École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5e année
2012 - 2013

Rapport de projet de fin d'études

Prétraitement pour le problème du $1|r_i|L_{max}$

Encadrant
Vincent T'KINDT
tkindt@univ-tours.fr

Étudiant
Thomas NOGUER
thomas.noguer@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2012 - 2013

Version du 9 mai 2013

Table des matières

Table des figures

Liste des tableaux

Introduction

Ce rapport présente le projet de fin d'études de Thomas Noguer étudiant en dernière année d'école d'ingénieur informatique à Polytech Tours. Il s'agit d'un projet qui se situe sur toute la durée de dernière année d'études et constitue le projet de plus grande envergure lors de cette formation. Ce projet de fin d'études est proposé par Vincent T'kindt enseignant chercheur à Polytech Tours et est directement lié avec ses travaux de chercheur. Il s'agit donc évidemment d'un projet orienté recherche ce qui influence fortement son déroulement et la démarche de gestion de projet à adopter. Le sujet est le prétraitement pour le problème du $1|r_i|L_{max}$. Il s'agit d'un problème d'ordonnancement de travaux sur une machine. La méthode du prétraitement permet de connaître la valeur de variables du modèle mathématique de notre problème avant sa résolution. L'approche par modèle mathématique permet également l'ajout de coupes afin de diminuer l'espace des solutions. Le projet consiste globalement en la prise en main de l'existant, son amélioration, l'ajout de coupes et la réalisation de campagnes de tests afin d'en tirer des résultats exploitables.

Ce rapport se découpe en plusieurs grandes parties, la première est la présentation de l'environnement du sujet, de ses contraintes et objectifs. La deuxième présente les différentes parties de l'existant et comment ils interviennent dans le projet. La troisième partie présente la phase de développement en elle-même et regroupe les grandes avancées effectuées au cours du projet. La quatrième partie présente les résultats expérimentaux obtenus à la suite de ce projet et leurs différentes interprétations. Les deux dernières parties présentent la partie gestion de projet et les perspectives d'améliorations possibles du projet.

Présentation du sujet

2.1 Contexte

Ce projet se situe à la fin du projet de fin d'études et du stage de master de Zinan Liu. L'objectif de ces précédents projets était de mettre en place une librairie de prétraitement la plus générique possible afin de permettre sa réutilisation sur d'autres problèmes. Cette librairie a été développée et testée pour le problème du $1|r_i|L_{max}$. Ce projet fait donc directement suite à ce travail précédent.

Le modèle utilisé a été conçu par Briand et al. [1] et a été réutilisé pour ce projet. Ce modèle a fait l'objet d'un papier présent en annexe A.1 et a donné des résultats prometteurs, mais pas suffisants pour battre la meilleure méthode de résolution pour ce problème.

La méthode du prétraitement a déjà été utilisée sur d'autres problèmes d'ordonnancement par Vincent Tkindt et avait permis d'obtenir des résultats intéressants. C'est une méthode qui a donc déjà fait ses preuves sur d'autres problèmes et qui peut s'avérer intéressante à exploiter plus en profondeur.

2.2 Contraintes

Ce projet est un problème d'optimisation. Il est donc important de chercher à optimiser le temps de calcul et la mémoire utilisée. Si l'application utilise une technique d'optimisation efficace et que les contraintes de temps et de mémoire sont respectées, nous aurons alors une vision fidèle des capacités de la méthode utilisée.

De plus, ce projet reprend l'API du prétraitement créée par Zinan Liu précédemment. Lorsque des modifications sont effectuées sur cette partie du code, il est indispensable de rendre ce code générique et documenté afin de pouvoir réutiliser l'API sur d'autres projets et problèmes.

Enfin, Vincent T'kindt a proposé de présenter les résultats de ce projet à la conférence ROADEF. Cette conférence est le rendez-vous annuel de la communauté de recherche opérationnelle et d'aide à la décision. Afin de pouvoir présenter plus que de la théorie au moment de la conférence il était donc nécessaire d'avoir des résultats à ce moment-là. Ce projet était donc soumis à une contrainte de temps supplémentaire. Le papier soumis à la conférence ROADEF 2013 est présent en annexe A.3.

2.3 Objectifs

L'objectif premier de ce projet est de montrer l'intérêt de la méthode du prétraitement pour le problème du $1|r_i|L_{max}$. Si cette méthode peut avoir un intérêt sur ce problème, nous pouvons alors imaginer qu'elle sera aussi intéressante à appliquer sur d'autres problèmes d'ordonnancement similaires.

Le second objectif est de chercher à obtenir de meilleurs résultats que ceux obtenus par Briand et al. [1] dans le papier présentant le modèle mathématique qui a été réutilisé pour ce projet.

Dans un second temps, si les résultats du prétraitement sont meilleurs que ceux du modèle seul, il est intéressant de comparer les résultats à ceux obtenus par la meilleure méthode mise en place sur ce problème : la PSE (procédure par séparation évaluation) de Carlier [3].

Présentation de l'existant

3.1 $1|r_i|L_{max}$

Le problème du $1|r_i|L_{max}$ est un problème classique d'ordonnancement. Il y a eu beaucoup de recherches à son sujet, il est par conséquent très intéressant d'avoir une possibilité d'avancée pour ce problème.

Pour ce problème, nous considérons n travaux et une machine m sans préemption autorisée. Chaque travail j possède une date de réveil r_j, une date de fin au plus tard d_j et une durée opératoire p_j. Nous définissons aussi C_j la date de fin effective du travail j ainsi que son retard $L_j = d_j - C_j$. L'objectif est de minimiser le retard maximum des travaux défini par la relation suivante $L_{max} = \max_{1\leq i\leq n}(C_i - d_i) = \max_{1\leq i\leq n}(L_j)$.

Ce problème est NP-difficile au sens fort ce qui justifie l'utilisation de l'outil informatique pour le résoudre à l'optimum.

Il est aussi intéressant de considérer le fait que ce problème possède diverses déclinaisons, qu'il y a un lien entre chaque qui permet de réutiliser les résultats présentés ici pour les problèmes $1|r_i, d_i\ |-$ ou du $1|r_i|max(C_i, q_i)$.

3.2 Modèle mathématique en pyramides

Nous considérons 7 travaux à séquencer sur une seule machine. Il y a $7! = 5040$ possibilités. Avec une complexité factorielle, il est très compliqué de mettre en place des méthodes de résolutions qui prouvent l'optimum et qui sont efficaces sur des instances avec beaucoup de travaux.

Afin de réduire le nombre de séquences possibles, il est avisé de rechercher des séquences dominantes. Le modèle de programmation mathématique présenté par Briand et al. [1] (voir annexe A.1) démontre l'existence de conditions de dominance entre les séquences qui permet de réduire le nombre de séquences à explorer à 2^n et donc 128 pour une séquence à 7 travaux au lieu des 5040 précédents.

Ce modèle repose sur la notion de pyramides et de travail-sommet ou sommet. Un travail t est sommet s'il ne contient aucun autre travail dans sa fenêtre d'exécution $[r_t; d_t]$. La figure 3.1 présente schématiquement une pyramide constituée de son sommet et de plusieurs travaux.

L'objectif central du modèle est de déterminer la pyramide auquel appartient un travail et si il est situé du côté gauche ou droit de la pyramide. S'il est situé du côté gauche, nous savons que ce travail sera ordonnancé avant le sommet. S'il est situé du côté droit, nous savons que ce travail sera ordonnancé après le sommet. En construisant les pyramides selon le modèle de Cyril nous cherchons à déterminer une séquence de travaux. La construction en pyramide permet de réduire le nombre de séquences possibles tout en prouvant l'optimum.

Dans le papier présentant le modèle mathématique, une démonstration est faite prouvant que la solution optimale du modèle permet de minimiser le retard maximum. Dans les grandes lignes, les sommets des pyramides sont les travaux qui donnent le L_{max}. Le modèle tend à minimiser le retard des sommets des pyramides et donc le retard maximum de tous les travaux. Pour plus de détails, voir le rappel du modèle en 3.2, le papier de Briand et al. est également présent en annexe A.1.

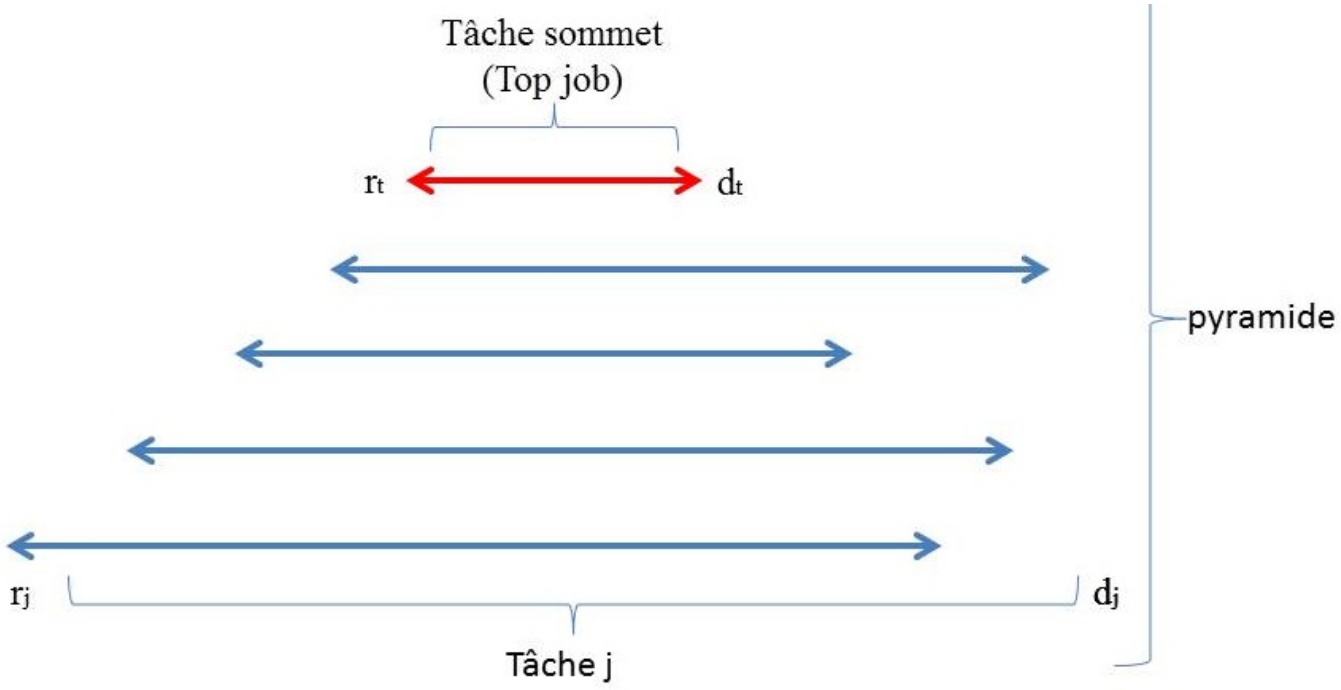


FIGURE 3.1 – Schéma d'une pyramide

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

$$\text{s.t.} \begin{cases}
R_k \geq r_{t_k} & , \; \forall k \in [1\ m] & (5.1)\\
D_k \leq d_{t_k} & , \; \forall k \in [1\ m] & (5.2)\\
R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \; \forall i | k = u(i) & (5.3)\\
D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \; \forall i | k = v(i) & (5.4)\\
R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j \in P_k\}} p_j x^+_{kj} & , \; \forall k \in [2\ m] & (5.5)\\
D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j \in P_k\}} p_j x^-_{kj} & , \; \forall k \in [1\ (m-1)] & (5.6)\\
\sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) = 1 & , \; \forall i & (5.7)\\
x^-_{ki}\ ,\ x^+_{ki} \in \{0,1\} & , \; \forall k \in [1\ m], \forall i \in P_k & \\
D_k\ ,\ R_k \in \mathbb{Z} & , \; \forall k \in [1\ m] &
\end{cases}$$

3.3 Prétraitement

La méthode phare utilisée dans ce projet est la méthode du prétraitement. Nous considérons un modèle mathématique (MIP : *Mixed Interger Programming*) qui contient des variables booléennes $\in \{0, 1\}$. Nous relâchons la contrainte d'intégrité du domaine de ces variables booléennes. En d'autres mots, les variables

booléennes $\in [0;1]$, il est possible d'avoir des valeurs non entières pour ces variables. Ce modèle relâché devient alors un modèle de programmation linéaire (LP : *Linear Programming*).

Prenons l'exemple du modèle 3.2, les variables booléennes sont les x_{kj}^- et x_{kj}^+ qui déterminent l'appartenance à un travail j du côté gauche ou droit de la pyramide k (Si $x_{kj}^- = 1 \Rightarrow$ côté droit, si $x_{kj}^+ = 1 \Rightarrow$ côté gauche). Lorsque nous construisons le modèle LP en relâchant les contraintes d'appartenance au domaine des variables x_{kj}^- et x_{kj}^+, nous acceptons le fait qu'un travail puisse, par exemple, appartenir à 30% au côté droit d'une pyramide et à 70% au côté droit d'une autre pyramide. Ce nouveau modèle LP est différent de son équivalent MIP et n'est pas représentatif du problème du $1|r_i|L_{max}$, mais nous permet d'effectuer du prétraitement.

Le modèle LP précédemment construit est résoluble avec l'algorithme du simplexe. Cette relaxation continue est très rapide à résoudre. Soit z_{LP}^* la solution optimale du LP, B^* la base de variables associée et z_{MIP}^* la solution optimale du MIP. La technique du prétraitement va permettre de déduire la valeur de variables booléennes, pour cela nous partons de la relation entre un MIP et son LP associé suivante :

$$z_{MIP}^* = z_{LP}^* + \sum_{i \notin B^*} c_i x_i$$

avec c_i le coût réduit associé à la variable x_i (x_i est une notation générale d'une variable booléenne d'un modèle quelconque, il ne s'agit pas de variables du modèle qui nous intéresse). Les coûts réduits sont des composantes de l'algorithme du simplexe. Un coût réduit correspond grossièrement à l'augmentation de la solution optimale lorsque la variable est à 1. Nous définissons UB la borne supérieure (*Upper Bound*) calculée par l'algorithme de Schrage dans notre cas. De la relation précédente, nous pouvons déduire la relation suivante :

$$UB \geq z_{LP}^* + \sum_{i \notin B^*} c_i x_i$$

$$\Leftrightarrow \sum_{i \notin B^*} c_i x_i \leq UB - z_{LP}^*$$

De cette dernière inégalité nous pouvons en déduire une règle de fixation suivante : $\forall x_i \notin B^*$,si $c_i > UB - z_{LP}^*$ alors dans toute solution optimale du MIP, $x_i = 0$. En utilisant le raisonnement similaire sur les variables d'écarts s_i (*slack variable*, $s_i = \overline{x_i}$), il est possible de définir la règle de fixation suivante : $\forall s_i \notin B^*$,si $c_i > UB - z_{LP}^*$ alors dans toute solution optimale du MIP, $x_i = 1$.

Les deux règles de fixation précédentes permettent de fixer uniquement les variables hors base du modèle. Pour le cas des variables de base il faut utiliser les pseudo-coûts l_i et u_i qui sont calculés, dans notre cas, en utilisant les pénalités de Driebeek. Ces pénalités correspondent à une évaluation par défaut de l'augmentation de z_{LP}^* si x_i est mise à 0 ou 1 : elles dépendent des valeurs des coûts réduits des variables hors base. Pour les variables de base, nous avons les règles de fixation suivante :

- $\forall x_i \in B^*$, si $(l_i x_i) \geq (UB - z_{LP}^*)$ alors dans toute solution optimale du MIP nous avons $x_i = 1$,
- $\forall x_i \in B^*$, si $(u_i(1-x_i)) \geq (UB - z_{LP}^*)$ alors dans toute solution optimale du MIP nous avons $x_i = 0$.

Cette technique permet donc de connaître la valeur de certaines variables avant la résolution du modèle. La résolution de la relaxation continue d'un modèle est très rapide, le but de ce projet est aussi de montrer que le temps passé à fixer les variables est plus rapide que de le résoudre directement. Dans le cas de notre modèle mathématique, cette technique peut être très intéressante. En effet, si toutes les valeurs des variables sont connues à l'avance, l'ordonnancement de tous les travaux est connu et donc la solution du problème. De plus, connaître la valeur de variables permet aussi d'aider à la résolution du MIP pendant la

phase de *branch-and-bound*. La taille de l'arbre d'exploration est réduite puisqu'il y a moins de variables à valeur inconnue.

Développement

4.1 Intérêt des coupes

Un des avantages de la relaxation continue du modèle MIP est de pouvoir introduire la notion de coupes sur ce modèle LP. En effet, la méthode du prétraitement cherche à fixer des variables avant la résolution en *branch-and-bound* afin de rendre cette résolution plus rapide. Mais il est aussi possible de réduire l'espace des solutions par l'ajout de coupes sur le modèle.

Une coupe est une propriété sur le modèle qui permet de déduire une relation entre les variables et donc d'interdire certaines valeurs de variables. Cela permet de réduire l'espace des solutions comme présenté schématiquement dans la figure 4.1. Une coupe est généralement spécifique au modèle auquel elle s'applique. Il y a cependant certains principes de coupes qui sont communs à tous les modèles. Par exemple, la méthode de construction de la coupe sur les coûts réduits présentée ci-dessous (voir 4.3) peut être appliquée à tous les modèles. Cependant, les coupes qui sont effectivement ajoutées au modèle sont dépendantes des variables du modèle.

Néanmoins, l'ajout de coupes alourdi le modèle et peut ralentir sa résolution au lieu de l'accélérer. Il faut donc trouver le bon compromis et déterminer quelles sont les coupes les plus intéressantes à ajouter.

