7e Conférence Francophone de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 - Paris - France
«Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination»

ORDONNANCEMENT DES TRAVAUX INTERFÉRANT DANS UN FLOWSHOP À DEUX MACHINES

N. HUYNH TUONG, A. SOUKHAL et J.-C. BILLAUT *

Laboratoire d'Informatique, Université François Rabelais, Tours
64 Avenue Jean Portalis - 37200 Tours
nguyen.huynh@etu.univ-tours.fr, {ameur.soukhal, jean.billaut}@univ-tours.fr

RÉSUMÉ : *Dans la pratique, il n'est pas toujours pertinent de vouloir mesurer la qualité d'un ordonnancement à l'aide d'un critère qui soit le même pour tous les travaux. Il arrive qu'un critère imposé par un client soit pertinent vis-à-vis de sa demande, i.e. pour les travaux qui le concernent, mais pas nécessairement pour les autres travaux à réaliser. On cherche à ordonnancer n travaux dans un atelier de type flowshop à deux machines. Deux critères réguliers sont à considérer lors du calcul de l'ordonnancement : le premier critère porte sur un sous-ensemble donné de travaux tandis que le second porte sur la globalité des travaux. On parle de problème d'ordonnancement avec "travaux interférants". Des résultats de complexité sont présentés montrant la NP-complétude au sens ordinaire des problèmes considérés. Un algorithme pseudo-polynomial est proposé. Un cas particulier est identifié polynomial où un algorithme en $O(n\log(n))$ est proposé.*

MOTS-CLÉS : *Ordonnancement ; Job Interferant ; Flowshop ; Complexité; Algorithme pseudo-polynomial.*

1. INTRODUCTION

Depuis l'apparition du premier article en ordonnancement en 1954 et jusqu'à la fin des années 80, la fonction objectif était classiquement définie par un unique critère à optimiser pour l'ensemble des travaux. En effet, les modèles traditionnels d'ordonnancement considèrent que les travaux sont tous équivalents et que l'ordonnancement global doit être mesuré sur tous les travaux sans distinction entre eux.

On s'intéresse dans cet article aux nouveaux problèmes d'ordonnancement qui apparaissent dans le cadre de la gestion de la chaîne logistique (GCL, *Supply Chain Management* en anglais) où plusieurs entités, fournisseurs, producteurs et clients coopèrent pour obtenir des ordonnancements compatibles. Il arrive qu'on souhaite achever les travaux dans les plus brefs délais ($C_{\max}$), qu'on souhaite minimiser le temps de séjour de l'ensemble des travaux ($\overline{C}$) voire minimiser un critère lié au retard des travaux ($T_{\max}, L_{\max}, \overline{U}$). Cependant, dans le cadre de l'ordonnancement au sein de la GCL, les modèles d'ordonnancement basés sur ces critères réguliers classiques ($C_{\max}, \overline{C}, T_{\max}, L_{\max}, \ldots$) ne sont pas toujours totalement satisfaisants. Dans les situations pratiques, un décideur doit tenir compte simultanément de plusieurs objectifs. Par exemple, lors de la phase de la planification à court terme, les objectifs tels que la satisfaction du client et la minimisation des encours aussi bien que des coûts de fabrication, le temps de séjour moyen des produits ne doivent pas être négligés.

Afin d'atteindre un compromis acceptable, on doit mesurer la qualité d'une solution sur tous les critères importants. Cette notification a mené au développement du domaine de l'ordonnancement multicritère. L'étude des problèmes d'ordonnancement multicritère a débuté il y a plus d'une vingtaine d'années avec toujours un intérêt de plus en plus croissant (voir [T'kindt et Billaut, 2006] et [Hoogeveen, 2005]). Elle vise à aider les décideurs qui doivent tenir compte simultanément de plusieurs objectifs à la fois, à trouver la solution de meilleur compromis.

Il n'est donc pas toujours pertinent de vouloir mesurer la qualité d'un ordonnancement à l'aide d'un unique

*Les auteurs remercient le GDR CNRS "Recherche Opérationnelle" pour son soutien au projet "ORDO-COO-SC" (Ordonnancements coopératifs liés au management futur des chaînes logistiques) dont cette étude fait partie.

critère. Parfois même, un critère imposé par un client est pertinent pour sa demande, donc pour certains travaux, mais pas pour l'ensemble des travaux à réaliser au sein de l'atelier de production. La notion de problème d'ordonnancement bi-critère considérée dans cette étude est cependant différente de la notion classique. On parle alors de "jobs interférant". Dans [Ehrgott, 2005], l'auteur cite un problème lié à l'industrie pharmaceutique où deux types de médicament A et B sont produits. Selon les demandes instantanées des clients pour le produit A, ce dernier est plus prioritaire. Cependant, le niveau d'importance de ces deux types n'est pas quantifiable. L'auteur propose une résolution de ce problème par l'approche ε-contrainte.

