduling problems with interfering jobs : the aim is to determine a schedule that optimizes a criterion for the whole set of jobs and so that the solution optimizes another objective only for a subset of jobs. It is here a new multi-criteria scheduling problem, different from the classical notion. The approaches considered for finding a non-dominated solution are the ε-constraint approach, the linear combination of criteria and the goal programming approach. New complexity results are proposed and exact algorithms are developed.