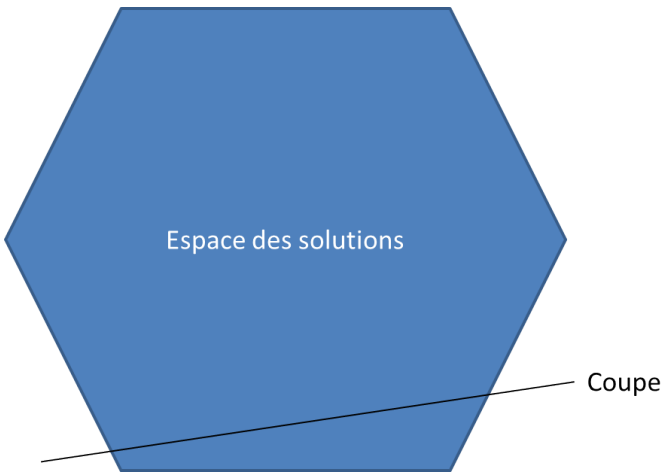


FIGURE 4.1 – Schéma d'une coupe

4.2 Coupe du JPSImproved

4.2.1 Théorie

Cette coupe est en fait une borne inférieur venant de l'algorithme préemptif de Jackson amélioré par Della Croce et T'kindt en $O(\log(n))$. Sa démonstration théorique a été présentée dans le papier *Improving the preemptive bound for the single machine min-max lateness problem subject to release times* [2]. Il s'agit d'une amélioration du *Jackson's semi-preemptive scheduling algorithm*. Cette coupe est réutilisée dans ce

projet et se présente sous cette forme :

$$L_{max} \geq L_{max}^{JPS}$$

Avec L_{max}^{JPS} la valeur du retard maximum de la solution optimale de l'algorithme préemptif de Jackson amélioré.

4.2.2 Algorithme

L'algorithme de cette coupe est très simple. Il consiste simplement en l'exécution du code fourni par V. T'kindt et en la récupération de la valeur du L_{max}^{JPS} afin de l'ajouter dans le modèle.

Algorithm 1 Add JPSImproved cut

$executeJPSImproved()$
$L_{max}^{JPS} \leftarrow getLmaxFromJPSImproved()$
$addCutToLP(L_{max} \geq L_{max}^{JPS})$

4.2.3 Implémentation

Lors de l'implémentation de cette coupe qui s'est faite très tôt, un constat s'est imposé. Les pourcentages de variables fixés avaient changé par rapport aux résultats sans la coupe. En effet, l'ajout de la coupe dans le modèle LP provoque une modification du modèle et donc une modification des coûts réduits et des pseudo coûts et par conséquent des variables fixées. C'est de ce constat qu'est venue l'idée d'effectuer plusieurs passes du prétraitement. En effet, l'intérêt du prétraitement est de connaitre la valeur des variables avant la résolution du MIP. Il faut donc chercher à en avoir le plus possible, si l'ajout d'une coupe permet de fixer de nouvelles variables il est intéressant de fixer les variables avant et après l'ajout de la coupe.

La méthode du prétraitement se présente ainsi après l'ajout de la coupe sur le JPSImproved :

Algorithm 2 Double pass of preprocessing

$solveLP()$
$Preprocessing()$
$addJPSCutToLP()$
$solveLP()$
$Preprocessing()$

4.3 Coupe sur les coûts réduits

4.3.1 Théorie

Les coupes sur les coûts réduits reposent sur un principe similaire à celui utilisé pour fixer les variables hors base. Soit deux variables x_i, $x_j \notin B^*$, nous sommes dans le cas ou x_i et x_j n'ont pas été fixées par le prétraitement sinon l'ajout d'une coupe n'a pas d'intérêt. Nous avons alors $c_i \leq UB - z_{LP}^*$ et $c_j \leq UB - z_{LP}^*$ avec c_i et c_j les coûts réduits de x_i et x_j.

Si dans ce cas nous avons $c_i + c_j > UB + z_{LP}^*$, nous pouvons en déduire que les deux variables x_i et x_j ne peuvent pas être à 1 en même temps. Par conséquent, nous pouvons formuler la coupe suivante pour ces deux variables : $x_i + x_j \leq 1$.

Cette contrainte est inutile dans le modèle relâché. En effet, dans le LP pour une variable booléenne x_i nous avons les propriétés suivantes :

Si $x \in B^* \longrightarrow x^*_{LP} \geq 0$, $c = 0$
Si $x \notin B^* \longrightarrow x^*_{LP} = 0$, $c \geq 0$
Avec x^*_{LP} la valeur optimale de x dans le modèle LP, c le coût réduit associé à la variable x.

Dans ces conditions, dans le LP lorsqu'une variable est une variable hors base sa valeur vaut 0. Les coupes sur les coûts réduits sont donc inutiles dans le modèle relâché, nous les ajoutons seulement dans le MIP avant sa résolution.

4.3.2 Algorithme

L'algorithme de création des coupes sur les coûts réduits est très simple. Il suffit de faire une double boucle parcourant toutes les variables booléennes et de vérifier les conditions énoncées plus tôt. Si toutes les conditions sont remplies, la coupe est ajoutée dans le modèle MIP.

Algorithm 3 Reduced cost cuts creation

```
1: for i ← 1 to numberVariables do
2:     x_i ← variables[i]
3:     c_i ← x_i.getReduceCost()
4:     if x_i ∉ B* AND c_i ≤ UB − z*_LP then
5:         for j ← i to numberVariables do
6:             x_j ← variables[j]
7:             c_j ← x_j.getReduceCost()
8:             if x_j ∉ B* AND c_j ≤ UB − z*_LP then
9:                 addCutToMIP(x_i + x_j ≤ 1)
10:            end if
11:        end for
12:    end if
13: end for
```

4.3.3 Implémentation

Comme dit précédemment, la fonction de création des coupes sur les coûts réduits peut être appliquée à n'importe quel modèle. Ainsi, la fonction a été ajoutée dans l'API de prétraitement.

Dans le cas de notre modèle, il est possible de générer moins de coupes. En effet, le modèle possède la contrainte suivante :

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

$u(i)$ étant la première pyramide au quelle appartient le travail i et $v(i)$ la dernière. Pour chaque travail, il ne peut y avoir qu'une seule variable booléenne qui soit à la valeur 1. Par conséquent, l'ajout des coupes sur les coûts réduits est redondant dans le cas où les deux variables booléennes concernées appartiennent au même travail. C'est ainsi qu'une fonction de création des coupes sur les coûts réduits spécifique au problème du $1|r_i|L_{max}$ a été mise en place. Elle est identique à l'algorithme présenté ci-dessus, mais ajoute une condition : les variables x_i et x_j doivent appartenir à des travaux différents.

Cependant, sur de grands modèles cette fonction peut créer énormément de coupes et alourdir fortement le modèle. Ce phénomène provoque des débordements mémoire et nuit aux résultats. Dans ce contexte,

il était indispensable de générer moins de coupes afin de pouvoir observer la pertinence de l'ajout de ces coupes. Le mécanisme utilisé est en commun avec celui des coupes logiques sur les travaux multi-pyramidaux présentés ci-dessous, celle-ci provoque le même problème d'où la présence d'une solution commune.

La démarche est la suivante, il ne faut pas générer toutes les coupes sur les coûts réduits sinon il en résulte en une surcharge du modèle. Il faut donc générer les coupes sur quelques travaux uniquement. Il reste le problème du choix des travaux pour lesquels nous voulons générer les coupes sur les coûts réduits, il faut générer les coupes les plus pertinentes. En étudiant le cas des coupes logiques sur les travaux multi-pyramidaux il a été décidé de sélectionner les travaux qui, dans le modèle relâche LP, appartiennent au plus grand nombre de pyramides. En effet, ce qui nous intéresse c'est de faire en sorte que la solution du LP soit la plus proche du MIP. Or dans le modèle LP la contrainte d'intégrité du domaine des variables booléennes a été relâchée, un travail peut appartenir à plusieurs pyramides. C'est cette différence qui fait qu'une solution optimale du LP peut être très différente du MIP. Nous cherchons alors à faire en sorte de rajouter des contraintes sur les travaux qui appartiennent à plusieurs pyramides : les travaux multi-pyramidaux.

Néanmoins, l'ajout des coupes sur uniquement les travaux multi-pyramidaux n'est pas suffisant. Ces travaux peuvent être très nombreux, il a donc été décidé de générer les coupes sur les coûts réduits pour les travaux qui appartiennent au plus de pyramides. Expérimentalement, il a été décidé de prendre les 5% supérieurs de travaux multi-pyramidaux en considérant que les travaux sont triés par nombre de pyramides.

Avec cette solution, il a été possible d'obtenir les résultats présentés dans la partie 5.

4.4 Coupe logique sur les travaux multi-pyramidaux

4.4.1 Théorie

Cette coupe a été proposée par Vincent T'kindt et sa démonstration est présente en annexe A.2. Cette coupe est à définir pour chaque travail multi-pyramidal, les travaux qui appartiennent à plusieurs pyramides. Il a été dit précédemment que dans le LP un travail pouvait appartenir à plusieurs pyramides puisque la contrainte d'intégrité de domaine des variables booléennes à été relâchée. Cette relaxation peut entrainer une solution du LP très éloignée de la réalité. Afin de renforcer le LP et de rapprocher sa solution de celles du MIP cette coupe est ajoutée pour chaque travail multi-pyramidal. La définition de la coupe est la suivante :

$$\sum_{k=f}^{l}(R_k - D_k) \geq \sum_{k=f}^{l}\max(\alpha_{kj}, \beta_{kj}, \gamma_{kj}, \delta_{kj}) + p_j$$

- $\alpha_{kj} = r_j - d_j + \sum_{\{i\in P_k, i\neq j | r_i \geq r_j\}} p_i x^+_{ki} + \sum_{\{i\in P_k, i\neq j | d_i \leq d_j\}} p_i x^-_{ki}$,
- $\beta_{kj} = r_j + \sum_{\{i\in P_k, i\neq j | r_i \geq r_j\}} p_i x^+_{ki} + \sum_{\{i\in P_{k+1}, i\neq j\}} p_i x^+_{k+1,i} + p_{k+1} + \sum_{\{i\in P_k, i\neq j\}} p_i x^-_{ki} - D_{k+1}$,
- $\gamma_{kj} = R_{k-1} - d_j + \sum_{\{i\in P_k, i\neq j | d_i \leq d_j\}} p_i x^-_{ki} + \sum_{\{i\in P_{k-1}, i\neq j\}} p_i x^-_{k-1,i} + p_{k-1} + \sum_{\{i\in P_k, i\neq j\}} p_i x^+_{ki}$,
- $\delta_{kj} = R_{k-1} + \sum_{\{i\in P_{k+1}, i\neq j\}} p_i x^+_{k+1,i} + \sum_{\{i\in P_{k-1}, i\neq j\}} p_i x^-_{k-1,i} + \sum_{\{i\in P_k, i\neq j\}} p_i x^+_{ki} + \sum_{\{i\in P_k, i\neq j\}} p_i x^-_{ki} + p_{k-1} + p_{k+1} - D_{k+1}$

Où l est la dernière pyramide auquelle appartient le travail j et f la première. Il est important de noter que dans le cas ou $k+1$ n'est pas définie $D_{k+1} = +\infty$. Dans tous les autres cas, les valeurs non définies sont considérées comme nulles.

4.4.2 Algorithme

Algorithm 4 Logical cuts creation

for $k \leftarrow 1$ to $nbPyramides$ **do**
  for all $job_j \in pyramide_k$ **do**
    if $job_j.getNbPyramides() > 1$ **then**
      for all $job_i \in pyramide_k$ AND $i \neq j$ **do**
        if $r_i \geq r_j$ **then**
          $A^+_{kj} \leftarrow A^+_{kj} + p_i x^+_{ki}$
        end if
        if $d_i \leq d_j$ **then**
          $B^-_{kj} \leftarrow B^-_{kj} + p_i x^-_{ki}$
        end if
        $T^+_{kj} \leftarrow T^+_{kj} + p_i x^+_k i$
        $T^-_{kj} \leftarrow T^-_{kj} + p_i x^-_k i$
      end for
    end if
  end for
end for
for $j \leftarrow 1$ to $nbJobs$ **do**
  if $job_j.getNbPyramides() > 1$ **then**
    for all $pyramide_k$ containing job_j **do**
      $\alpha_{kj} \leftarrow r_j - d_j + A^+_{kj} + B^-_{kj}$
      if $k < nbPyramides$ **then**
        $\beta_{kj} \leftarrow r_j + A^+_{kj} + T^+_{k+1,j} + p_{k+1} + T^-_{kj} - D_{k+1}$
      else
        $\beta_{kj} \leftarrow +\infty$
      end if
      if $k > 1$ **then**
        $\gamma_{kj} \leftarrow R_{k-1} - d_j + B^-_{kj} + T^-_{k-1,j} + p_{k-1} + T^+_{kj}$
      else
        $\gamma_{kj} \leftarrow -d_j + B^-_{kj} + T^+_{kj}$
      end if
      $\delta_{kj} \leftarrow T^+_{kj} + T^-_{kj}$
      if $k > 1$ **then**
        $\delta_{kj} \leftarrow delta_{kj} + R_{k-1} + T^-_{k-1,j} + p_{k-1}$
      end if
      if $k < nbPyramides$ **then**
        $\delta_{kj} \leftarrow delta_{kj} + T^+_{k+1,j} + p_{k+1} - D_{k+1}$
      end if
    end for
  end if
  $addCutToLP(\sum_{k=f}^{l}(R_k - D_k) \geq \sum_{k=f}^{l}\max(\alpha_{kj}, \beta_{kj}, \gamma_{kj}, \delta_{kj}) + p_j)$;
end for

4.4.3 Implémentation

L'implémentation de l'algorithme présenté ci-dessus ne peut pas être faite directement. Il n'est pas possible d'ajouter une contrainte avec une fonction max(). Il faut créer une variable intermédiaire pour obtenir le comportement de la fonction max(). La contrainte se découpe alors en 5 contraintes :

- $M_{kj} \geq \alpha_{kj}$,
- $M_{kj} \geq \beta_{kj}$,
- $M_{kj} \geq \gamma_{kj}$,
- $M_{kj} \geq \delta_{kj}$,
- $\sum_{k=f}^{l}(R_k - D_k) \geq \sum_{k=f}^{l} M_{kj} + p_j$

Il faut donc ajouter 5 contraintes par travail multi-pyramidal. Comme pour l'ajout des coupes logiques sur les coûts réduits, l'ajout de toutes les coupes logiques est trop gourmand en mémoire et surcharge le modèle très rapidement. La solution expliquée en partie 4.3.3 est la même que celle utilisée ici afin de réduire le nombre de coupes à ajouter. Cependant, le problème n'est pas complètement disparu et le choix des coupes ajoutées peut être discuté.

L'ajout de ces coupes modifie le modèle LP, une nouvelle passe de prétraitement est alors effectuée après l'ajout des coupes logiques comme pour la coupe sur le JPSImproved (partie 4.2.3).

4.5 Recherche locale

Après la présentation à ROADEF des premiers résultats, un constat a été fait. Les résultats initiaux (voir partie 5.1) sont très éloignés des résultats "idéaux" avec pour borne supérieure la valeur optimale donnée par la résolution du MIP seul (voir partie 5.2). Lorsque la solution optimale est placée en borne supérieure, les résultats du prétraitement sont très bons. Cette borne supérieure est idéale, mais en quelque sorte une triche, nous connaissons la valeur optimale et l'utilisons pour le prétraitement. Il est cependant possible de chercher à s'en rapprocher en améliorant l'heuristique actuelle de la borne supérieure. C'est pour cette raison qu'il a été décidé de chercher à améliorer la solution donnée par l'algorithme de Schrage par une recherche locale.

4.5.1 Théorie

L'algorithme de Schrage permet d'obtenir une borne supérieure à la solution optimale. Si nous cherchons à baisser la valeur de cette solution, nous nous rapprocherons de la valeur optimale et donc des résultats obtenus en partie 5.2.

Pour rappel, l'algorithme de Schrage est le suivant. À un instant t, nous considérons tous les travaux i disponibles ($r_i \leq t$). Ces travaux sont ensuite triés par ordre croissant de date de fin au plus tard. Le travail avec le d_i le plus bas est le travail choisi. La date courante est mise à jour ($t = t + p_i$, avec i le travail sélectionné), puis la même opération est répétée jusqu'à ce que tous les travaux soient ordonnancés.

Cet algorithme donne en sortie un ordonnancement de tous les travaux avec une valeur de L_{max}. Afin d'effectuer la recherche locale, nous étudions la séquence de travaux donnée par Schrage en détail. La valeur du L_{max} est donnée par le ou les travaux ayant le retard maximum, nous appellerons ces travaux des travaux critiques. Afin de réduire la valeur du L_{max} de la séquence, il faut placer le ou les travaux critiques plus tôt dans la séquence d'ordonnancement. Il faut cependant chercher à ne pas trop perturber l'ordonnancement au risque de dégrader la valeur du L_{max}.

Nous voulons donc essayer de placer les travaux critiques vers la gauche de la séquence (avec un axe de temps classique gauche-droite) et donc de réduire son retard tout en espérant réduire le L_{max} de la séquence. Il faut tout d'abord définir un ou plusieurs opérateurs de voisinage afin de créer les voisins de la séquence de Schrage. Un premier opérateur simple est l'échange d'un travail critique avec un travail situé à sa gauche ayant un retard négatif. De cette façon, nous plaçons le retard critique plus tôt dans la séquence afin de réduire son retard et nous le remplaçons par un travail qui n'avait pas de retard auparavant. La figure 4.2 montre schématiquement cet opérateur de voisinage.