Le cas de flowshop à deux machines a été peu traité dans la littérature. Dans [Agnetis et al., 2000], les auteurs considèrent deux types de travaux classés selon les objectifs des différents clients à ordonnancer dans un atelier de type job shop. Dans leur analyse, ils montrent comment calculer des ordonnancements non-dominants de sorte que les négociations avec les clients puissent se faire de façon efficace. Dans [Agnetis et al, 2004], les auteurs considèrent un problème d'ordonnancement bi-critère avec deux clients distincts où l'approche ε-contrainte est appliquée. Il s'agit donc de minimiser le premier critère lié au premier client sous la contrainte que la valeur du deuxième critère défini par le second client soit bornée par ε. Trois classes de fonctions objectif suivantes sont étudiées : critère régulier, somme pondérée des dates d'achèvement et nombre de travaux en retard. De nouveaux résultats de complexité sont élaborés pour les neuf cas étudiés.

Dans [Baker et Smith, 2003], les auteurs étudient des problèmes d'ordonnancement à une seule machine avec différents critères réguliers $(C_{max}, \sum w_i C_i, L_{max})$ associés aux différentes catégories de clients. On définit autant de types de clients que d'objectifs. Les auteurs s'intéressent à deux et trois catégories de clients, c'est-à-dire deux et trois sous-ensembles de travaux. Il s'agit donc de calculer une séquence optimale pour minimiser les combinaisons linéaires convexes de ces critères. Quelques cas polynomiaux sont identifiés et d'autres sont montrés NP-difficiles au sens fort.

Cet article est organisé comme suit. Dans la section *2*, nous présentons notre modèle, les hypothèses de travail et les notations utilisées. Dans la section *3*, de nouveaux résultats de complexité sont exposés montrant principalement la NP-Complétude du problème considéré. Un cas polynomial est identifié. La section *4* est consacrée à la résolution des problèmes montrés NP-difficiles dans la section *3*. Notamment, un algorithme optimal pseudo-polynomial est présenté.

2. NOTATIONS ET DÉFINITIONS DU PROBLÈME

Il s'agit d'ordonnancer n travaux dans un flowshop à deux machines. Nous supposons que tous les travaux sont disponibles à la date zéro ; la préemption n'est pas autorisée ; le chevauchement des opérations n'est pas autorisé ; les durées opératoires sont connues, indépendantes et entières ; les deux machines sont disponibles et ne peuvent traiter qu'une opération à la fois. On note N l'ensemble des travaux et N_1 un sous-ensemble de N ($N_1 \subset N$). L'ensemble N_1 peut correspondre aux travaux associés à un certain client, par exemple. Deux critères sont considérés à la fois : la durée totale de l'ordonnancement des travaux de $N1$ notée $C^{N1}_{\max}$ et l'autre lié à la durée totale de l'ordonnancement des autres travaux notée $C^{N}_{\max}$. Dans la pratique on pourrait considérer des critères différents, mais le C_{max} présente l'avantage d'être le plus simple, ce qui fait l'intérêt de l'étude.

À chaque travail J_k ($1 \leq k \leq n$) on associe une durée $p_{i,k}$ sur M_i ($i \in \{1, 2\}$). On note C_k la date de fin d'exécution de J_k sur M_2. De façon classique, $C_{max} = \max_{1 \leq k \leq n} C_k$ est le maximum des dates de fin d'exécution, ou makespan.

Selon [Graham et al, 1979] et [T'kindt et Billaut, 2006], la notation des problèmes considérés est : $F2||\epsilon(C^{N}_{\max}/C^{N_1}_{\max})$ (on minimise $C^{N}_{\max}$ sous la contrainte $C^{N_1}_{\max} \leq \epsilon$) ou $F2||\epsilon(C^{N_1}_{\max}/C^{N}_{\max})$ (l'inverse).

3. NOUVEAUX RÉSULTATS DE COMPLEXITÉ

Selon Brucker [Brucker et Knust, 2006], les problèmes d'ordonnancement avec la minimisation d'un critère régulier dans un flowshop à deux machines sont en général NP-difficiles à l'exception de la minimisation du makespan [Johnson, 1954].

Dans cette section, il s'agit d'étudier les deux cas de figures suivants : Optimiser le critère $C_{\max}$ défini sur le sous-ensemble de travaux $N1$ tout en essayant d'optimiser la date de fin de la globalité des travaux et inversement.

Nous étudions dans la sous-section 3.1 la complexité du problème $F2||\epsilon(C^{N}_{\max}/C^{N_1}_{\max})$. Le problème inverse sera étudié dans la sous-section 3.2.