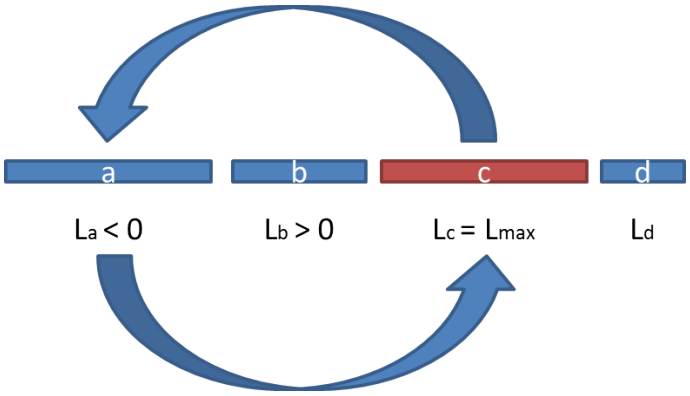


FIGURE 4.2 – Schéma présentant l'échange d'un travail a avec le travail critique c

Un deuxième opérateur de voisinage a été défini, l'insertion d'un travail i situé avant un travail critique ayant un $L_i < 0$ dans une position après le travail critique. De cette façon, nous déplaçons le travail critique vers la gauche afin de réduire son retard et nous plaçons un travail qui n'était pas en retard plus loin dans la séquence. La figure 4.3 montre schématiquement cet opérateur de voisinage.

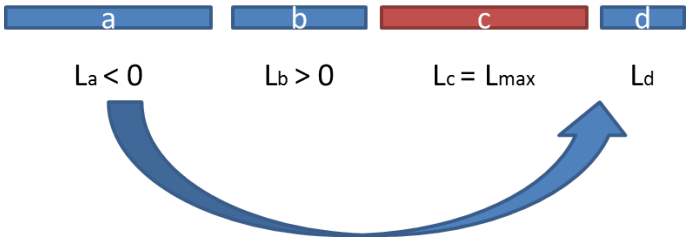


FIGURE 4.3 – Schéma présentant l'insertion d'un travail a avant le travail critique c

Les opérateurs de voisinage sont en place, il faut maintenant décider de la boucle principale de la recherche locale et de la sélection du meilleur voisin. Ce que nous cherchons à faire avec les opérateurs de voisinage c'est de diminuer le retard du travail critique concerné tout en cherchant à ne pas augmenter la valeur du L_{max} de la séquence. Ce seront donc ces deux critères qui détermineront le choix du meilleur voisin, le meilleur voisin sera celui qui aura le plus diminué le retard du travail critique tout en ne faisant pas augmenter la valeur du L_{max} de la séquence entière, ou celui qui aura diminué le plus le L_{max}. Si aucun voisin ne correspond à ces conditions, la recherche locale est terminée.

Il reste encore un point à éclaircir, il a été dit plus haut qu'il était possible d'avoir plusieurs travaux critiques. Or il n'est pas spécifié qu'il fallait appliquer les opérateurs de voisinages à tous les travaux critiques, mais uniquement au premier. En effet, la boucle de voisinage le permet déjà de par sa structure. Si la recherche locale diminue le retard du premier travail critique tout en conservant la valeur du L_{max} le prochain premier travail critique sera le ou les travaux critiques précédents qui n'ont pas été pris en compte (sauf si un autre travail critique est apparu plus tôt dans la séquence).

4.5.2 Algorithme

Algorithm 5 Local research

```
1: bestOrdo ← ordoSchrage
2: while end = false do
3:     end ← true
4:     ordo ← bestOrdo
5:     bestNeighbor ← ordo
6:     criticalJob ← ordo.getFirstCriticalJob()
7:     bestCriticalJobLateness ← criticalJob.getLateness()
8:     for all job_j before criticalJob in ordo do
9:         if job_j.getLateness() < 0 then
10:            ordo.swap(job_j, criticalJob)
11:            if (ordo.getLmax() < bestNeighbor.getLmax()) OR (criticalJob.getLateness() <
    bestCriticalJobLateness() AND ordo.getLmax() <= bestNeighbor.getLmax()) then
12:                bestNeighbor ← ordo
13:                bestCriticalJobLateness ← criticalJob.getLateness()
14:            end if
15:            for all position after criticalJob in ordo do
16:                ordo.insert(Job, position)
17:                if (ordo.getLmax() < bestNeighbor.getLmax()) OR (criticalJob.getLateness() <
    bestCriticalJobLateness() AND ordo.getLmax() <= bestNeighbor.getLmax()) then
18:                    bestNeighbor ← ordo
19:                    bestCriticalJobLateness ← criticalJob.getLateness()
20:                end if
21:            end for
22:        end if
23:    end for
24:    if bestNeighbor ≠ bestOrdo then
25:        bestOrdo ← bestNeighbor
26:        end ← false
27:    end if
28: end while
```

4.5.3 Implémentation

L'implémentation de la recherche locale ne présente pas de difficultés particulières. Il y a eu cependant un point qui a ralenti cette tâche. L'implémentation existante de l'algorithme de Schrage ne permettait pas de récupérer la séquence des travaux avec les données nécessaires à la recherche locale, et ce, à cause des structures de données utilisées. L'algorithme de Schrage a donc été ré-implémenté avant de pouvoir mettre en place la recherche locale.

4.6 Génération d'instances

Une étape importante lors de la validation expérimentale des différentes solutions mises en place est la génération d'instances. En effet, nous souhaitons réaliser des tests de validation et des statistiques sur de nombreux jeux de données afin de pouvoir tirer des conclusions sur les méthodes utilisées pour la résolution. Nous voulons être sûrs que notre méthode trouve la même solution optimale, qu'il n'y a pas d'erreurs en

somme, et qu'une méthode de résolution est plus ou moins performante qu'une autre.

Afin de réaliser cela, il faut donc un grand nombre d'instances de travaux de tailles et valeurs différentes. Soit nous possédons un jeu de données et il suffit donc de lancer un testeur sur ces données et d'en tirer des statistiques, soit il faut générer ces instances dans le testeur. Pendant ce projet, plusieurs générations d'instances ont été utilisées, mais seulement deux ont été retenues.

La première est celle qui est utilisée pour les tableaux sur les instances dites "faciles". Il s'agit de la génération présentée dans le papier de présentation du modèle mathématique pyramidal présent en annexe A.1. Cette génération est la suivante :

Nous prenons pour taille d'instances $n \in \{100, 400, 700, ..., 2500\}$, la durée opératoire des travaux suit une loi aléatoire uniforme U dans l'intervalle $[1, 50]$, nous avons donc $p_i \in U[1, 50]$. La valeur r_i des travaux suit une loi aléatoire uniforme dans l'intervalle $[0, \alpha \times \sum_{i=1}^{n} p_i]$ avec α prenant 5 valeurs : $\{0,2; 0,4; 0,6; 0,8; 1\}$. De la même façon, la valeur d_i suit une loi aléatoire uniforme dans l'intervalle $[(1-\beta) \times \sum_{i=1}^{n} p_i, a \times \sum_{i=1}^{n} p_i]$ avec le paramètre $a \in \{100\%, 110\%\}$ et le paramètre $\beta \in \{0,2; 0,4; 0,6; 0,8; 1\}$. De plus, pour respecter la cohérence entre les r_i, d_i et p_i, si la valeur $d_i < r_i + p_i$ elle prend la valeur $r_i + p_i$. Pour chaque valeur de n la variation des 3 paramètres α, β et a permet de générer 50 instances.

Cette première génération n'était pas satisfaisante puisque les instances générées sont faciles à résoudre, le tableau de résultat 5.2 montre que 47% des instances générées en moyenne sont des instances dont l'algorithme de Schrage permet de prouver l'optimum et que 40 nœuds sont parcourus en moyenne. Différentes générations ont donc été essayées, mais c'est la génération suivante qui a permis de générer la base des 32 instances "difficiles".

Cette génération est similaire à celle présentée ci-dessus, la seule différence est située au niveau de la définition de p_i. Nous avons $p_i = U[1, 50] \times 50n$, nous avons multiplié la valeur de p_i par 50 fois n. Cette différence permet d'obtenir de grande valeur de p_i, d_i et r_i ce qui augmente la valeur d'écart entre les travaux. Cette génération a ensuite été utilisée pour trouver des instances très difficiles, elles ont été retenues lorsque leur résolution dépassait les 10000 nœuds. Ces instances constituent un échantillon d'instances très dures utilisées pour le deuxième testeur qui n'utilise pas de générateur d'instances, mais parcourt cet échantillon.

Résultats expérimentaux

Cette partie présente les résultats des différentes campagnes de test qui ont été effectuées. Ces résultats permettent de faire des déductions sur les différentes solutions développées. Il est aussi important de noter que les testeurs ont servi à effectuer des tests de validation pendant les campagnes de tests. Le test de validation principal a été de vérifier que les solutions optimales obtenues étaient les mêmes entre la version du code sans prétraitement et celle avec prétraitement. La limite de temps d'exécution était fixée à 900s, toute exécution $> 900s$ n'est donc pas résolue à l'optimum.

La génération des instances faciles et difficilse est présentée plus haut (4.6). Pour les instances faciles, chaque ligne est le résultat de l'exécution de 50 instances de n travaux. Dans les tableaux suivants, différentes abréviations et symboles sont utilisés :

- $T =$ Temps d'exécution total.
- $T^{Pre} =$ Temps d'exécution de la phase de prétraitement.
- $n =$ Nombre de travaux de l'instance.
- *%inst résolues* = Le pourcentage d'instances résolues à l'otpimum.
- *%inst non Schrage Opti* = Le pourcentage d'instances qui ne sont pas Schrage Optimale. Une instance est Schrage optimale si le L_{max} de la séquence donnée par l'algorithme de Schrage est égale au L_{max} donné par le LP avant la phase de prétraitement.
- $\%VF =$ Le pourcentage de variables booléennes fixées par le prétraitement.
- $\%AmélioUB =$ Le pourcentage d'amélioration de la borne supérieure. Il s'agit du pourcentage d'amélioration de la solution de Schrage par la recherche locale.
- *MOY* = Moyenne, cette ligne fait la moyenne de toute les lignes du tableau.
- $\oslash =$ Pas de donnée.

5.1 Résultats initiaux

Les tableaux 5.1 et 5.3 présentent les résultats lors de la résolution du modèle mathématique directement sans prétraitement. Ces tableaux vont servir de comparaison pour les autres tableaux. Nous cherchons à prouver que le prétraitement plus la résolution du modèle est plus rapide que la résolution seule du modèle.

La comparaison des tableaux sur les instances faciles avec (5.2) et sans prétraitement (5.1) permet de faire quelques remarques. Le temps moyen d'exécution est moins bon pour le code avec prétraitement (24,57s contre 36,31s) ainsi que le nombre de nœuds moyen, globalement le code avec prétraitement prend plus de temps. L'étude des résultats avec UB = MIP optimal (partie 5.2) permettra d'analyser plus en profondeur la raison de ces résultats.

Pour le cas des instances difficiles, (tableaux 5.3 et 5.4) le constat global est le même. Le temps moyen d'exécution est globalement moins bon. Il est cependant important de noter que sur les deux instances avec le temps en vert (tableau 5.4) sont résolues à l'optimum dans le cas du prétraitement alors que dans la résolution directe du modèle non. Les 1% et quelques de variables booléennes fixées ont permis de prouver l'optimalité. Le prétraitement commence à présenter un intérêt. Nous avons cependant le cas inverse qui se présente. Les 3 lignes avec le temps en orange sur le tableau 5.4 ne sont plus résolues à l'optimum. Pour la deuxième ligne orange seulement 3% des variables ont été fixées. Cet effet de bord peut s'expliquer par

le fait que l'arbre de recherche lors de la résolution du MIP est modifié lorsque l'on fixe des variables et il n'est pas impossible de se trouver dans un endroit complètement différent que lors de la résolution du MIP seul et que cet endroit ne mène pas à la solution optimale en un temps suffisant.

	(MIP)						
n	%inst résolues	T_{min}	T_{moy}	T_{max}	#Noeuds$_{min}$	#Noeuds$_{moy}$	#Noeuds$_{max}$
100	100	0,06	0,09	0,39	0	12,70	464
400	100	0,19	0,59	4,35	0	44,62	1013
700	100	0,49	1,57	8,74	0	0,22	11
1000	100	0,98	6,44	44,97	0	12,26	463
1300	100	2,43	11	71,03	0	10,32	486
1600	100	3,07	27,25	239,71	0	88,42	3407
1900	100	5,69	36,47	233,18	0	26,20	510
2200	100	7,59	59,79	387,73	0	26,06	483
2500	100	8,10	77,96	458,84	0	17,20	400
MOY	**100**	**3,18**	**24,57**	**160,99**	**0**	**26,44**	**804,11**

TABLE 5.1 – Résultats sur les instances faciles pour MIP

	Prétraitement puis (MIP)													
n	%inst résolues	%inst non Schrage Opti	T_{min}	T_{moy}	T_{max}	T^{Pre}_{min}	T^{Pre}_{moy}	T^{Pre}_{max}	#Noeuds$_{min}$	#Noeuds$_{moy}$	#Noeuds$_{max}$	$\%VF_{Min}$	$\%VF_{Moy}$	$\%VF_{Max}$
100	100	38	0,02	0,09	0,35	0	0,01	0,03	0	12,42	500	0	15,22	60,92
400	100	44	0,14	1,10	12,63	0,04	0,15	0,45	0	51,58	624	0	7,02	61,23
700	100	38	0,59	2,65	15,62	0,12	0,57	2,15	0	11,80	200	0	10,93	46,22
1000	100	50	0,92	8,71	44,07	0,20	1,50	6,29	0	13	400	0	6,49	54,49
1300	100	50	1,84	18,31	93,66	0,39	2,90	13,04	0	54,88	510	0	9,02	49,71
1600	100	54	3,62	41,01	379,75	0,59	5,80	26,52	0	61,92	969	0	10,42	67,06
1900	98	50	4,34	54,85	457,71	1,08	8,07	33,67	0	31,55	511	0	4,38	37,57
2200	100	52	6,51	85,28	433,53	0,80	12,67	66,72	0	57,66	528	0	7,92	75,76
2500	100	50	8,88	114,78	631,08	1,44	17,76	99,89	0	58,64	521	0	3,58	14,15
MOY	**99,78**	**47,33**	**2,98**	**36,31**	**229,82**	**0,52**	**5,49**	**27,64**	**0**	**39,27**	**529,22**	**0**	**8,33**	**51,9**

TABLE 5.2 – Résultats sur les instances faciles pour prétraitement puis MIP

	(MIP)	
n	T	#Noeuds
100	900	1660965
100	92,15	175710
100	900,02	1756717
100	900,02	1959113
100	1,73	3134
100	900,01	1757550
200	900,02	1256019
300	900,03	173182
300	900	349850
300	900,04	66676
300	900,01	1098891
400	83,85	50020
400	899,84	57883
400	900,02	20911
500	900,02	130256
600	900,03	15307
600	900,02	27594
600	900,02	15349
700	900,03	1090792
700	455,11	235458
700	900,02	282021
700	900,05	7038
700	67,99	11698
800	900,06	6489
800	12,93	3323
800	524,50	74902
900	900,03	12167
900	23,48	8078
900	900,01	263710
900	900,05	30725
900	900,03	53401
1000	900,05	31577
MOY	**714,44**	**396453,31**

TABLE 5.3 – Résultats sur les instances difficiles pour MIP

	Prétraitement puis (MIP)			
n	T^{PRE}	T	#Noeuds	% VF
100	0,04	900,06	1511826	0
100	0,03	7,54	9065	13,74
100	0,03	900,06	1618799	5,12
100	0,04	900,07	1810405	12,71
100	0,03	2,46	3809	10,26
100	0,03	900,07	1717984	0,35
200	0,06	900,08	1548599	26,91
300	0,42	900,49	212765	6,96
300	0,39	7,55	1502	1,32
300	0,16	900,23	90776	2,41
300	0,35	900,45	475601	48,77
400	0,56	89,72	18233	55,23
400	0,78	900,82	123943	0,45
400	0,96	901,06	50019	0
500	1,11	901,23	62031	4,96
600	0,87	901,04	19027	0
600	1,02	901,15	30012	0
600	2,56	684,83	12527	1,64
700	1,55	901,77	152631	73,28
700	0,64	900,91	581284	61,81
700	0,19	900,27	66104	0,41
700	3,22	905,04	10607	0,20
700	2,53	825,45	261979	42,89
800	7,02	907,62	10775	0,64
800	0,62	83,46	31064	42,81
800	5,02	905,28	77174	3,38
900	3,25	903,53	11437	5,16
900	3,75	904,11	95695	46,77
900	2,98	903,36	44559	25,89
900	3,44	903,67	43657	5,59
900	2,23	903,46	10838	9,36
1000	5,18	907,26	10775	2,88
MOY	**1,60**	**757,94**	**335171,94**	**16,00**

TABLE 5.4 – Résultats sur les instances difficiles pour prétraitement puis MIP

5.2 Résultats avec UB = MIP Optimal

Une variable est fixée ou non selon l'inégalité présentée en partie 3.3 qui dépend de la valeur de la borne supérieure UB. Plus la borne supérieure est basse plus le taux de variables fixées sera grand. C'est pourquoi la campagne de test de cette partie a été faite. Il est intéressant de voir les résultats du prétraitement avec la borne supérieure idéale, la valeur optimale elle-même. Ces résultats ne peuvent pas être réutilisés directement, ils ont été réalisés et présentés dans cette partie pour obtenir des pistes d'avancées du projet.

La comparaison du tableau 5.5 présentant la version du prétraitement avec pour borne supérieure la valeur optimale avec le tableau 5.1 de la résolution classique du modèle sur les instances faciles est très intéressante et va permettre de justifier certains choix qui ont été faits par la suite. Cette version du prétraitement possède un temps d'exécution moyen et un nombre de nœuds meilleur que la version MIP seule. La comparaison est sans appel, toutes les lignes du tableau sont meilleures. De plus, le pourcentage de variables fixées est le plus haut de toutes les versions du prétraitement. Ceci est facilement explicable puisque la borne supérieure ne peut pas être meilleure que celle utilisée ici. C'est de ce constat qu'est venue l'idée d'améliorer la valeur de la borne supérieure actuelle (obtenue avec l'algorithme de Schrage) avec une recherche locale. Il serait aussi possible d'utiliser une borne supérieure différente ou d'améliorer la borne supérieure actuelle différemment. De plus, le pourcentage d'instances optimales avant la phase de prétraitement est proche de 99%, l'optimum est trouvé très rapidement après résolution du LP dans presque tous les cas ce qui explique les résultats obtenus. Les résultats du tableau 5.5 prouvent dans tous les cas que si la borne supérieure est idéale, les résultats du prétraitement sont très bons.