3.1. $F2|C^{N_1}_{\max} \leq \varepsilon|C^{N}_{\max}$

Rappelons que le cas classique $F2||C_{\max}$ est polynomial. Notons par $C^*_{\max}(N_1)$ la valeur optimale obtenue en appliquant l'algorithme de Johnson en ne considérant que les travaux de N_1. Deux cas de figure se présentent : dans le premier cas le décideur cherche à optimiser $C^N_{\max}$ sous la contrainte $\varepsilon = C^*_{\max}(N_1)$. Dans le second cas le décideur tolère une certaine marge sur la date d'achèvement des travaux de $N1$, i.e. $\varepsilon > C^*_{\max}(N_1)$.

3.1.1. $\varepsilon = C^*_{\max}(N_1)$

Il s'agit ici de trouver un ordonnancement qui permet de finir au plutôt les travaux de N_1 tout en minimisant le makespan global.

Soit l'algorithme suivant.

Algorithm 1 $F2|C^{N1}_{\max} \leq C^*_{\max}(N_1)|C^N_{\max}$

1: Soit σ_1 l'ensemble des travaux de N_1 ordonnés selon la règle de Johnson
2: Soit σ_2 l'ensemble des travaux de $N - N_1$ ordonnés selon la règle de Johnson
3: Ordonnancer les travaux de σ_1
4: Ordonnancer les travaux de σ_2
5: Retourner $C^{N_1}_{\max}$ et $C^N_{\max}$

Théorème 1. *L'algorithme 1 retourne une solution optimale pour le problème* $F2|C^{N_1}_{\max} \leq C^*_{\max}(N_1)|C^N_{\max}$ *en* $O(n\log(n))$

Preuve. Le makespan obtenu par Johnson sur N_1 est optimal. Il est bien connu que résoudre le $F2||C_{\max}$ avec la non disponibilité des machines M_1 et M_2 à l'instant zéro se fait également par l'algorithme de Johnson. □

3.1.2. $\varepsilon > C^*_{\max}(N_1)$

Nous considérons qu'une certaine marge par rapport à la valeur optimale de la date de fin d'exécution des travaux de N_1 est tolérée. Ce cas est intéressant car il est alors possible de retarder un peu les travaux de N_1 pour permettre l'exécution de certains travaux de N, et donc terminer les travaux de N plus tôt.

Théorème 2: Le problème $F2|C^{N_1}_{\max} \leq \varepsilon|C^N_{\max}$ est NP-difficile au sens ordinaire.

Preuve: Le problème est dans $\mathcal{NP}$. La preuve de complexité est donnée par la réduction du problème de *Partition avec cardinalités égales* qui est NP-difficile au sens ordinaire.

Problème de *Partition avec cardinalités égales* [Garey et Johnson, 1979] : Soit l'ensemble de $2n$ entiers $a_1, \ldots, a_{2n}$ et un entier B tel que $\sum_{i=1}^{2n} a_i = 2B$. Existe-t-il une partition des indices $\{1, \ldots, 2n\}$ en deux sous-ensembles A_1 et A_2 de telle sorte que : $\sum_{i\in A_1} a_i = \sum_{i\in A_2} a_i = B$ et $|A_1| = |A_2| = n$?

On va montrer que si on sait résoudre le problème $F2|C^{N_1}_{\max} \leq \varepsilon|C^N_{\max}$ alors on sait résoudre ce problème de Partition.

Pour une instance donnée de la Partition avec cardinalités égales, nous construisons l'instance pour le problème $F2|C^{N_1}_{\max} \leq \varepsilon|C^N_{\max}$ comme suit :

Soit l'ensemble N de $2n + 2$ travaux à ordonnancer. Soit l'ensemble $N1$ défini comme suit : $N_1 = \{J_{2n+1}, J_{2n+2}\}$. Les durées opératoires des travaux sont présentées dans le tableau de la figure 1.

	$p_{1,i}$	$p_{2,i}$
$J_i, i = 1, \ldots, 2n$	1	$2nB + na_i$
J_{2n+1}	$2n^2B + nB - n + 1$	$2n^2B + nB$
J_{2n+2}	$2n^2B + nB$	n

Figure 1: Durées opératoires des $2n + 2$ travaux

On pose $\epsilon = 2nB(2n + 1) + n + 1$ et $y = 3nB(2n + 1) + n + 1$.

Nous montrons que le problème *Partition avec cardinalités égales* a une réponse 'oui' si et seulement si le problème d'ordonnancement avec $C^{N_1}_{\max} \leq \epsilon$ et $C^N_{max} \leq y$ (version décisionnelle de notre problème) a une réponse 'oui'.

($\Rightarrow$) Si la réponse au problème de Partition est 'oui', alors nous pouvons construire une solution pour le problème d'ordonnancement comme suit. Nous avons d'abord A_1 et A_2 les ensembles d'indices qui constituent une solution du problème de Partition.

Nous obtenons par la suite trois sous-ensembles de travaux :

- L'ensemble X contient les travaux avec a_i défini dans $p_{2,i}$ correspond au élément appartenant à A_1
- L'ensemble Y contient les travaux avec a_i défini dans $p_{2,i}$ correspond au élément appartenant à A_2
- L'ensemble $N_1 = \{J_{2n+1}, J_{2n+2}\}$.

La séquence à réponse 'oui' est obtenue en ordonnançant les travaux de X dans un ordre quelconque puis de J_{2n+1}, J_{2n+2} puis ceux de Y dans un ordre

quelconque. Par conséquent, nous avons : $C^{N_1}_{\max} = 2nB(2n+1)+n+1$ et $C^{N}_{\max} \leq 3nB(2n+1)+n+1$ et donc la réponse au problème d'ordonnancement est bien 'oui'.

($\Leftarrow$) Supposons qu'il existe une solution S au problème d'ordonnancement qui termine les travaux de N_1 avant la date $\varepsilon = 2nB(2n+1)+n+1$ et avec $C^{N}_{\max} = 3nB(2n+1)+n+1$.

On introduit la notation suivante :

- $S^{(S)}_{(1,k)}$ et $C^{(S)}_{(1,k)}$: la date du début et de fin d'exécution de J_k sur M_1 selon S,
- $S^{(S)}_{(2,k)}$ et $C^{(S)}_{(2,k)} = C_k$: la date de début et de fin d'exécution de J_k sur M_2 selon S,
- $S^{(S)}_{(1,N_1)}$ et $C^{(S)}_{(1,N_1)}$: la date du début et de fin d'exécution des travaux de N_1 sur M_1,
- $S^{(S)}_{(2,N_1)}$ et $C^{(S)}_{(2,N_1)}$: la date de début et de fin d'exécution des travaux de N_1 sur M_2,

Soit LB une borne inférieure à C^{N}_{max}.

$$LB = \sum_{i=1}^{2n+2} p_{2,i} + \min_{1\leq i\leq 2n+2} p_{1,i} = \sum_{i=1}^{2n+2} p_{2,i} + 1.$$

On montre facilement que $LB = 6n^2B+3nB+n+1 = C^{N}_{max}$.

Par conséquent, la somme des temps mort sur M_2 vaut 1, donc on a un temps mort d'une unité de temps au début de $M2$. Donc, les travaux de N_1 ne peuvent pas commencer à la date zéro sur M_1. De plus, selon Johnson, il vaut mieux que J_{2n+1} soit placé avant J_{2n+2}. De plus, si certains jobs sont ordonnancés entre J_{2n+1} et J_{2n+2}, on peut déplacer ces jobs juste après la fin de J_{2n+2} sans générer de temps mort sur M_2. Cela permet d'avancer la date de fin des travaux de N_1 sur M_2 (notée $C^{(S)}_{(2,N_1)}$).

Notons X le sous-ensemble de travaux ordonnancé avant N_1 et $k = |X|$. Le reste des travaux constitue l'ensemble Y et est ordonnancé après le travail J_{2n+2}. Il faut montrer que $k = n$ et que la somme des a_i pour les travaux de X est égale à B.

On sait que J_{2n+2} doit finir avant ϵ donc $C_{2n+2} \leq \epsilon$

$\Leftrightarrow 1 + 2nBk + n\sum_X a_i + 2n^2B + nB + n \leq 4n^2B + 2nB + n + 1 \quad \Leftrightarrow 2nBk + n\sum_X a_i \leq 2n^2B + nB$

$\Leftrightarrow 2nB(n-k) + n(B - \sum_X a_i) \geq 0$ (1)

D'autre part, J_{2n+1} finit sur $M1$ avant les travaux de X sur $M2$ sinon il provoquerait un temps mort sur M_2, et on sait que la borne y est respectée.

$C^{(S)}_{(1,2n+1)} \leq C^{(S)}_{(2,X)} \Leftrightarrow C^{(S)}_{(1,2n+1)} \leq 1+2nBk+n\sum_X a_i$

$\Leftrightarrow k + 2n^2B + nB - n + 1 \leq 1 + 2nBk + n\sum_X a_i$

$\Leftrightarrow k + 2n^2B + nB - n \leq 2nBk + n\sum_X a_i$ (2)

$\Leftrightarrow (2nB-1)(k-n) + n(\sum_X a_i - B) \geq 0$ (3)

Déduire deux inégalités (1) et (3), on obtient l'inégalité suivante :

$n - k \geq 2nB(n-k) + n(B - \sum_X a_i) \geq 0$ (4)

Supposons $k < n$. Dans ce cas, $k+1 \leq n$. Puisque $\sum_X a_i \leq 2B$, on a $2nBk + n\sum_X a_i < 2nBk + 2nB$ donc $2nBk + n\sum_X a_i < 2nB(k+1)$. Donc $2nBk + n\sum_X a_i < 2n^2B < 2n^2B + (n(B-1)+k)$, soit $2nBk + n\sum_X a_i < 2n^2B + n(B-1) + k$. Ceci est en contradiction avec (2), donc on n'a pas $k < n$. Par conséquent, $k = n$ et on déduit de (4) que $\sum_X a_i = B$.