L'étude des tableaux de résultats sur les instances dures (tableaux 5.6 et 5.3) permet de dégager une nouvelle piste de progression. Le même constat que pour les instances faciles peut être fait ici. Le temps d'exécution moyen est très largement inférieur à celui du tableau de résultat de MIP seul. Ce qui va nous intéresser ici ce sont les lignes qui ne sont pas résolues à l'optimum et la ligne orange du tableau 5.6 également. Les instances qui ne sont toujours pas résolues à l'optimum prouvent que la borne supérieure ne fait pas tout même si elle permet d'obtenir de nombreuses instances dites Schrage optimales (bien que Schrage ne soit pas utilisé ici, l'optimum est prouvé après la résolution du LP la borne inférieur est égale à la borne supérieure) caractérisées par le ⊘ dans la ligne du pourcentage de variables fixées. Étant donné que la borne supérieure est idéale, il serait possible de renforcer le LP. Il est possible d'étudier les propriétés du modèle et du problème afin d'ajouter des coupes pour que la solution du LP se rapproche d'une solution réalisable, ce sera notamment l'approche adoptée lors de la création des coupes logiques. La ligne avec le temps en orange montre encore une fois un des effets de bords du prétraitement, 79% des variables ont été fixées, mais la solution n'est pas résolue à l'optimum. Il est possible d'imaginer encore une fois que l'arbre de recherche a été modifié et que le hasard fait qu'il se peut que le restant de l'arbre soit très désavantageux.

	Prétraitement avec UB = MIP Optimal puis (MIP)													
n	%inst résolues	%inst non Schrage Opti	T_{min}	T_{moy}	T_{max}	T^{Pre}_{min}	T^{Pre}_{moy}	T^{Pre}_{max}	#Noeuds$_{min}$	#Noeuds$_{moy}$	#Noeuds$_{max}$	$\%VF_{Min}$	$\%VF_{Moy}$	$\%VF_{Max}$
100	100	0	0,01	0,04	0,21	0	0,01	0,02	0	1,20	60	10,77	31	45,04
400	100	0	0,14	0,46	5,60	0,04	0,13	0,30	0	17,32	864	2,89	30,61	76,13
700	100	2	0,61	1,43	7,53	0,12	0,49	1,49	0	0	0	9,52	16,01	22,50
1000	100	0	0,92	3,22	10,22	0,20	1,14	3,41	0	0	0	9,38	9,38	9,38
1300	100	4	1,87	7,58	32,60	0,40	2,36	8,41	0	4	200	1,64	3,49	5,60
1600	100	0	3,64	12,34	48,62	0,60	4,49	14,87	0	0	0	59,21	59,21	59,21
1900	100	0	4,42	19,21	40,42	1,15	6,75	22,71	0	0	0	0	0	0
2200	100	2	6,85	29,68	70,87	0,83	9,99	42,36	0	0,02	1	0,75	0,75	0,75
2500	100	0	9,11	40,88	86,17	1,41	13,70	54,90	0	0	0	0	0	0
MOY	**100**	**0,89**	**3,06**	**12,76**	**33,58**	**0,53**	**4,34**	**16,5**	**0**	**2,5**	**125**	**10,46**	**16,72**	**24,29**

TABLE 5.5 – Résultats sur les instances faciles pour prétraitement avec UB = MIP Optimal puis MIP

	Prétraitement avec UB = MIP Optimal puis (MIP)			
n	T^{Pre}	T	#Noeuds	% VF
100	0,03	900,09	1893728	0
100	0,02	6,32	9065	14,10
100	0,03	900,12	1969842	12,72
100	0,02	900,10	4047084	16,65
100	0,02	9,77	27141	38,67
100	0,03	900,10	2212185	3,55
200	0,04	900,14	2191950	34,38
300	0,23	900,58	272596	12,11
300	0,24	816,65	592968	1,35
300	0,10	900,37	294791	16,38
300	0,24	423,10	254465	51,37
400	0,27	900,81	788533	79,20
400	0,23	0,45	0	⊘
400	0,29	0,53	0	⊘
500	0,65	0,98	0	⊘
600	0,57	901,76	27244	0,20
600	0,56	901,70	37539	0
600	1,69	208,45	7545	3,32
700	0,73	1,73	0	⊘
700	0,32	1,08	0	⊘
700	0,16	900,78	188345	0
700	0,94	1,73	0	⊘
700	0,88	1,71	0	⊘
800	1,11	2,34	0	⊘
800	0,33	1	0	⊘
800	1,17	2,11	0	⊘
900	1,89	3,64	0	⊘
900	1,18	3,22	0	⊘
900	0,75	3,09	0	⊘
900	1,96	3,37	0	⊘
900	1,29	2,54	0	⊘
1000	2,60	4,89	0	⊘
MOY	**0,64**	**356,41**	**462969,41**	**17,75**

TABLE 5.6 – Résultats sur les instances difficiles pour prétraitement avec UB = MIP Optimal puis MIP

5.3 Résultats avec recherche locale

La comparaison des résultats sur les instances faciles entre MIP seul (tableau 5.1) et la version avec prétraitement et recherche locale (tableau 5.7) est intéressante. En effet, le temps moyen d'exécution est globalement moins bon, mais le nombre de nœuds moyen parcouru est meilleur. Cela peut s'expliquer par le fait que la recherche locale prends du temps et que l'algorithme est un peu gourmand, mais avec un taux d'amélioration moyen du L_{max} de Schrage de 0,06 par la recherche locale le pourcentage de variables fixées est légèrement meilleur et le nombre de nœud parcouru plus faible. L'amélioration de la borne supérieure a donc un réel impact positif sur les résultats du prétraitement sur ce problème. Il serait peut-être intéressant de changer la recherche locale pour la rendre moins gourmande en temps de calcul afin d'améliorer les temps d'exécutions globaux.

Dans le cas des instances difficiles entre les versions du code avec (tableau 5.8) et sans recherche locale (tableau 5.4), deux phénomènes sont à observer. La première ligne du tableau 5.8 avec le temps en vert est résolue à l'optimum contrairement à précédemment, l'amélioration de la borne supérieure à permis de fixer plus de variables et de résoudre l'instance à l'optimum. La ligne orange et la deuxième ligne verte du tableau 5.8 montrent le deuxième phénomène. L'instance de la ligne avec la cellule orange n'est plus résolue à l'optimum alors que le pourcentage de variables fixées est le même. Il est possible d'imaginer que ce ne sont pas les mêmes variables qui sont fixées, mais il a été vérifié que sont bien les mêmes dans les deux cas. Cet effet de bord est dû à la solution initiale qui est fournie au MIP avant sa résolution. La solution initiale qui est donc Schrage amélioré est responsable de ce deuxième phénomène. La deuxième ligne verte du tableau 5.8 présente le même effet, mais avec le résultat inverse, l'instance est résolue à l'optimum alors qu'elle ne l'était pas sans la recherche locale. Ce point est encore assez mystérieux et mériterait approfondissement.

Au vu des résultats, il a été décidé de combiner la recherche locale avec l'ajout des coupes afin de chercher à améliorer la solution sur les deux fronts possibles.

	Prétraitement et recherche locale puis (MIP)																
n	%inst résolues	%inst non Schrage Opti	T_{min}	T_{moy}	T_{max}	T^{Pre}_{min}	T^{Pre}_{moy}	T^{Pre}_{max}	#Nœuds$_{min}$	#Nœuds$_{moy}$	#Nœuds$_{max}$	$\%VF_{Min}$	$\%VF_{Moy}$	$\%VF_{Max}$	$\%Amélio\ UB_{Min}$	$\%Amélio\ UB_{Moy}$	$\%Amélio\ UB_{Max}$
100	100	26	0,02	0,08	0,37	0	0,01	0,03	0	7,14	300	0	15,64	43,44	0	6,91	73,33
400	100	28	0,15	1,25	6,46	0,04	0,15	0,45	0	25,84	496	0	12,33	72,18	0	8,92	200
700	100	20	0,66	4,01	14,41	0,12	0,55	2,18	0	4	200	0	14,65	46,22	0	3,49	53,85
1000	100	32	0,99	14,13	48,46	0,20	1,36	5,01	0	2	50	0	9,89	54,49	0	7,77	100
1300	100	26	1,99	29,17	96,28	0,39	2,85	13,65	0	8	400	0	16,07	49,71	0	5,61	81,82
1600	100	32	3,79	57,27	224,04	0,61	5,60	27,67	0	57,76	2337	0,17	15,32	67,06	0	5,69	90,91
1900	100	28	4,66	79,88	209,47	1,11	8,23	36,59	0	9,30	200	0	7,23	37,57	0	6,35	69,12
2200	100	38	7,79	150,47	632,45	0,83	12,57	66,99	0	38,50	513	0,68	10,75	75,76	0	5,61	100
2500	100	36	8,82	193,59	711,13	1,35	16,54	96,01	0	4	200	0	4,67	14,15	0	3,92	50
MOY	**100**	**29,56**	**3,21**	**58,87**	**215,9**	**0,52**	**5,32**	**27,62**	**0**	**17,39**	**521,78**	**0,09**	**11,84**	**51,18**	**0**	**6,03**	**91**

TABLE 5.7 – Résultats sur les instances faciles pour prétraitement avec recherche locale puis MIP

	Prétraitement et recherche locale puis (MIP)				
n	T^{Pre}	T	#Noeuds	% VF	%*Amélio* UB
100	0,02	900,10	1873739	0	11,15
100	0,02	15,22	23723	13,74	0
100	0,03	900,12	1653675	8,30	42,79
100	0,01	900,12	2996216	13,98	6,84
100	0,03	1,49	2847	18,40	4,48
100	0,03	900,12	2102759	2,13	26,25
200	0,04	900,20	2008887	30,96	5,01
300	0,25	900,97	162166	7,89	5,74
300	0,24	28,77	10839	1,32	0,07
300	0,12	900,74	179010	14,67	52,25
300	0,23	901,07	879183	48,77	0,05
400	0,29	100,21	30884	56,97	0,53
400	0,40	901,71	92422	0,45	1,18
400	0,51	901,75	113129	0,67	16,46
500	0,83	903,12	117345	8,60	8,24
600	0,59	906,60	38888	0,20	8,65
600	0,59	906,69	30874	0	5,31
600	1,90	165,08	10127	1,74	0,90
700	0,96	906,82	389640	75,14	0,29
700	0,47	699,70	206406	63,69	0,27
700	0,16	903,38	82513	0,41	0,21
700	1,53	909,93	13239	0,20	6,46
700	1,51	47,70	11413	42,89	0,35
800	2,74	917,28	14580	0,75	5,33
800	0,49	902,42	358007	42,81	0
800	2,04	752,14	120806	3,38	0
900	2,32	914,76	11228	5,61	6,67
900	2,18	917,15	69905	47,83	0,03
900	1,41	918,77	270955	69,05	16,40
900	2,69	914,89	29941	6,53	9,72
900	1,79	915,19	43383	9,36	0,51
1000	3,44	923,83	17017	3,44	9,76
MOY	**0,93**	**736,81**	**436429,56**	**18,75**	**7,87**

TABLE 5.8 – Résultats sur les instances difficiles pour prétraitement avec recherche locale puis MIP

5.4 Résultats avec coupes sur coups réduits

Les résultats du prétraitement avec coupes sur les coûts réduits pour la résolution du MIP sur les instances faciles présentées dans le tableau 5.9 montrent les limites de la coupe. La chose la plus évidente et importante à noter dans ce tableau de résultat est qu'il est incomplet. En effet comme dit dans la partie 4.3.3, l'ajout des coupes sur les coûts réduits est énormément coûteux en terme de mémoire. La raison pour laquelle il n'y a pas de résultats à partir de 1500 travaux est que les problèmes de mémoire commencent à se manifester à ce moment-là. Bien qu'uniquement 5% des coupes sur les travaux multi-pyramidaux sont générées la génération de ces coupes continue à poser des problèmes de mémoire, il serait possible de réduire encore le nombre de coupes générées, mais en plus de perdre l'intérêt de leur ajout cela ne ferait que reporter le problème à des tailles d'instances plus grandes.

La comparaison des résultats avec le tableau 5.1 de la version MIP seul sur les instances faciles permet en plus de montrer que l'impacte des coupes sur les coûts réduits est presque insignifiant voir négatif.

L'étude des tableaux de résultats (tableaux 5.10 et 5.3) pour les instances difficiles permet une étude plus minutieuse de l'impact de l'ajout des coupes sur les coûts réduits. La ligne avec le temps en orange dans le tableau 5.10 montre que l'ajout des coupes peut avoir un effet négatif et dégrader la solution, l'optimale n'est plus trouvée alors qu'elle l'était dans la version sans prétraitement et aussi dans la version avec prétraitement et recherche locale uniquement. Au contraire, l'instance avec le temps en vert montre que l'ajout de ces coupes peut aussi aider à la résolution d'une instance dont l'optimum n'était jamais trouvé jusque-là.

C'est un problème délicat que de choisir les bonnes coupe à ajouter, il faudrait pouvoir avoir un outil de mesure afin de savoir quelle coupe serait utile à la résolution et quelle coupe serait nuisible. Dans notre cas nous n'avons pas de tels outils, il est donc choisi de laisser de côté l'ajout des coupes sur les coûts réduits.

	Prétraitement et recherche locale puis (MIP) avec coupes sur les coûts réduits													
n	%inst résolues	%inst non Schrage Opti	T_{min}	T_{moy}	T_{max}	T^{Pre}_{min}	T^{Pre}_{moy}	T^{Pre}_{max}	#Noeuds$_{min}$	#Noeuds$_{moy}$	#Noeuds$_{max}$	$\%VF_{Min}$	$\%VF_{Moy}$	$\%VF_{Max}$
100	100	26	0,02	0,09	0,49	0	0,01	0,03	0	7,14	300	0	15,64	43,44
400	100	28	0,15	1,28	6,89	0,04	0,15	0,44	0	37,62	501	0	12,33	72,18
700	100	20	0,65	3,94	14,74	0,12	0,56	2,29	0	5,80	290	0	14,65	46,22
1000	100	32	0,98	13,53	43,21	0,20	1,44	4,78	0	5,34	217	0	9,89	54,49
1300	100	26	1,94	28,43	90,81	0,38	3,12	13,98	0	5,80	200	0	16,07	49,71
MOY	**100**	**26,4**	**0,75**	**9,45**	**31,23**	**0,15**	**1,06**	**4,3**	**0**	**12,34**	**301,6**	**0**	**13,72**	**53,21**

TABLE 5.9 – Résultats sur les instances faciles pour prétraitement avec recherche locale puis MIP avec coupes sur les coûts réduits

	Prétraitement et recherche locale puis (MIP) avec coupes sur les coûts réduits			
n	T^{Pre}	T	#Nœuds	% VF
100	0,02	900,14	1959704	0
100	0,03	14,40	23723	13,74
100	0,03	900,11	1742573	8,30
100	0,02	900,10	3141422	13,98
100	0,03	1,42	2847	18,40
100	0,02	900,08	2198602	2,13
200	0,03	900,21	2110719	30,96
300	0,23	900,96	177288	7,89
300	0,24	27,41	10839	1,32
300	0,12	900,70	188695	14,67
300	0,24	901,06	910827	48,77
400	0,31	902,31	370142	56,97
400	0,40	901,67	96658	0,45
400	0,54	901,77	121354	0,67
500	0,78	902,95	127561	8,60
600	0,60	906,37	41175	0,20
600	0,59	906,51	32822	0
600	1,78	135,81	10317	1,74
700	0,93	906,59	300656	75,14
700	0,49	905,99	477516	63,69
700	0,16	903,31	85412	0,41
700	1,54	909,86	13532	0,20
700	1,30	24,44	1881	42,89
800	2,44	916,25	15122	0,75
800	0,46	902,36	395348	42,81
800	2,04	678,76	120806	3,38
900	2,26	914,28	17757	5,61
900	2,37	49,98	712	47,83
900	1,30	690,19	253637	69,05
900	2,52	914,07	37058	6,53
900	1,68	914,51	49258	9,36
1000	3,23	922,63	12304	3,44
MOY	**0,90**	**729,91**	**470258,34**	**18,75**

TABLE 5.10 – Résultats sur les instances difficiles pour prétraitement avec recherche locale puis MIP avec les coupes sur les coûts réduits

5.5 Résultats avec coupe logique

L'étude des tableaux de résultats sur la version du prétraitement avec l'ajout des coupes logiques 5.11 avec le tableau classique 5.1 sur les instances faciles ne permet pas de faire de nombreuses conclusions. Le même problème qu'avec l'ajout des coupes sur les coûts réduits apparaît, le tableau de résultat 5.11 est incomplet puisque les problèmes de débordement mémoire surviennent à partir d'une taille d'instances de 1000. Les 3 lignes de résultats sont relativement similaires à la version du prétraitement avec recherche locale, il faut donc étudier les instances difficiles pour avoir un autre regard sur l'impact de ces coupes.

Les deux lignes avec le temps en vert du tableau 5.12 sur le prétraitement avec coupes logiques sur les instances dures sont intéressantes. En effet, ces deux instances sont résolues à l'optimum alors qu'elles ne l'étaient pas dans la version du prétraitement avec recherche locale (tableau 5.8). La deuxième ligne verte n'était pas non plus résolue à l'optimum dans la version du prétraitement seul alors que dans la version MIP seule l'optimum est trouvé. Cette ligne caractérise le fait que la solution donnée par le LP peut être très éloignée de la solution optimale du MIP. Cela est dû à la nature même du LP qui autorise un travail à appartenir à plusieurs pyramides, l'ajout des coupes logiques permet de limiter cet effet et de rapprocher la solution du LP de la solution du MIP. Il n'y a pas d'instances dont l'optimum n'est pas trouvé à cause de l'ajout des coupes logiques, mais certaines lignes ont des temps dégradés par rapport à la solution du prétraitement avec recherche locale.