Ainsi, le problème $F2|C^{N_1}_{\max} \leq \varepsilon|C^{N}_{\max}$ est NP-difficile. □

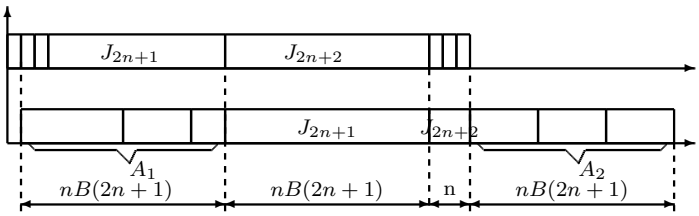


Figure 2: Exemple illustrant une solution S

3.2. $F2|C^{N}_{\max} \leq \varepsilon|C^{N_1}_{\max}$

Considérons maintenant le cas où on souhaite minimiser le makespan pour le sous-ensemble N_1 sous la condition que la valeur de l'ordonnancement de la globalité des travaux vérifie la contrainte $C^{N}_{\max} \leq \varepsilon$.

On se place ici dans le cas où le décideur donne une borne sur la date de fin de l'ensemble des travaux, tout en minimisant la date de fin des travaux de N_1. Comme pour le cas précédent, on distingue deux cas selon la valeur de ϵ.

3.2.1. $\varepsilon = C^{*}_{\max}(N)$

Théorème 3: Le problème $F2|C^{N}_{\max} \leq C^{*}_{\max}(N)|C^{N_1}_{\max}$ est NP-difficile au sens ordinaire.

Preuve: Nous considérons l'instance de la figure 1. En suivant les mêmes démarches que dans la preuve du théorème 2 (section 3.1.2.), nous montrons que le problème $F2|C^{N}_{\max} \leq C^{*}_{\max}(N)|C^{N1}_{\max}$ se réduit au problème de Partition avec cardinalités égales .

D'après l'instance choisie, on montre que $\varepsilon = C^*_{\max}(N) = 3nB(2n+1)+n+1$ (cf. figure 2). Ainsi, la séquence obtenue est optimale pour les deux critères $C^{N_1}_{\max}$ et $C^N_{\max}$.□

3.2.2. $\varepsilon > C^*_{\max}(N)$

Théorème 4: Le problème $F2|C^N_{\max} \leq \varepsilon|C^{N_1}_{\max}$ est NP-difficile au sens ordinaire.

Preuve: Une petite modification est apportée à l'instance de la figure 1 : la durée de la première opération du travail J_{2n+1} est augmentée de α unités de temps ($1 \leq \alpha \leq n-1$), i.e $p_{1,2n+1} = 2n^2B + nB - n+1+\alpha$. Sans perte de généralité, supposons que $\alpha = 1$. En suivant la même démarche que dans la preuve du théorème 2, pour $\varepsilon \leq 3nB(2n+1) + n + 1 + \alpha$, nous montrons (voir figure 3):

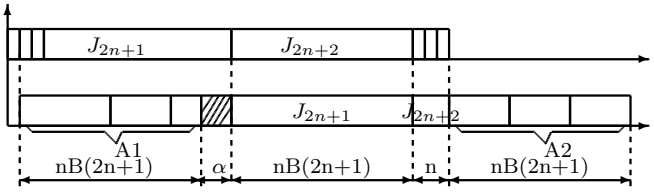


Figure 3: Exemple illustrant une solution S avec 6 travaux

- $y = LB = 2nB(2n+1) + n + 1 + \alpha$
- Dans S, n travaux appartenant à l'ensemble $(N - N_1)$, sont ordonnancés dans l'intervalle $[0, 2n^2B + nB + 1]$
- Dans S, J_{2n+1} précède J_{2n+2} et ces travaux sont ordonnancés dans l'intervalle $[n, 2nB(2n+1) + n + 1 + \alpha]$

Ainsi, le problème $F2|C^N_{\max} \leq \varepsilon|C^{N_1}_{\max}$ est NP-difficile au sens ordinaire.□

4. RÉSOLUTION DU $F2|C^{N_1}_{max} \leq \varepsilon|C^N_{max}$ ET DU $F2|C^N_{max} \leq \varepsilon|C^{N_1}_{max}$

On introduit les notations suivantes :