Contrairement aux coupes sur les coûts réduits, il est possible de savoir si une coupe est utile avant de l'ajouter au modèle. En effet, elles sont ajoutées au LP et non au MIP. Il est donc possible de tester si une coupe viole la solution du LP après sa résolution et ainsi ajouter uniquement les coupes logiques utiles au modèle. Ce principe est appelé algorithme des plans coupants. Cette solution permettrait de pallier au problème de mémoire lors de l'ajout des coupes.

	Prétraitement, recherche locale et coupes logiques puis (MIP)													
n	%inst résolues	%inst non Schrage Opti	T_{min}	T_{moy}	T_{max}	T^{Pre}_{min}	T^{Pre}_{moy}	T^{Pre}_{max}	#Noeuds$_{min}$	#Noeuds$_{moy}$	#Noeuds$_{max}$	$\%VF_{Min}$	$\%VF_{Moy}$	$\%VF_{Max}$
100	100	26	0,02	0,09	0,37	0	0,01	0,03	0	7,14	300	0	15,64	43,44
400	100	26	0,18	1,68	7,58	0,04	0,31	2,12	0	34,44	938	0	12,33	72,18
700	100	20	0,80	7,03	38,86	0,12	2,06	20,57	0	0	0	0	14,65	46,22
MOY	**100**	**24**	**0,33**	**2,93**	**15,6**	**0,05**	**0,8**	**7,58**	**0**	**13,86**	**412,67**	**0**	**14,21**	**53,95**

TABLE 5.11 – Résultats sur les instances faciles pour prétraitement avec recherche locale et coupes logiques puis MIP

	Prétraitement, recherche locale et coupes logiques puis (MIP)			
n	T^{Pre}	T	#Noeuds	% VF
100	0,02	899,99	1965339	0
100	0,03	14,41	23723	13,74
100	0,03	899,99	1772494	8,30
100	0,02	900,10	3130204	13,98
100	0,02	1,44	2847	18,40
100	0,03	900,13	1830366	2,13
200	0,05	900,26	2126489	30,96
300	0,31	901,24	315758	7,89
300	0,39	800,08	462008	1,32
300	0,27	901,02	99800	14,67
300	0,33	901,40	967546	48,77
400	0,48	785,02	502684	56,97
400	1,46	903,25	86690	0,45
400	2,59	904,37	92535	0,67
500	1,12	904,08	103122	8,60
600	4,84	911,82	41738	0,20
600	3,95	910,97	32581	0
600	27,30	443,62	39105	1,74
700	2,83	910,87	375940	75,14
700	1,25	848,27	336621	63,69
700	0,22	903,56	85909	0,41
700	30,34	941,28	28117	0,20
700	7,09	27,94	475	42,89
800	34,32	952,47	15118	0,75
800	1,19	187,30	107087	42,81
800	36,18	277	34272	3,38
900	10,15	926,04	11734	5,61
900	5,19	666,41	124437	47,83
900	11,93	18,24	0	69,05
900	19,66	935,95	31624	6,53
900	15,20	931,96	48764	9,36
1000	37,24	958,83	17877	3,44
MOY	**8,00**	**727,17**	**462906,38**	**18,75**

TABLE 5.12 – Résultats sur les instances difficiles pour prétraitement avec recherche locale et coupes logiques puis MIP

Déroulement du projet

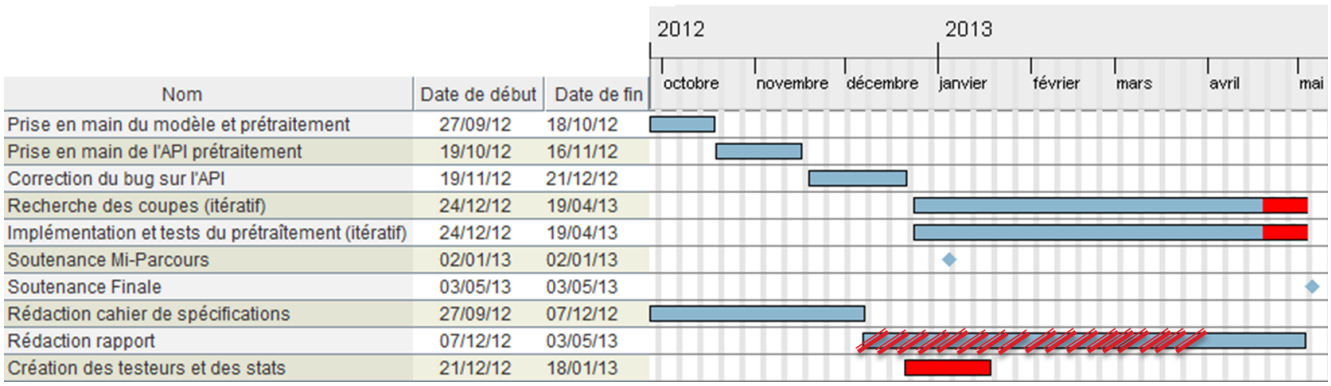
Il a été décidé au début du projet de ne pas mettre en place de système de versionnement afin de ne pas perdre de temps sur le temps de développement et parce que la nature du projet n'imposait pas un tel fonctionnement. Néanmoins, des sauvegardes ont quand même été effectuées par archives créées et sauvegardées à la main.

Pendant toute la durée de ce projet, il a été important de conserver un suivi élève/encadrant régulier. En effet, il s'agit d'un projet de recherche dont les décisions à prendre sont dépendantes des résultats obtenus. Il a donc été important d'avoir un contact régulier afin de discuter des résultats et des démarches à suivre pour chercher à améliorer ces résultats. Cette collaboration s'est principalement faite par entrevues et échange de mails.

Dans le cahier de spécifications, il a été présenté un diagramme prévisionnel, ce diagramme a été repris avec quelques modifications pour former le diagramme effectif du projet. Ce diagramme est présent en figure 6.1. Les différences notables sur ce diagramme effectif par rapport à son homologue prévisionnel sont présentes en rouge. Une tâche est apparue qui avait été omise dans l'analyse initiale du projet, la création des différents testeurs et la mise en place des statistiques. Cette tâche a pris plus de temps puisqu'il a été difficile de trouver une génération d'instances difficiles, plusieurs générations ont été mises en place pour arriver à ce but (voir partie sur la génération d'instances 4.6).

La deuxième différence notable est l'extension des tâches de recherche et développement des coupes. Cela peut s'expliquer par le fait qu'il est très difficile de faire une bonne estimation de temps de développement restant pour un projet de recherche. Les résultats ne sont pas connus à l'avance et peuvent mettre en avant de nouveaux problèmes ou de nouvelles pistes qui demandent un nouveau développement. Il est important de noter que ces tâches de recherche et implémentation des coupes incluent la recherche locale, il ne s'agit pas d'une coupe en soi mais il n'était pas prévu initialement de chercher à améliorer la borne supérieure.

Enfin la dernière différence est la rédaction du rapport qui a été commencée plus tard que prévu initialement. Il avait été prévu de faire la rédaction du rapport dès que possible en parallèle des tâches de développement, mais il est difficile de commencer une rédaction finale avant la fin du projet.



Nom	Date de début	Date de fin
Prise en main du modèle et prétraitement	27/09/12	18/10/12
Prise en main de l'API prétraitement	19/10/12	16/11/12
Correction du bug sur l'API	19/11/12	21/12/12
Recherche des coupes (itératif)	24/12/12	19/04/13
Implémentation et tests du prétraitement (itératif)	24/12/12	19/04/13
Soutenance Mi-Parcours	02/01/13	02/01/13
Soutenance Finale	03/05/13	03/05/13
Rédaction cahier de spécifications	27/09/12	07/12/12
Rédaction rapport	07/12/12	03/05/13
Création des testeurs et des stats	21/12/12	18/01/13

FIGURE 6.1 – Diagramme de Gantt effectif

Perspectives

Après analyse des différents résultats présentés dans la partie 5, il est possible de tirer quelques conclusions. Le prétraitement peut présenter un intérêt pour le problème du $1|r_i|L_{max}$ comme le montrent les résultats du prétraitement avec comme borne supérieure la solution optimale (partie 5.2). De nombreuses variables sont fixées et la résolution du LP permet de prouver l'optimum dans de nombreux cas, c'est peut-être ce deuxième point qui serait la plus grande force de cette méthode qui n'est pas directement liée au prétraitement. Afin de se rapprocher de ces résultats idéaux, la recherche locale a été mise en place, mais pourrait être améliorée pour être moins coûteuse en temps de calcul. Il reste aussi la possibilité d'utiliser une borne supérieure différente et plus efficace.

Les résultats sur le prétraitement avec l'ajout des coupes sur les coûts réduits mettent en avant la non-viabilité de ces coupes. Leur ajout surcharge trop le modèle et il n'est pas possible de savoir à l'avance quelles coupes pourraient être intéressantes à ajouter. Cette solution doit être abandonnée.

Les résultats avec les coupes logiques peuvent être intéressants, cependant le même problème de surcharge du modèle apparait. Mais contrairement aux coupes sur les coûts réduits, il est possible de vérifier si une coupe est utile avant de l'ajouter au modèle. En utilisant un algorithme de plans coupants, il serait possible d'obtenir d'autres résultats plus intéressants avec ces coupes. Cette solution n'est pas encore mise en place à l'heure où ces lignes sont écrites.

Ce sont donc 3 pistes de progression qui sont dégagées de par l'étude des résultats présentés dans ce rapport. La première serait l'amélioration de la borne supérieure, soit par amélioration de la recherche locale soit par utilisation d'une autre heuristique. La deuxième serait l'implémentation d'un algorithme des plans coupants pour l'ajout des coupes logiques. La troisième serait la recherche de nouvelles coupes.

Pendant ce projet, une base d'instances très difficiles pour la résolution du modèle mathématique a été construite. Vincent T'kindt a également réussi à dégager des instances difficiles pour la PSE de Carlier [3]. Il serait intéressant de tester ces bases d'instances sur les deux solutions opposées afin d'en voir les résultats.

Cyril Briand a aussi proposé une autre avancée, modifié le modèle mathématique afin d'empêcher l'appartenance d'un travail à plusieurs pyramides. Cela permettrait notamment d'empêcher la solution du LP d'être trop éloignée de la solution du MIP.

Conclusion

Ce projet m'a permis d'avoir une première expérience du travail d'un chercheur. Je suis particulièrement intéressé par cette méthode de travail c'est pourquoi cette expérience m'a été particulièrement enrichissante. La collaboration avec Vincent T'kindt s'est très bien déroulée et m'a beaucoup apporté, c'est pourquoi je tenais à le remercier ici pour sa disponibilité et sa patience. J'espère que ma contribution sur ce projet pourra mener à des résultats publiables dans le future. J'ai fait de mon mieux pour donner les informations nécessaires à la reprise de mon travail dans ce rapport et dans le code que j'ai été amené à produire.

Bibliographie

[1] Briand C., Ourari S., Bouzouia B. "An efficient ILP formulation for the single machine scheduling problem". *submitted* to *RAIRO - Operations Research*, 2008.

[2] Della Croce F., T'kindt V. "Improving the preemptive bound for the single machine min-max lateness problem subject to release times". *Operations Research Letters*, 38(10), 2010.

[3] Carlier, J. "The one-machine sequencing problem". *European Journal of Operational Research*, 11, 42-47, 1982.

Annexes

A.1 Papier de présentation du modèle mathématique

An efficient ILP formulation for the single machine scheduling problem

Cyril Briand [a,b,*], Samia Ourari [a,b,c], Brahim Bouzouia [c]

[a] *CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France*

[b] *Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France*

[c] *CDTA, lotissement du 20 août 1956, Baba Hassen, Alger, Algeria*

Abstract

This paper considers the problem of scheduling n jobs on a single machine. A fixed processing time and an execution interval are associated with each job. Preemption is not allowed. On the basis of analytical and numerical dominance conditions, an efficient integer linear programming formulation is proposed for this problem, aiming at minimizing the maximum lateness ($L_{\max}$). Experiments have been performed by means of a commercial solver that show that this formulation is effective on large problem instances. A comparison with the branch-and-bound procedure of Carlier is provided.

Key words: Single machine scheduling, integer linear programming, dominance conditions.

1 INTRODUCTION

A single machine scheduling problem (SMSP) consists of a set V of n jobs to be sequenced on a single disjunctive resource. The interval $[r_j, d_j]$ defines the execution window of each job j, where r_j is the release date of j and d_j, its due-date. The processing time p_j of j is known and preemption is not allowed. A feasible job sequence has to be found so that, s_j being the starting time of job j, $s_j \geq r_j$ and $s_j + p_j \leq d_j$.

* Corresponding author.

Email addresses: briand@laas.fr (Cyril Briand), sourari@cdta.dz (Samia Ourari), bbouzouia@cdta.dz (Brahim Bouzouia).

Seeking whether it exists a feasible sequence (i.e., all jobs meet their due dates) is NP-complete [6] and consequently, finding a sequence that minimizes the maximum lateness (problem refers to as $1|r_i|L_{\max}$) is NP-hard. For this problem, the well-known branch-and-bound procedure of Carlier [2] can be used for solving problems having more than one thousand jobs. Nevertheless, even if the Carlier's procedure provides good performance, it is still interesting to design other exact approaches that can be used in a more generic framework such as integer linear programming (ILP).

This paper presents an original ILP formulation for the SMSP that is based on analytical and numerical dominance conditions. The paper first presents a dominance theorem that is the foundation of the work. Then ILP models are progressively issued, starting from particular SMSP cases up to the general case. The last section is devoted to the experimentations.

2 A GENERAL DOMINANCE THEOREM FOR THE SMSP

In the sequel, some analytical dominance conditions are used for the SMSP. They have been originally proposed in the early eighties by Erschler et al. [4] within a theorem. This theorem uses the notions of a *top* and a *pyramid*, which are defined below. It defines a set S_{dom} of dominant job sequences for the SMSP with regard to the feasibility problem. Let us recall that a job sequence σ_1 dominates another job sequence σ_2 if σ_2 feasible $\Rightarrow \sigma_1$ is feasible. By extension, a set of job sequences S_{dom} is said dominant if, for any job sequence $\sigma_2 \notin S_{\text{dom}}$, it exists $\sigma_1 \in S_{\text{dom}}$ such that σ_2 feasible $\Rightarrow \sigma_1$ is feasible. When searching a feasible job sequence, only the set of dominant sequences need to be explored since, if there does not exist any feasible sequence in the dominant set, then it can be asserted that the original problem does not admit any feasible solution. This leads to a significant reduction of the search space.

Definition 1 *A job $t \in V$ is a top if there does not exist any other job $i \in V$ such that $r_i > r_t \wedge d_i < d_t$ (i.e., the execution window of a top job does not strictly include any other execution window).*

The top jobs are indexed in ascending order with respect to their release dates or, in case of a tie, in ascending order with respect to their due dates. When both their release dates and due dates are equal, they can be indexed in an arbitrary order. Thus, if t_a and t_b are two top jobs then $a < b$ if and only if $(r_{t_a} \leq r_{t_b}) \wedge (d_{t_a} \leq d_{t_b})$. Let m be the total number of top jobs.

Definition 2 *A pyramid P_k related to a top job t_k is the set of jobs $i \in V$ such that $r_i < r_{t_k} \wedge d_i > d_{t_k}$ (i.e., the set of jobs for which their execution window strictly includes the execution window of the top job).*

Considering the previous definition, let us observe that a non-top job can belong to several pyramids. Let $u(j)$ ($v(j)$ respectively) denotes the index of the first pyramid (the last pyramid resp.) to which Job j can be assigned.

The following theorem, proved in [4], can now be stated.

Theorem 1 *The set of sequences in the form* $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \prec \ldots \prec \alpha_k \prec t_k \prec \beta_k \prec \ldots \prec \alpha_m \prec t_m \prec \beta_m$ *is dominant for finding a feasible sequence, where:*

- t_k *is the top job of the pyramid* P_k, $\forall k = 1 \ldots m$;
- α_k *and* β_k *are two job subsequences located at the left and the right of top job* t_k *respectively, such that, for any* k*, the jobs of* α_k *are ordered by increasing release dates and the jobs of* β_k *are ordered by increasing due dates ;*
- *any non-top job* j *is assigned to one sequence* α_k *or* β_k*, with* $u(j) \leq k \leq v(j)$.

In the previous theorem, it must be pointed out that the numerical values of the processing times p_j as well as those of r_j and d_j are not used. Only the relative order of the release and due dates is considered, hence the genericity of the result. In order to illustrate the theorem, let us consider an instance of seven jobs, so that the relative order among the release dates r_j and the due dates d_j of the jobs is $r_6 < r_1 < r_3 < r_2 < r_4 < d_2 < d_3 < d_4 < (r_5 = r_7) < d_6 < d_5 < d_1 < d_7$. This instance has four top jobs: $t_1 = 2$, $t_2 = 4$, $t_3 = 5$ and $t_4 = 7$, which characterize four pyramids: $P_1 = \{1, 3, 6\}$, $P_2 = \{1, 6\}$, $P_3 = \{1\}$ and $P_4 = \emptyset$ (in accordance with the definition, a top job does not belong to the pyramid it characterizes). According to Theorem 1, whatever the real values of r_i and d_i (provided they are compatible with the previous interval structure), the dominant sequences are in the form $\alpha_1 \prec 2 \prec \beta_1 \prec \alpha_2 \prec 4 \prec \beta_2 \prec \alpha_3 \prec 5 \prec \beta_3 \prec 7$, where jobs belonging to subsequence α_k (β_k respectively) are sequenced with respect to the increasing order of their r_j (d_j respectively). Thus, 24 dominant sequences (out of $7! = 5040$) are characterized.