- S : une séquence des travaux.
- $J_{[i]}$: le travail qui se trouve à la position i dans la séquence S,
- $\sigma^{(S)}_{(k,N_1)}$: la sous-séquence de S des travaux ordonnancés à partir du job J_k jusqu'à le dernier travail de N_1,
- $S^{(S)}_{(i,\sigma)}$: la date du début d'exécution de la sous-séquence σ sur M_i, $i \in \{1, 2\}$,
- $X^{(S)}_{(k)}$: le temps mort créé par l'exécution de la seconde opération de J_k sur M_2,
- $\psi^{(S)}_k$: la somme des temps mort sur M_2 avant la date de fin de J_k, $k = 1, 2, \ldots, n$;
- $X^{(S)}_{(N_1)}$: la somme des temps mort sur M_2 entre le premier et dernier travail de N_1
- $X^{(S)}_{(k,N_1)}$: la somme des temps mort après le travail J_k, incluant J_k, et avant la date de fin de l'ensemble des travaux N_1 sur M_2,
- $X^{(S)}_{(N_1,k)}$: la somme des temps mort après la date de fin de l'ensemble des travaux N_1 avant le travail J_k sur M_2,
- $X^{(S)}_{(N_2)}$: la somme des temps mort après le dernier travail de N_1 sur M_2 ($N_2 = N - N_1$),
- $X^{(S)}_{\sigma}$: la somme des temps mort créés par la sous-séquence σ sur M_2,
- $X^{(S)}$: la somme des temps mort sur M_2 selon la séquence S,
- $Y^{(S)}_k$: la distance entre la date de fin de J_k sur M_1 et sa date de fin sur M_2 $k = 1, 2, \ldots, n$.

Pour résoudre les problèmes identifiés NP-difficiles au sens ordinaire, nous proposons une méthode pseudo-polynomiale exacte de programmation dynamique. Notre démarche sera montrée uniquement pour le cas $F2|C^{N_1}_{max} \leq \varepsilon|C^N_{max}$ mais elle reste valide pour le deuxième problème considéré $F2|C^N_{max} \leq \varepsilon|C^{N_1}_{max}$.

Afin de décrire notre méthode de résolution, nous présentons d'abord quelques propriétés. Pour plus de clarté, les propriétés élaborées sont classées selon trois catégories que nous nommons : propriété fondamentale, propriété supplémentaire et propriétés cruciales. La première catégorie définit les caractéristiques d'une solution optimale qui peut être déduite à partir de la règle de Johnson [Johnson, 1954]. Cette première propriété nous permet de définir la structure d'une solution exacte. La propriété de la seconde classe est pour déterminer les caractéristiques d'une solution dans le cas du $F2||C_{\max}$ avec contrainte d'indisponibilité à la date zéro. Grâce à cette deuxième classe, les propriétés de la dernière classe peuvent être montrées commodément. Cette dernière classe nous permet de prouver l'optimalité de notre algorithme.

4.1. PROPRIÉTÉS FONDAMENTALES

Propriété 1

Soit J_k le dernier travail ordonnancé de l'ensemble N_1. Il existe une solution optimale S^* qui respecte les deux propriétés suivantes :

1. tous les travaux ordonnancés avant J_k et incluant J_k sont classés selon la règle de Johnson.
2. tous les travaux ordonnancés après J_k sont également classés selon la règle de Johnson.

Preuve. La preuve est donnée par la règle d'échange entre deux travaux adjacents.□

On définit une solution initiale basée sur cette propriété, où les travaux de N_1 sont ordonnancés en première position dans l'ordre de Johnson, suivis du reste des travaux de $N_2 = N \setminus N_1$ dans l'ordre de Johnson également.

Selon la valeur de la solution initiale obtenue, nous déduisons alors :

- Si $C_{max}^{N_1} > \varepsilon$, alors, il n'existe pas de solution faisable,

- Si $C_{max}^{N_1} = \varepsilon$, la solution initiale est optimale,

- Sinon, nous essayons d'insérer certains travaux de N_2 dans la première sous-séquence en respectant $C_{max}^{N_1} \leq \varepsilon$.□

Dans le dernier cas de figure, il est clair que l'insertion d'un travail parmi les travaux de N_1 avant la date ε, doit respecter l'ordre de Johnson. C'est le principe de base de notre programme dynamique.

Afin de déterminer quel travail doit être déplacé dans la première partie de l'ordonnancement partiel, nous étudions d'abord le cas de $F2||C_{\max}$ avec une contrainte d'indisponibilité. En général, ce problème est NP-difficile sauf si l'indisponibilité des machines est définie à la date zéro [Lee, 1997].