3 THE MONO-PYRAMIDAL SMSP

In this section, the focus is on the mono-pyramidal SMSP: it is assumed that the problem has a single top job $t \in V$ (hence a single pyramid P_t). With respect to Theorem 1, the set S_{dom} of dominant job sequences is in the form $\alpha \prec t \prec \beta$, where jobs belonging to subsequence α are sequenced with respect to the increasing order of their r_j, and jobs belonging to β, with respect to the increasing order of their d_j. Hence, 2^n sequences need to be explored for checking the feasibility (instead of $(n + 1)!$). Below, another numerical dominance theorem is stated that allows to compare the 2^n sequences each other. It is based on a necessary and sufficient condition of feasibility.

As stated in [3] regarding the general SMSP, let us first recall that a job sequence σ is feasible iff:

$$p_i + \sum_{i \prec k \prec j} p_k + p_j \leq d_j - r_i \ ,\forall (i,j) \in \sigma \text{ such that } i \prec j \tag{1}$$

The previous necessary and sufficient condition of feasibility can be drastically simplified for the mono-pyramidal SMSP. For matter of notation simplification, given a dominant sequence in the form $\alpha \prec t \prec \beta$, the following notations are set:

- $R_a = r_a + p_a + \sum_{k \in \alpha, a \prec k}(p_k), \forall a \in \alpha$;
- $D_b = d_b - p_b - \sum_{k \in \beta, k \prec b}(p_k), \forall b \in \beta$;
- $R^{\max} = \max\left[r_t, \max_{a \in \alpha}(R_a)\right]$;
- $D^{\min} = \min\left[d_t, \min_{b \in \beta}(D_b)\right]$.

Clearly, $R^{\max}$ corresponds to the earliest starting time of Top t, further refers to as est_t. Symmetrically, $D^{\min}$ corresponds to the latest completion time of Top t, further refers to as lft_t.

Now the following theorem can be stated.

Theorem 2 *P being a pyramid of top t, any dominant sequence in the form $\alpha \prec t \prec \beta$ is admissible if and only if $D^{\min} - R^{\max} - p_t \geq 0$.*

Proof Let us first prove that $D^{\min} - R^{\max} - p_t \geq 0$ implies the feasibility of $\alpha \prec t \prec \beta$, i.e., the condition 1 is verified. Condition 1 can be simplified since the case where i and j ($i \prec j$) both belong to α can be ignored. Indeed, $D^{\min} - R^{\max} - p_t \geq 0$ implies that $d_t - R_i - p_t \geq 0 \ \forall i \in \alpha$ and, because $d_j > d_t \ \forall j$, then $d_j - R_i - p_t > 0$. By replacing R_i by its value it comes that $d_j - r_i > p_t + \sum_{j \prec k \prec t} p_k + p_j + \sum_{i \prec k \prec j} p_k + p_i > p_i + \sum_{i \prec k \prec j} p_k + p_j$. Therefore, if Condition 1 holds between any job $i \in \alpha$ and the top job t, then it also holds between i and any $j \in \alpha$ such that $i \prec j$. A similar reasoning can be made for discarded the case where i and j both belongs to β. Indeed, if Condition 1 holds between any $j \in \beta$ and the top job t, then it also hods between any $i \in \beta$ and j, provided that $i \prec j$. So we only need to prove that $D_b - R_a - p_t \geq 0$, $\forall a \in (\alpha \prec t)$ and $\forall b \in (t \prec \beta)$. This is obvious since $D_b - R_a \geq D^{\min} - R^{\max}$ by definition and $D^{\min} - R^{\max} - p_t \geq 0$ by assumption.
Now, we are going to prove that the feasibility of $\alpha \prec t \prec \beta$ implies that $D^{\min} - R^{\max} - p_t \geq 0$. It is also obvious since if $\alpha \prec t \prec \beta$ is feasible then, from Condition 1, $D_b - R_a - p_t \geq 0$, $\forall (a,b) \in \alpha \times \beta$ and $d_t - r_t - p_t \geq 0$. Of course, this inequality needs to be verified in the worst case, i.e., $\min\left(d_t, \min_{b \in \beta} D_b\right) - \max\left(r_t, \max_{a \in \alpha} R_a\right) - p_t \geq 0$. Hence $D^{\min} - R^{\max} - p_t \geq 0$. $\square$

The fact that Theorem 2 permits to numerically compare sequences each other is helpful since it immediately involves the following dominance condition.

Corollary 1 *P being a pyramid, job t being its top job, and σ_1 and σ_2 being two dominant sequences in the form $\alpha_1 \prec t \prec \beta_1$ and $\alpha_2 \prec t \prec \beta_2$ respectively, if $D_1^{\min} - R_1^{\max} \geq D_2^{\min} - R_2^{\max}$, then σ_1 dominates σ_2 with regard to the feasibility problem.*

Proof By definition, stating that σ_1 dominates σ_2 requires to prove that the feasibility of σ_2 implies the one of σ_1. Now we know that σ_2 is feasible iff (see Theorem 2) $D_2^{\min} - R_2^{\max} \geq p_t$, therefore, under the assumption that $D_1^{\min} - R_1^{\max} \geq D_2^{\min} - R_2^{\max}$, it can be deduced that $D_1^{\min} - R_1^{\max} \geq p_t$, so σ_1 is necessarily also feasible. □

Consequently, the problem of searching a feasible job sequence in a mono-pyramidal SMSP can be reformulated as an optimization problem where the objective function is to maximize $D^{\min} - R^{\max}$. An integer linear program (ILP1) is given below for modelling such a problem.

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

$$\text{s.t.} \begin{cases} R \geq r_t & (3.1) \\ D \leq d_t & (3.2) \\ R \geq r_i + \sum_{\{j \in V \setminus \{t\} | r_j \geq r_i\}} p_j x_j^+ \ , \forall i \in V & (3.3) \\ D \leq d_i - \sum_{\{j \in V \setminus \{t\} | d_j \leq d_i\}} p_j x_j^- \ , \forall i \in V & (3.4) \\ x_i^- + x_i^+ = 1 \ , \forall i \in V & (3.5) \\ x_i^- \ , \ x_i^+ \in \{0,1\} \ , \forall i \in V & \\ D \ , \ R \in \mathbb{Z} & \end{cases}$$

In the above ILP1 formulation, every job $j \in V \setminus \{t\}$ has to be sequenced either at the left side (i.e., $x_j^+ = 1$ and $j \in \alpha$) or at the right side (i.e., $x_j^- = 1$ and $j \in \beta$) of Top t, but not both (i.e., $x_i^- + x_i^+ = 1$). If $z = D - R$ is maximized, while respecting the constraints, then, from Corollary 1, the obtained sequence dominates all the others. Moreover, if $z^* \geq p_t$ then, from Theorem 2, the sequence is feasible. If $z^* < p_t$ then it can be asserted that it does not exist any feasible sequence for the considered problem.

We highlight that, since $x_i^- + x_i^+ = 1$, the variables x_i^- (or x_i^+) are redundant and can be removed from the ILP. Nevertheless, we prefer to keep them for enhancing the clarity of the model.

Another important remark concerns the constraints (3.3). According to Theorem 2, they only need to be satisfied for the jobs of α. What happens now if a job x is in β? Could the constraint of type (3.3) related to x be unnecessarily restrictive? We prove below that, in such a case, this constraint cannot be active since it always exists a job $y \in (\alpha \prec t)$ inducing a more restrictive constraint, i.e. :

$$r_y + \sum_{\{j \in V \setminus \{t\} | r_j \geq r_y\}} p_j x_j^+ \geq r_x + \sum_{\{j \in V \setminus \{t\} | r_j \geq r_x\}} p_j x_j^+$$

Indeed, let y be the job in $(\alpha \prec t)$ having the lowest release date r_y, provided that $r_y \geq r_x$. Since we obviously have $\sum_{\{j \in V \setminus \{t\} | r_j \geq r_y\}} p_j x_j^+ = \sum_{\{j \in V \setminus \{t\} | r_j \geq r_x\}} p_j x_j^+$, the previous relation necessarily holds.

A similar reasoning can be made for the constraints (3.4) since, if a job x is in α, then, considering the job $y \in (t \prec \beta)$ having the greatest due date d_y , provided that $d_y \leq d_x$, on can proved that :

$$d_y - \sum_{\{j \in V \setminus \{t\} | d_j \leq d_y\}} p_j x_j^- \leq d_x - \sum_{\{j \in V \setminus \{t\} | d_j \leq d_x\}} p_j x_j^-$$

Now, let us show that the optimal solution of ILP1 is also optimal with regard to the maximum lateness minimization.

Theorem 3 *The optimal ILP1 solution minimizes the maximum lateness and* $L^*_{\max} = p_t - z^*$.

Proof Let $\alpha \prec t \prec \beta$ be a feasible solution of ILP1 and $z = D - R$ be the corresponding objective value. Let us determine the lateness of a job $a \in \alpha$. By definition, when all the jobs are scheduled at the latest, the maximum lateness L_a of a equals $L_a = r_a - lst_a$, where lst_a refers to as the latest starting time of a. Obviously, there exists a job $u \in \beta$ such that $D = d_u - \sum_{\{k \in \beta | d_k \leq d_u\}} p_k$. Therefore, the latest starting time of t is $lst_t = D - p_t$ and the latest starting time of any job $a \in \alpha$ is $lst_a = D - p_t - \sum_{\{k \in \alpha | a \prec k\}} p_k - p_a$. Then it is easy to see that the job $v \in \alpha$, which has the maximum lateness (i.e., $L_v = \max_{a \in \alpha}(L_a)$) is precisely the one that gives to R its value. Hence $L_v = R - D + p_t$.
Let us now determine the lateness L_b of a job $b \in \beta$. Symmetrically to the previous reasoning, one can determine that the earliest finishing time of b is $eft_b = R + p_t + \sum_{\{k \in \beta | k \prec b\}} p_k + p_b$. Since, when all the jobs are scheduled at the earliest, $L_b = eft_b - d_b$, the job $u \in \beta$ which has the maximum lateness (i.e., $L_u = \max_{b \in \beta}(L_b)$)) is precisely the one that gives to D its value. Hence $L_u = R - D + p_t$. In conclusion, there is a strict equivalence between the maximum

lateness minimization and the maximization of the objective function z since $L_{\max} = p_t - z$. $\square$

4 THE INDEPENDENT MULTI-PYRAMIDAL SMSP

Before considering the general SMSP, let us focus on the particular SMSP characterized by m top jobs, hence m pyramids, such that any non-top job only belongs to a single pyramid (i.e., the pyramids are independent). With respect to Theorem 1, any dominant job sequence is in the form $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \prec \ldots \prec \alpha_m \prec t_m \prec \beta_m$, where t_k is the top job of the pyramid P_k, $\forall k = 1 \ldots m$ and the jobs of α_k and β_k only belong to pyramid P_k.

Therefore, the problem of finding the most dominant job sequence for the independent multi-pyramidal SMSP can be decomposed into m interdependent mono-pyramidal subproblems. For each of them, the most dominant job sequence $\alpha_k \prec t_k \prec \beta_k$ has to be found. In other words, if R_k refers to as the earliest starting time of Top t_k and D_k to as the latest completion time of Top t_k, we have to maximize $D_k - R_k - p_{t_k}$.

These mono-pyramidal subproblems are interdependent because the sequences $\alpha_k \prec t_k \prec \beta_k$ should not overlap. This requirement can easily be achieved by setting, for any job $j \in P_k$, the constraints: $r_j = \max(r_j, eft_{k-1})$ and $d_j = \min(d_j, lst_{k+1})$, where eft_{k-1} (lst_{k+1} resp.) refers to as the earliest completion time of Subsequence β_{k-1} (the latest starting time of Subsequence α_{k+1} resp.).

The value of eft_{k-1} corresponds to the earliest completion time of Top t_{k-1} (i.e., $R_{k-1} + p_{t_{k-1}}$), plus the sum of the processing times of the jobs of β_{k-1}, i.e. $eft_{k-1} = R_{k-1} + p_{t_{k-1}} + \sum_{\{j \in P_{k-1}\}}(p_j x_j^-)$. Similarly, the value of lst_{k+1} corresponds to the latest starting time of Top t_{k+1} (i.e. $D_{k+1} - pt_{k+1}$), minus the sum of the processing times of the jobs of α_{k+1}, i.e. $lst_{k+1} = D_{k+1} - p_{t_{k+1}} - \sum_{\{j \in P_{k+1}\}}(p_j x_j^+)$. Therefore, eft_{k-1} and lst_{k+1} only depends on the ILP variables.

Consequently, the problem of searching a feasible job sequence in a multi-pyramidal independent SMSP can be formulated by the following ILP (ILP2):

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

$$\text{s.t.}\quad \begin{cases}
R_k \geq r_{t_k} & , \ \forall k \in [1\ m] & (4.1)\\
D_k \leq d_{t_k} & , \ \forall k \in [1\ m] & (4.2)\\
R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_j^+ & , \ \forall i \in P_k & (4.3)\\
D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_j^- & , \ \forall i \in P_k & (4.4)\\
R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x_j^- + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x_j^+ & , \ \forall k \in [2\ m] & (4.5)\\
D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x_j^+ - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x_j^- & , \ \forall k \in [1\ (m-1)] & (4.6)\\
x_i^- + x_i^+ = 1 & , \ \forall i \in P_k & (4.7)\\
x_i^- \ , \ x_i^+ \in \{0,1\} & , \ \forall i \in P_k & \\
D_k \ , \ R_k \in \mathbb{Z} & , \ \forall k \in [1\ m] &
\end{cases}$$

In the previous formulation, similarly with ILP1, the binary variable x_i^+ (x_i^- resp.) is set to 1 if Job i, which only belongs to pyramid P_k, is sequenced in α_k (in β_k resp.), provided that it cannot be sequenced both in α_k and β_k (i.e., $x_i^- + x_i^+ = 1$). The constraints (4.1)-(4.4) are identical with the constraints (3.1)-(3.4) of ILP1. The constraints (4.5) and (4.6) impose that the subsequences $\alpha_k \prec t_k \prec \beta_k$ do not overlap. Under these constraints, one can determine for each pyramid P_k, the maximum lateness, which equals $R_k - D_k + p_{t_k}$. Hence, maximizing z is equivalent to minimize the global maximum lateness.

5 THE GENERAL SMSP

Now, the general SMSP is considered. Here, a non-top job j can belong to several pyramids, that is the only difference with the multi-pyramidal independent SMSP. Let us note $u(j)$ ($v(j)$ resp.) the index of the first pyramid (the last pyramid resp.) to which the job j belongs. With respect to Theorem 1, any dominant sequence is in the form $\sigma = \alpha_1 \prec t_1 \prec \beta_1 \prec \ldots \prec \alpha_m \prec t_m \prec \beta_m$, where t_k is the top job of the pyramid P_k, $\forall k = 1 \ldots m$, and any non-top job j can be sequenced either in α_k or β_k, for any k such that $u(j) \leq k \leq v(j)$.

It is easy to see, once the assignments of the non-top jobs to their pyramid done, that the problem turns into a multi-pyramidal independent problem (that can be solved using the ILP2 formulation). Consequently, a general formulation can be obtained by simply adding binary variables to the ILP2 formulation, so that each non-top job can only be assigned to a single pyramid

P_k. The following general formulation (ILP3) is then obtained:

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

$$\text{s.t.}\quad \begin{cases}
R_k \geq r_{t_k} & , \ \forall k \in [1\ m] & (5.1)\\
D_k \leq d_{t_k} & , \ \forall k \in [1\ m] & (5.2)\\
R_k \geq r_i + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \ \forall i | k = u(i) & (5.3)\\
D_k \leq d_i - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \ \forall i | k = v(i) & (5.4)\\
R_k \geq R_{k-1} + \sum_{\{j \in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j \in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \ \forall k \in [2\ m] & (5.5)\\
D_k \leq D_{k+1} - \sum_{\{j \in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j \in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \ \forall k \in [1\ (m-1)] & (5.6)\\
\sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) = 1 & , \ \forall i & (5.7)\\
x^-_{ki}\ ,\ x^+_{ki} \in \{0,1\} & , \ \forall k \in [1\ m], \forall i \in P_k & \\
D_k\ ,\ R_k \in \mathbb{Z} & , \ \forall k \in [1\ m] &
\end{cases}$$

In ILP3, similarly with ILP2, the binary variable x^+_{ki} (x^-_{ki} resp.) is set to 1 if Job i, belonging to Pyramid P_k (i.e. $u(i) \leq k \leq v(i)$), is sequenced in α_k (in β_k resp.), provided that it cannot be sequenced both in α_k and β_k and it is assigned to a single pyramid (i.e., the constraint (5.7) is satisfied for i). The constraints (5.1)-(5.6) remain unchanged, when compared to ILP2.

Now let us give few comments concerning the number of constraints and variables that are involved. There is one constraint of type (5.1) and (5.2) for each top job. There is one constraint of type (5.3) and (5.4) for each non-top job and there are $(n-m)$ non-top jobs. There are $(m-1)$ constraints of type (5.5) and $(m-1)$ constraints of type (5.6). There is one constraint of type (5.7) for each non-top job. Therefore, the total number of constraint is $(m+3n-2)$. The number of binary variables is $(2\sum_{k=1}^{m}(n_k))$, where n_k is the number of non-top jobs that can be assigned to P_k. There are $(2m)$ integer variables but we notice that their value can be directly deduced from those of the binary variables.

6 NUMERICAL EXPERIMENTS

The ILP3 model has been implemented using the Ilog-concert library and tested on problem instances having from 100 up to 3100 jobs. For each problem instance, we both ran the ILP3 model and the Carlier's procedure, aiming at

comparing their computational time. All experiments have been processed on a 1.60 GHz Intel Xeon processor with 2 Gbytes of RAM. For each problem size, 50 problem instances have been considered.