4.2. PROPRIÉTÉS SUPPLÉMENTAIRES

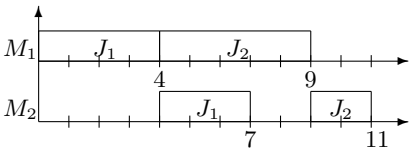
Propriété 2

Considérons le problème d'indisponibilité $F2|[0, d_1], [0, d_2], d_2 \geq d_1|C_{max}$ où l'intervalle de temps $[0, d_1]$ (resp. $[0, d_2]$) est la fenêtre d'indisponibilité de la machine M_1 (resp. $M2$). Soit X la somme des temps morts sur M_2 à partir de d_2 d'une solution optimale du problème d'indisponibilité. Soit $X^{Johnson}$ la somme des temps morts sur M_2 de la séquence obtenue par Johnson pour la même instance sans la contrainte d'indisponibilité. Nous avons donc :

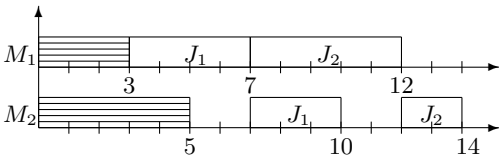
$$X = max(0, X^{Johnson} - (d_2 - d_1))$$

Preuve. Avec ou sans ces périodes d'indisponibilité, la séquence optimale est donnée par Johnson. Si $d_2 \leq d_1$, alors il faut ajouter au temps mort $X^{Johnson}$ la première quantité de temps mort qui correspond à la valeur de $(d_1 - d_2)$ puisque cela n'a aucune influence sur la date de début d'exécution de la deuxième opération du premier travail $J_{(1)}$ sur M_2 qui commence après $d_1 + p_{(1),1}$. Sinon, il suffit de considérer un travail fictif J_0 avec $p_0, 1 = 0$ et $p_0, 2 = d_2 - d_1$. Selon Johnson, afin de réduire les temps mort, ce travail sera placé au début de l'ordonnancement. □

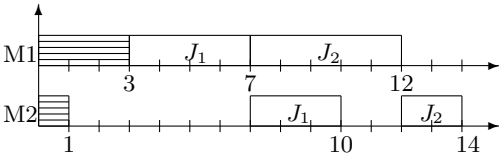
Exemple Considérons un exemple suivant avec la somme des temps morts de 6:



S'il existe des fenêtres d'indisponibilité avec $d_1 = d_2$ (=3 par exemple), la somme des temps mort pour la même séquence est inchangé. Dans le cas où $d_1 = 3$ et $d_2 = 5$, la somme des temps morts est de 4:



Inversement, dans le cas où $d_1 = 3$ et $d_2 = 1$, alors, la somme des temps morts est 8:



De plus, la somme des temps morts peut être égale à zéro si $d_1 = 3$ et $d_2 \geq 9$.

Corollaire

Soit deux solutions S et S' de l'instance I et l'instance I' respectivement. Supposons que:
$\begin{cases} d_2 - d_1 = d'_2 - d'_1 \\ X(S) \leq X(S') \end{cases}$

Si on change la fenêtre d'indisponibilité de $[0, d_1]$ et $[0, d_2]$ (resp. $[0, d'_1]$ et $[0, d'_2]$) à $[0, d_3]$ et $[0, d_4]$ (resp. $[0, d'_3]$ et $[0, d'_4]$) telque $d_4 - d_3 = d'_4 - d'_3$. Nous pouvons déduire que $X(S_1) \leq X(S'_1)$.

Preuve. Soit $X^J(S)$ (resp. $X^J(S')$) la somme des temps morts de S (resp. S'). Selon la propriété *2*:
$\begin{cases} X(S) &= max(0, X^J(S) - (d_2 - d_1)) \\ X(S') &= max(0, X^J(S') - (d'_2 - d'_1)) \end{cases}$

$\Rightarrow max(0, X^J(S) - (d_2 - d_1)) \leq max(0, X^J(S') - (d'_2 - d'_1)) \Rightarrow X^J(S) \leq X^J(S')$

$\Rightarrow max(0, X^J(S) - (d_4 - d_3)) \leq max(0, X^J(S') - (d'_4 - d'_3)) \Rightarrow X(S_1) \leq X(S'_1)$. □

4.3. PROPRIÉTÉS CRUCIALES

À partir de la solution initiale, nous allons analyser l'effet de déplacer un travail de $N_2 = N - N_1$ et de l'insérer dans la première partie de l'ordonnancement.

Considérons le cas suivant :

Soient deux solutions S, S' telles que $C^{(S)}_{max} < C^{(S')}_{max}$ **(1)**. On suppose que dans les deux solutions, le travail J_k est ordonnancé après le dernier travail de $N1$.

Soit J_{j1} (resp. J_{j2}) le premier travail qui est ordonnancé dans S (resp. S') avant le dernier travail de $N1$ et qui est ordonné après J_k selon l'ordre Johnson.