To generate data for the single machine scheduling problem, we reuse the method introduced in [5]. The processing times of jobs correspond to uniform random variables between [1,50]. Each r_i is represented by an uniform random variable in the range $[0, \alpha \times \sum_{i=1}^{n} p_i]$, where α is a parameter allowing to adjust the distribution of r_i. Five values of α were considered: $\{0.2; 0.4; 0.6; 0.8; 1\}$. Similarly, each d_i is represented by an uniform random variable in the range $[(1-\beta) \times a \times \sum_{i=1}^{n} p_i, a \times \sum_{i=1}^{n} p_i]$.The parameter $a \in \{100\%, 110\%\}$ allows to adjust the maximal temporal margin associated to jobs and the parameter β controls the dispersion of d_i. Five values of β were considered $\beta \in \{0.2; 0.4; 0.6; 0.8; 1\}$. To ensure the coherence among the release date, the due date and the processing time of each job, any d_i smaller than $r_i + p_i$ has been updated to this value. For each problem size, combining each possible value of α, β and a, we obtain 50 problem instances.

Using those generating principles, we provide instances for single machine problems having 100, 400, 700, 1000, 1300, 1600, 1900, 2200, 2500, 2800 and 3100 jobs. For each problem size, Table 1 reports the following indicators :

- The min/mean/max cpu time for the Carlier's procedure (T_{Car});
- The min/mean/max cpu time for running ILP3 (T_{ILP});
- The percentage of times $T_{ILP} > T_{Car}$ ($\%W$);
- The number of Schrage-optimal instances ($\%S$);
- The min/mean/max cpu time for the Carlier's procedure only considering the hard instances (T_{Car}^{hard});
- The min/mean/max cpu time for running ILP3 only considering the hard instances (T_{ILP}^{hard});
- The percentage of times $T_{ILP}^{hard} > T_{Car}^{hard}$ ($\%W^{hard}$).

The branch-and-bound procedure of Carlier [2] intensively uses the Schrage's algorithm in order to build up, at each node of the search tree, a job sequence. The local optimality of this job sequence is easily checked using a sufficient condition of optimality. We remark that it sometimes happens that the first Schrage's job sequence (i.e., the one computed at the root node considering the initial instance data) is optimal. In this case, we say that the problem instance is Schrage-optimal and we know that the Carlier's procedure only develops a single node for returning the optimal $L_{\max}$ value. As displayed on Table 1, this situation occurs many time in our experiments since, when $n \geq 2200$, we observe that the generating procedure often fails to provide instances which are not Schrage-optimal (i.e., $\#S$ is close to 50).

This is why we decided to provide Cplex with the initial Schrage's sequence,

n	All instances								Hard instances							
	T_{Car}			T_{ILP}			$\%W$	$\#S$	T^{hard}_{Car}			T^{hard}_{ILP}			$\%W^{hard}$
	min	mean	max	min	mean	max			min	mean	max	min	mean	max	
100	0	0.02	1	0	0.04	1	4	16	0	0.06	1	0	0.06	1	2.9
400	0	0.3	2	0	0.22	1	16	38	0	1	2	0	0.33	1	8.3
700	0	1.4	6	0	1.24	8	40	35	1	3.66	6	0	1.33	3	0
1000	0	1.22	27	1	3.18	26	90	46	5	14.00	27	1	8.25	26	25
1300	0	2.46	20	2	4.82	12	80	41	4	9.22	20	3	5	9	22.2
1600	0	5.48	29	4	8.4	17	70	33	7	14.30	29	4	8.47	17	17.6
1900	0	3.42	40	4	15.1	44	96	44	11	18.66	40	10	22.5	44	66.7
2200	0	1.06	20	5	16.22	36	100	49	20	20	20	26	26	26	100
2500	0	1.9	31	7	21.42	40	100	47	17	23	31	30	33	37	100
2800	0	0.28	1	8	27.96	54	100	50	—	—	—	—	—	—	—
3100	0	0.4	1	11	35.32	71	100	50	—	—	—	—	—	—	—

Table 1
Experimentation report

intending to decrease the computational effort needed to solve ILP3. Nevertheless, we highlight that in the case where the problem instance is Schrage-optimal, Cplex has still to prove the optimality of the sequence using LP techniques (while the Carlier's procedure is able to immediately deduce the optimality). This observation explains the first part of Table 1. Indeed, T_{Car} progressively increases as n increases until $n = 1900$, which is the limit where $\#S$ becomes very close to 50, and then T_{Car} decreases when n increases. Regarding now T_{ILP}, we observe that the mean CPU time always increases as n increases, even when $\#S \cong 50$, since proving the optimality of the Schrage's sequence becomes more and more time expensive. Beyond this observation, we also observe that T_{ILP} is good for small instances (even better in average than T_{Car} when $n \leq 700$) and remains acceptable for largest ones.

In order to refine the analysis, we decided to discard the Schrage-optimal problem instances from our statistics, only focusing on the hardest ones (see second part of Table 1). This time, we observe that the ILP approach is less time expensive than the Carlier's procedure in most of the cases, at least until $n = 1600$. This seems to indicate that the ILP approach becomes competitive only when the problem instances are hard (i.e., the number of nodes that the Carliers's procedure develops is large), although for large n values, it is difficult to make an opinion due to the low number of hard instances that were generated.

Another limit of the ILP approach concerns the size of the problem instances that it is able to cope with. Indeed, even if Cplex succeeds in solving instances having more than 4000 jobs, we observe that when there are many top jobs in the instance, the number of binary variables increase drastically and Cplex may run out of memory. On our test platform, we observe that this situation

often occurs for instances of 4500 jobs having more than 300 top jobs (in this case, the corresponding reduced MIP has around 9000 rows and 700000 columns). Conversely, the Carlier's procedure is not affect by the number of non-top jobs and is able to solve instances with more than 10000 jobs without any problem.

7 CONCLUSION

This paper proposes an efficient ILP formulation for solving the general $1|r_j|L_{\max}$ problem. The experiments tend to demonstrate that our ILP approach is well suited to tackle medium-size instances, particularly when they are not Schrage-optimal (i.e., hard to solve). We also point out that, to the best of our knowledge, there does not exist any other ILP approach that can be compared with the clever branch-and-bound procedure of Carlier in term of efficiency, although we recognize that the Carlier's procedure remains the best-suited technique to solve efficiently $1|r_j|L_{\max}$ in most of the situations. Beyond these observations, we think that the interest of our ILP approach lies in its ability to be easily adapted for considering different objective function for the SMSP (e.g., the minimization of the number of late jobs) or, hopefully, for tackling more complex scheduling environment (e.g., parallel machines or job shop) with similar objective.

References

[1] Briand C., La H.T., Erschler J. "Une approche pour l'ordonnancement robuste de tâches sur une machine", *4ème Conférence Francophone de MOdélisation et SIMulation* (MOSIM'03), pp 205-211, Toulouse, France, 2003.

[2] Carlier, J., "The one-machine sequencing problem", 1982, *European Journal of Operational Research*, 11, 42-47.

[3] Erschler J., Roubellat, F., Vernhes J.-P, "Characterizing the set of feasible sequences for n jobs to be carried out on a single machine", *European Journal of Operational Research*, Vol. 4, pp189-194, 1980.

[4] Erschler, J., Fontan, G., Merce, C., Roubellat, F.,"A New Dominance Concept in Scheduling n Jobs on a Single Machine with Ready Times and Due Dates", *Operations Research*, Vol. 31, pp114-127, 1983.

[5] Hariri A.M., Potts C.N., "An algorithm for single machine sequencing with release dates to minimize total weighted completion time", *Discrete Applied Mathematics*, vol. 5, pp99-109, 1983.

[6] Lenstra J.K., Rinnooy Han A.H.G , Brucker P., "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, vol. 1, pp343-362,1977.

A.2 Démonstration de la coupe logique sur les travaux multi-pyramidaux

Logical cuts for the $1|r_j|L_{max}$ problem

1 Introduction

Let us first recall the mathematical formulation (IP) for the $1|r_j|L_{max}$ problem.

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

$$\text{s.t.}\ \begin{cases}
R_k \geq r_{t_k} & , \ \forall k \in [1\ m] & (5.1)\\
D_k \leq d_{t_k} & , \ \forall k \in [1\ m] & (5.2)\\
R_k \geq r_i + \sum_{\{j\in P_k | r_j \geq r_i\}} p_j x^+_{kj} & , \ \forall i | k = u(i) & (5.3)\\
D_k \leq d_i - \sum_{\{j\in P_k | d_j \leq d_i\}} p_j x^-_{kj} & , \ \forall i | k = v(i) & (5.4)\\
R_k \geq R_{k-1} + \sum_{\{j\in P_{k-1}\}} p_j x^-_{(k-1)j} + p_{t_{k-1}} & & \\
\qquad + \sum_{\{j\in P_k\}} p_j x^+_{kj} & , \ \forall k \in [2\ m] & (5.5)\\
D_k \leq D_{k+1} - \sum_{\{j\in P_{k+1}\}} p_j x^+_{(k+1)j} - p_{t_{k+1}} & & \\
\qquad - \sum_{\{j\in P_k\}} p_j x^-_{kj} & , \ \forall k \in [1\ (m-1)] & (5.6)\\
\sum_{k=u(i)}^{v(i)} (x^-_{ki} + x^+_{ki}) = 1 & , \ \forall i & (5.7)\\
x^-_{ki}\ ,\ x^+_{ki} \in \{0,1\} & , \ \forall k \in [1\ m], \forall i \in P_k & \\
D_k\ ,\ R_k \in \mathbb{Z} & , \ \forall k \in [1\ m] &
\end{cases}$$

Now, consider a job j belonging to several pyramids (see Figure 1), with $u(j)$ (resp. $v(j)$) the first (resp. last) pyramid including it. Besides, let b_j (resp. a_j) be the first pyramid before f (resp. after ℓ). Notice that b_j or a_j may not exist.

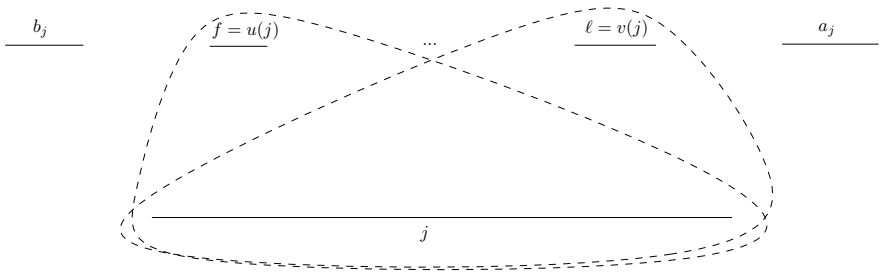


Figure 1: A multi-pyramidal job j

Let (LP) be the linear relaxation of (IP). The aim of logical cuts is to strenghen (LP) without loosing the optimal solution of (IP). To derive logical cuts from job j, we first

study the constraints from (LP) in which a variable x^+_{kj} or x^-_{kj} appears.

2 A reformulation of some constraints in (LP)

Consider a multi-pyramidal job j and the linear relaxation (LP). We make use of the following instrumental notations:

- $A^+_{kj} = \sum_{\{i \in P_k, i \neq j | r_i \geq r_j\}} p_i x^+_{ki}$,
- $B^-_{kj} = \sum_{\{i \in P_k, i \neq j | d_i \leq d_j\}} p_i x^-_{ki}$,
- $T^+_{kj} = \sum_{\{i \in P_k, i \neq j\}} p_i x^+_{ki}$,
- $T^-_{kj} = \sum_{\{i \in P_k, i \neq j\}} p_i x^-_{ki}$.

For a given multi-pyramidal job j and any pyramid $k = f, ...l$, we can rewrite constraints (5.3) to (5.6) as:

$$\text{(S)} \begin{cases} R_k \geq r_j + A^+_{kj} + p_j x^+_{kj}, & (5.3) \\ D_k \leq d_j - B^-_{kj} - p_j x^-_{kj} & (5.4) \\ R_k \geq R_{k-1} + T^-_{k-1,j} + p_j x^-_{k-1,j} + p_{k-1} + T^+_{k,j} + p_j x^+_{kj} & (5.5) \\ D_k \leq D_{k+1} - T^+_{k+1,j} - p_j x^+_{k+1,j} - p_{k+1} - T^-_{k,j} - p_j x^-_{kj} & (5.6) \end{cases}$$

with $x^-_{b_j,j} = x^+_{a_j,j} = 0$.

From system (S) we can derive 4 inequalities bounding $(R_k - D_k)$ as follows:

$$\text{(L)} \begin{cases} R_k - D_k \geq r_j - d_j + p_j(x^+_{kj} + x^-_{kj}) + A^+_{kj} + B^-_{kj} \\ R_k - D_k \geq r_j + p_j(x^+_{kj} + x^+_{k+1,j} + x^-_{kj}) + A^+_{kj} + T^+_{k+1,j} + p_{k+1} + T^-_{k,j} - D_{k+1} \\ R_k - D_k \geq R_{k-1} + p_j(x^-_{k-1,j} + x^+_{kj} + x^-_{kj}) - d_j + B^-_{kj} + T^-_{k-1,j} + p_{k-1} + T^+_{k,j} \\ R_k - D_k \geq R_{k-1} + p_j(x^-_{k-1,j} + x^+_{kj} + x^+_{k+1,j} + x^-_{kj}) + T^+_{k+1,j} + T^-_{k-1,j} + T^+_{k,j} \\ \qquad\qquad + T^-_{k,j} + p_{k-1} + p_{k+1} - D_{k+1} \end{cases}$$

with $x^-_{b_j,j} = x^+_{a_j,j} = 0$. Notice that if no pyramid b_j ($b_j = f - 1$) or a_j ($a_j = \ell + 1$) exists, then all the corresponding terms in system (L) disapear.

Again, we make use of the following additional notations:

- $\alpha_{kj} = r_j - d_j + A^+_{kj} + B^-_{kj}$
  $= r_j - d_j + \sum_{\{i \in P_k, i \neq j | r_i \geq r_j\}} p_i x^+_{ki} + \sum_{\{i \in P_k, i \neq j | d_i \leq d_j\}} p_i x^-_{ki}$,
- $\beta_{kj} = r_j + A^+_{kj} + T^+_{k+1,j} + p_{k+1} + T^-_{k,j} - D_{k+1}$
  $= r_j + \sum_{\{i \in P_k, i \neq j | r_i \geq r_j\}} p_i x^+_{ki} + \sum_{\{i \in P_{k+1}, i \neq j\}} p_i x^+_{k+1,i} + p_{k+1} + \sum_{\{i \in P_k, i \neq j\}} p_i x^-_{ki} - D_{k+1}$,

- $\gamma_{kj} = R_{k-1} - d_j + B^-_{kj} + T^-_{k-1,j} + p_{k-1} + T^+_{k,j}$
  $= \quad R_{k-1} \; - \; d_j \; + \; \sum_{\{i \in P_k, i \neq j | d_i \leq d_j\}} p_i x^-_{ki} \; + \; \sum_{\{i \in P_{k-1}, i \neq j\}} p_i x^-_{k-1,i} \; + \; p_{k-1} \; + \; \sum_{\{i \in P_k, i \neq j\}} p_i x^+_{ki}$,

- $\delta_{kj} = R_{k-1} + T^+_{k+1,j} + T^-_{k-1,j} + T^+_{k,j} + T^-_{k,j} + p_{k-1} + p_{k+1} - D_{k+1}$
  $= \; R_{k-1} \; + \; \sum_{\{i \in P_{k+1}, i \neq j\}} p_i x^+_{k+1,i} \; + \; \sum_{\{i \in P_{k-1}, i \neq j\}} p_i x^-_{k-1,i} \; + \; \sum_{\{i \in P_k, i \neq j\}} p_i x^+_{ki} \; + \; \sum_{\{i \in P_k, i \neq j\}} p_i x^-_{ki} + p_{k-1} + p_{k+1} - D_{k+1}$

Therefore, system (L) can be simply rewritten as:

$$(\mathrm{L}) \begin{cases} R_k - D_k \; \geq \; \alpha_{kj} + p_j(x^+_{kj} + x^-_{kj}) \\ R_k - D_k \; \geq \; \beta_{kj} + p_j(x^+_{kj} + x^+_{k+1,j} + x^-_{kj}) \\ R_k - D_k \; \geq \; \gamma_{kj} + p_j(x^-_{k-1,j} + x^+_{kj} + x^-_{kj}) \\ R_k - D_k \; \geq \; \delta_{kj} + p_j(x^-_{k-1,j} + x^+_{kj} + x^+_{k+1,j} + x^-_{kj}) \end{cases}$$

with $x^-_{b_j,j} = x^+_{a_j,j} = 0$. Notice that if no pyramid b_j ($b_j = f - 1$) or a_j ($a_j = \ell + 1$) exists, then all the corresponding terms in system (L) disapear.

3 Logical reasonning on the (IP)

Consider system (L) and variables x^+_{kj} and x^-_{kj}, $\forall k = f, ..., \ell$. Due to constraints (5.7) and the integrality definition of these variables, only one variable x^+_{kj} or x^-_{kj} can be equal to 1 in the (IP). This leads to $2(\ell - f + 1)$ possibilities. Let us denote by $\vec{x_j} = [x^+_{fj}; x^-_{fj}; x^+_{f+1,j}; x^-_{f+1,j}; ...; x^+_{\ell j}; x^-_{\ell j}]$. We enumerate the different systems (L), numbered from (L_1) to ($\mathrm{L}_{2(\ell-f+1)}$), depending on which variable in $\vec{x_j}$ is equal to 1.

Case $\vec{x_j} = [1; 0; ...; 0]$:

$$(\mathrm{L}_1) \begin{cases} R_f - D_f \; \geq \; \alpha_{fj} + p_j & \\ R_f - D_f \; \geq \; \beta_{fj} + p_j & \\ R_f - D_f \; \geq \; \gamma_{fj} + p_j & \\ R_f - D_f \; \geq \; \delta_{fj} + p_j & \\ R_k - D_k \; \geq \; \alpha_{kj} & \forall k = f+1, ..., \ell \\ R_k - D_k \; \geq \; \beta_{kj} & \forall k = f+1, ..., \ell \\ R_k - D_k \; \geq \; \gamma_{kj} & \forall k = f+1, ..., \ell \\ R_k - D_k \; \geq \; \delta_{kj} & \forall k = f+1, ..., \ell \end{cases}$$

Case $\vec{x_j} = [0; 1; ...; 0]$:

$$(\mathrm{L}_2)\ \begin{cases} R_f - D_f \ \geq\ \alpha_{fj} + p_j \\ R_f - D_f \ \geq\ \beta_{fj} + p_j \\ R_f - D_f \ \geq\ \gamma_{fj} + p_j \\ R_f - D_f \ \geq\ \delta_{fj} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \alpha_{f+1,j} \\ R_{f+1} - D_{f+1} \ \geq\ \beta_{f+1,j} \\ R_{f+1} - D_{f+1} \ \geq\ \gamma_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \delta_{f+1,j} + p_j \\ R_k - D_k \ \geq\ \alpha_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \beta_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \gamma_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \delta_{kj} \qquad \forall k = f+2, ..., \ell \end{cases}$$

Case $\vec{x}_j = [0; 0; 1; ...; 0]$:

$$(\mathrm{L}_3)\ \begin{cases} R_f - D_f \ \geq\ \alpha_{fj} \\ R_f - D_f \ \geq\ \beta_{fj} + p_j \\ R_f - D_f \ \geq\ \gamma_{fj} \\ R_f - D_f \ \geq\ \delta_{fj} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \alpha_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \beta_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \gamma_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \delta_{f+1,j} + p_j \\ R_k - D_k \ \geq\ \alpha_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \beta_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \gamma_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \delta_{kj} \qquad \forall k = f+2, ..., \ell \end{cases}$$

Case $\vec{x}_j = [0; 0; 0; 1; ...; 0]$:

$$
(\mathrm{L}_4)\ \begin{cases}
R_f - D_f \ \geq\ \alpha_{fj} & \\
R_f - D_f \ \geq\ \beta_{fj} & \\
R_f - D_f \ \geq\ \gamma_{fj} & \\
R_f - D_f \ \geq\ \delta_{fj} & \\
R_{f+1} - D_{f+1} \ \geq\ \alpha_{f+1,j} + p_j & \\
R_{f+1} - D_{f+1} \ \geq\ \beta_{f+1,j} + p_j & \\
R_{f+1} - D_{f+1} \ \geq\ \gamma_{f+1,j} + p_j & \\
R_{f+1} - D_{f+1} \ \geq\ \delta_{f+1,j} + p_j & \\
R_{f+2} - D_{f+2} \ \geq\ \alpha_{f+2,j} & \\
R_{f+2} - D_{f+2} \ \geq\ \beta_{f+2,j} & \\
R_{f+2} - D_{f+2} \ \geq\ \gamma_{f+2,j} + p_j & \\
R_{f+2} - D_{f+2} \ \geq\ \delta_{f+2,j} + p_j & \\
R_k - D_k \ \geq\ \alpha_{kj} & \forall k = f+3, ..., \ell \\
R_k - D_k \ \geq\ \beta_{kj} & \forall k = f+3, ..., \ell \\
R_k - D_k \ \geq\ \gamma_{kj} & \forall k = f+3, ..., \ell \\
R_k - D_k \ \geq\ \delta_{kj} & \forall k = f+3, ..., \ell
\end{cases}
$$

...

Case $\vec{x}_j = [0; ...; 1; 0]$:

$$
(\mathrm{L}_{2(\ell-f+1)-1})\ \begin{cases}
R_k - D_k \ \geq\ \alpha_{kj} & \forall k = f, ..., \ell-2 \\
R_k - D_k \ \geq\ \beta_{kj} & \forall k = f, ..., \ell-2 \\
R_k - D_k \ \geq\ \gamma_{kj} & \forall k = f, ..., \ell-2 \\
R_k - D_k \ \geq\ \delta_{kj} & \forall k = f, ..., \ell-2 \\
R_{\ell-1} - D_{\ell-1} \ \geq\ \alpha_{\ell-1,j} & \\
R_{\ell-1} - D_{\ell-1} \ \geq\ \beta_{\ell-1,j} + p_j & \\
R_{\ell-1} - D_{\ell-1} \ \geq\ \gamma_{\ell-1,j} & \\
R_{\ell-1} - D_{\ell-1} \ \geq\ \delta_{\ell-1,j} + p_j & \\
R_\ell - D_\ell \ \geq\ \alpha_{\ell,j} + p_j & \\
R_\ell - D_\ell \ \geq\ \beta_{\ell,j} + p_j & \\
R_\ell - D_\ell \ \geq\ \gamma_{\ell,j} + p_j & \\
R_\ell - D_\ell \ \geq\ \delta_{\ell,j} + p_j &
\end{cases}
$$

Case $\vec{x}_j = [0; ...; 0; 1]$:

$$(\mathrm{L}_{2(\ell-f+1)}) \begin{cases} R_k - D_k \geq \alpha_{kj} & \forall k = f, ..., \ell - 1 \\ R_k - D_k \geq \beta_{kj} & \forall k = f, ..., \ell - 1 \\ R_k - D_k \geq \gamma_{kj} & \forall k = f, ..., \ell - 1 \\ R_k - D_k \geq \delta_{kj} & \forall k = f, ..., \ell - 1 \\ R_\ell - D_\ell \geq \alpha_{\ell,j} + p_j \\ R_\ell - D_\ell \geq \beta_{\ell,j} + p_j \\ R_\ell - D_\ell \geq \gamma_{\ell,j} + p_j \\ R_\ell - D_\ell \geq \delta_{\ell,j} + p_j \end{cases}$$

What we know is that in the optimal solution of the (IP) one of these cases will hold, but we don't know which one. Moreover, the valid case (L_q) in that optimal solution contains valid cuts for the (LP) since the corresponding constraints are stronger that the equivalent ones in the (LP). This is true since in (L_q) the value p_j is added to some constraints whilst in the (LP) we add only a fraction of p_j.

To derive logical cuts, we remark that the following system ($\mathrm{L'}_1$) is less stronger than (L_1) and (L_2), but still valid for the (IP):

$$(\mathrm{L'}_1) \begin{cases} R_f - D_f \geq \alpha_{fj} + p_j \\ R_f - D_f \geq \beta_{fj} + p_j \\ R_f - D_f \geq \gamma_{fj} + p_j \\ R_f - D_f \geq \delta_{fj} + p_j \\ R_k - D_k \geq \alpha_{kj} & \forall k = f+1, ..., \ell \\ R_k - D_k \geq \beta_{kj} & \forall k = f+1, ..., \ell \\ R_k - D_k \geq \gamma_{kj} & \forall k = f+1, ..., \ell \\ R_k - D_k \geq \delta_{kj} & \forall k = f+1, ..., \ell \end{cases}$$

If (L_1) or (L_2) hold in an optimal solution of the (IP), then ($\mathrm{L'}_1$) also holds. Similarly, the following ($\mathrm{L'}_2$) is less stronger than (L_3) or (L_4):

$$(\text{L'}_2)\ \begin{cases} R_f - D_f \ \geq\ \alpha_{fj} \\ R_f - D_f \ \geq\ \beta_{fj} \\ R_f - D_f \ \geq\ \gamma_{fj} \\ R_f - D_f \ \geq\ \delta_{fj} \\ R_{f+1} - D_{f+1} \ \geq\ \alpha_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \beta_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \gamma_{f+1,j} + p_j \\ R_{f+1} - D_{f+1} \ \geq\ \delta_{f+1,j} + p_j \\ R_k - D_k \ \geq\ \alpha_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \beta_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \gamma_{kj} \qquad \forall k = f+2, ..., \ell \\ R_k - D_k \ \geq\ \delta_{kj} \qquad \forall k = f+2, ..., \ell \end{cases}$$

Consequently, we can evince that the $(\ell - f + 1)$ systems (L'_g) are still valid for the (IP) and may lead to cuts in the (LP):

$$(\text{L'}_g)\ \begin{cases} R_g - D_g \ \geq\ \alpha_{gj} + p_j \\ R_g - D_g \ \geq\ \beta_{gj} + p_j \\ R_g - D_g \ \geq\ \gamma_{gj} + p_j \\ R_g - D_g \ \geq\ \delta_{gj} + p_j \\ R_k - D_k \ \geq\ \alpha_{kj} \qquad \forall k = f, ..., \ell, k \neq g \\ R_k - D_k \ \geq\ \beta_{kj} \qquad \forall k = f, ..., \ell, k \neq g \\ R_k - D_k \ \geq\ \gamma_{kj} \qquad \forall k = f, ..., \ell, k \neq g \\ R_k - D_k \ \geq\ \delta_{kj} \qquad \forall k = f, ..., \ell, k \neq g \end{cases}$$

$$\Leftrightarrow$$

$$(\text{L'}_g)\ \begin{cases} R_g - D_g \ \geq\ \max(\alpha_{gj}, \beta_{gj}, \gamma_{gj}, \delta_{gj}) + p_j \\ R_k - D_k \ \geq\ \max(\alpha_{kj}, \beta_{kj}, \gamma_{kj}, \delta_{kj}) \qquad \forall k = f, ..., \ell, k \neq g \end{cases}$$

For each system (L'_g) summing all inequalities yield to the common logical cut:

$$\sum_{k=f}^{\ell}(R_k - D_k) \geq \sum_{k=f}^{\ell} \max(\alpha_{kj}, \beta_{kj}, \gamma_{kj}, \delta_{kj}) + p_j \qquad (LC_j)$$

4 Generating logical cuts

The generation of logical cuts is easy : for all multi-pyramidal job j generate in the (LP) its associate logical cut (LC_j).

References

A.3 Prétraitement d'une formulation mathématique pour le problème $1|r_i|L_{max}$, ROADEF

Prétraitement d'une formulation mathématique pour le problème $1|r_i|L_{\max}$

Briand Cyril[1], T'kindt Vincent[2], Noguer Thomas[2], Zinan Liu[2]

[1] LAAS-CNRS ; Université de Toulouse ; 7, avenue du Colonel Roche, F-31077 Toulouse, France.
briand@laas.fr

[2] Université François Rabelais Tours, Laboratoire d'Informatique (EA 6300), Equipe Ordonnancement et Conduite (ERL CNRS 6305),
64 Avenue Jean Portalis, 37200 Tours, France
tkindt@univ-tours.fr, {thomas.noguer,zinan.liu}@etu.univ-tours.fr

Mots-clés : *Ordonnancement, Prétraitement, Programmation Mathématique*

1 Introduction

Nous nous intéressons à un problème d'ordonnancement à une machine classique noté $1|r_i|L_{max}$ et défini comme suit. On dispose de n travaux à ordonnancer sans préemption, chaque travail j étant caractérisé par une date de début au plus tôt r_j, une date de fin au plus tard d_j et une durée opératoire p_j. L'objectif est de minimiser le plus grand retard algébrique définit par $L_{max} = \max_{1\leq i\leq n}(C_i - d_i)$. Ce problème est NP-difficile au sens fort et à fait l'objet de nombreuses études. En ce qui concerne les méthodes exactes, la procédure par séparation et évaluation proposée par Carlier ([3]) est capable de résoudre très efficacement des instances comptant plusieurs milliers de travaux. Récemment, Briand et al. ([2]) ont proposé une modélisation mathématique efficace de ce problème dont la résolution peut être réalisée avec une efficacité proche de celle de la procédure de Carlier. Cette formulation est fondée sur un théorème de dominance qui permet de restreindre l'espace de recherche à un sous ensemble de séquences de travaux ayant une forme particulière (pyramides).

Dans ce travail, nous nous sommes intéressés à l'application de techniques de prétraitement qui ont déjà montré leur efficacité sur certains problèmes d'ordonnancement $\mathcal{NP}$-difficiles. T'kindt et al. ([5]) ont abordé un problème de flowshop à deux machines particulier pour lequel ils ont pu résoudre à l'optimalité des instances contenant 3500 travaux. Par la suite, Baptiste et al. ([1]) se sont intéressés au problème $1|\tilde{d}_i|\sum_i w_i U_i$ pour lequel ils ont pu résoudre à l'optimalité des instances contenant jusqu'à 35000 travaux. Nous souhaitons, pour le problème $1|r_i|L_{max}$, appliquer cette technique de prétraitement en la particularisant et dans l'objectif d'améliorer les résultats obtenus par le modèle de Briand et al. ([2]).

2 Prétraitement pour le $1|r_i|L_{max}$

La technique de prétraitement repose sur la relation entre un modèle mathématique MIP et sa relaxation continue LP. Soit z^*_{LP} (resp. z^*_{MIP}) la solution optimale du LP (resp. MIP) et B^* la base associée. Le prétraitement consiste alors à effectuer des déductions sur la valeur des variables booléenne, notées $x^-_{k,i}$ et $x^+_{k,i}$ dans le MIP proposé dans [2]. Sans perte de généralité dans la présentation de la technique de prétraitement, nous noterons plus simplement x_i les variables du MIP. On distingue le prétraitement réalisé sur les variables hors base et sur les variables de base.

Il est bien connu en programmation mathématique qu'il existe le lien suivant entre la solution optimale du MIP et la solution optimale du LP :

$$z^*_{MIP} = z^*_{LP} + \sum_{x_i \notin B^*} c_i x_i \qquad \text{(I)},$$

avec c_i le coût réduit associé à la variable x_i. Soit UB une borne supérieure (par exemple, obtenue avec l'algorithme de Schrage). On peut alors déduire de (I) :

$$\begin{aligned} & UB \geq z^*_{LP} + \sum_{x_i \notin B^*} c_i x_i \\ \Leftrightarrow & \sum_{x_i \notin B^*} c_i x_i \leq UB - z^*_{LP} \qquad \text{(II)}. \end{aligned}$$

De l'inégalité (II) on peut en déduire la régle de fixation suivante : $\forall x_i \notin B^*$, si $c_i > UB - z^*_{LP}$ alors dans toute solution optimale du MIP, $x_i = 0$. En raisonnant sur les variables d'écart associées aux variables x_i on peut en déduire si $x_i = 1$.

Pour ce qui concerne les variable de base $x_i \in B^*$, on utilise les pseudo-couts l_i et u_i calculés par exemple en utilisant les pénalités de Driebeek. Ces pénalités correspondent à une évaluation par défaut de l'augmention de z^*_{LP} si x_i est mise à 0 ou 1 : elles dépendent des valeurs des coûts réduits des variables hors base. On a :

1. $\forall x_i \in B^*$, si $(l_i x_i) \geq (UB - z^*_{LP})$ alors dans une solution optimale du MIP on a $x_i = 1$,
2. $\forall x_i \in B^*$, si $(u_i(1 - x_i)) \geq (UB - z^*_{LP})$ alors dans une solution optimale du MIP on a $x_i = 0$.

En pratique, l'efficacité du prétraitement dépend de l'ajout d'informations (bornes, coupes) que l'on peut faire dans le LP. Pour le problème $1|r_i|L_{max}$, partant de la modélisation de Briand et al., nous pouvons ajouter une coupe très simple. Soit LB_{imp} proposée par Della Croce et T'kindt ([4]) et qui améliore la solution préemptive calculée à partir de l'ordonnancement préemptif de Jackson. On peut alors ajouter la contrainte $z_{LP} \geq LB_{imp}$.
De même, nous avons commencé à étudier l'ajout de coupes dans le modèle.

Lors de la conférence, nous présenterons des résultats expérimentaux qui permettront de mesurer si cette technique de prétraitement aide à la résolution du problème.

Références

[1] Baptiste P., Della Croce F., Grosso A., T'kindt V., " Sequencing a single machine with due dates and deadlines : An ILP-based approach"', *Journal of Scheduling*, 13(1) :39-47, 2010.

[2] Briand C., Ourari S., Bouzouia B. "An efficient ILP formulation for the single machine scheduling problem", *submitted* to *RAIRO - Operations Research*, 2008.

[3] Carlier, J., "The one-machine sequencing problem", 1982, *European Journal of Operational Research*, 11, 42-47.

[4] Della Croce F., T'kindt V., "Improving the preemptive bound for the single machine min-max lateness problem subject to release times"', *Operations Research Letters*, 38(10), 2010.

[5] T'kindt V., Della Croce F., Bouquard J.-L., "Enumeration of Pareto optima for a flowshop scheduling problem with two criteria"', *Informs Journal on Computing*, 19(1) :64-72, 2007.

Prétraitement pour le problème du $1|r_i|L_{max}$

Département Informatique
5e année
2012 - 2013

Rapport de projet de fin d'études

Résumé : Ce projet présente l'utilisation de la méthode du prétraitement pour le problème d'ordonnancement du $1|r_i|L_{max}$, son intérêt et ses possibilités. Il utilise un modèle mathématique pyramidal et une relaxation continue de ce modèle. Une démarche de recherche de coupes à été effectuée ainsi qu'une amélioration de la borne supérieure par une recherche locale de la solution de l'algorithme de Schrage. Des campagnes de tests ont été lancées et les résultats ainsi que les interprétations sont présents dans ce rapport.

Mots clefs : $1|r_i|L_{max}$, prétraitement, pyramide, coupe, relaxation continue, modèle mathématique

Abstract: This projet present the use of the preprocessing method for the $1|r_i|L_{max}$ scheduling problem, its interest and possibilities. It uses a pyramidal mathematical model and the continuous relaxation of the model. An approach of research of cuts has been performed aswell with an amelioration of the upper bound with a local research of the Schrage algorithm. Many test campagns have been launched, the results and interpretations are present in this report.

Keywords: $1|r_i|L_{max}$, preprocessing, pyramide, cut, continuous relaxation, mathematical model

---

Encadrant
Vincent T'KINDT
tkindt@univ-tours.fr

Étudiant
Thomas NOGUER
thomas.noguer@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2012 - 2013