Supposons que $j1 = j2 = j$. Ainsi, nous avons :

$\sigma^{(S)}_{(j1,N_1)} = \sigma^{(S')}_{(j2,N_1)}$ **(2)** et $\sigma^{(S)}_{(k,N_2)} = \sigma^{(S')}_{(k,N_2)}$ **(3)**.

Propriété 3

Soit S_2 (resp. S'_2) une solution construite à partir de S (resp. S') en insérant J_k juste après J_{j1} (resp. J_{j2}). Si nous avons :

$$\begin{cases} X^{(S)}_{(j,N_1)} &= X^{(S')}_{(j,N_1)} & \textbf{(4)} \\ X^{(S)}_{(N_1)} &= X^{(S')}_{(N1)} & \textbf{(5)} \\ C^{(S)}_{(1,N_1)} &= C^{(S')}_{(1,N_1)} & \textbf{(6)} \\ C^{(S)}_{(2,N_1)} &= C^{(S')}_{(2,N_1)} & \textbf{(7)} \end{cases}$$

alors $C^{(S_2)}_{max} \leq C^{(S'_2)}_{max}$ **(8)**

Preuve. En effet, si : $C^{(S_2)}_{max} \leq C^{(S'_2)}_{max} \Leftrightarrow X^{(S_2)} \leq X^{(S'_2)}$

et $\begin{cases} X^{(S_2)} &= \psi^{(S_2)}_k + X^{(S_2)}_{(j,N_1)} + X^{(S_2)}_{(N_2)} \\ X^{(S'_2)} &= \psi^{(S'_2)}_k + X^{(S'_2)}_{(j,N_1)} + X^{(S'_2)}_{(N_2)} \end{cases}$

Cela implique que :

$$\begin{cases} \psi^{(S_2)}_k &= \psi^{(S'_2)}_k & \textbf{(a)} \\ X^{(S_2)}_{(j,N_1)} &= X^{(S'_2)}_{(j,N_1)} & \textbf{(b)} \\ X^{(S_2)}_{(N_2)} &\leq X^{(S'_2)}_{(N_2)} & \textbf{(c)} \end{cases}$$

Autrement dit, pour montrer l'inéquation (8), il suffit de montrer les trois dernières équations/inéquations (a), (b), et (c).

(a). Par définition, nous avons

$$\begin{cases} X^{(S)}_{(N_1)} = \psi^{(S)}_j + X^{(S)}_{(j,N_1)} \\ X^{(S')}_{(N_1)} = \psi^{(S')}_j + X^{(S')}_{(j,N_1)} \end{cases}$$

Selon les hypothèses (4) et (5): $\Rightarrow \psi^{(S)}_j = \psi^{(S')}_j$.

De plus, avec $\sigma_{(j,N_1)} = \sigma^{(S)}_{(j,N_1)} = \sigma^{(S')}_{(j,N_1)}$ (hypothèse (2) et (3)) et les hypothèses (4), (6) et (7), nous avons $S^{(S)}_{(1,j)} = S^{(S')}_{(1,j)}$ et $S^{(S)}_{(2,j)} = S^{(S')}_{(2,j)}$.

$$\Rightarrow \begin{cases} S^{(S_2)}_{(1,k)} &= S^{(S)}_{(1,j)} &= S^{(S')}_{(1,j)} &= S^{(S'_2)}_{(1,k)} & \textbf{(d)} \\ S^{(S_2)}_{(2,k)} &= S^{(S)}_{(2,j)} &= S^{(S')}_{(2,j)} &= S^{(S'_2)}_{(2,k)} & \textbf{(e)} \end{cases}$$

Donc, $X^{(S_2)}_{(k)} = X^{(S'_2)}_{(k)}$.

D'autre part, nous avons : $\psi^{(S_2)}_k = \psi^{(S)}_j + X^{(S_2)}_{(k)}$ et $\psi^{(S'_2)}_k = \psi^{(S')}_j + X^{(S'_2)}_{(k)}$. Donc : $\psi^{(S_2)}_k = \psi^{(S'_2)}_k$.

(b). Selon (d) et (e), nous avons $Y^{(S_2)}_k = Y^{(S'_2)}_k$. De plus, selon la propriété (*2*) avec $\sigma^{(S_2)}_{(k,N_1)} = (J_k, \sigma) = \sigma^{(S'_2)}_{(k,N_1)}$. Nous pouvons déduire donc : $X^{(S_2)}_{(j,N_1)} = X^{(S'_2)}_{(j,N_1)}$.

(c). À partir de S (resp. S'), nous définissons S_1 (resp. S'_1) en tant qu'une séquence transitoire pour laquelle le travail J_k n'est pas encore déplacé; soit J_l le travail ordonnancé après J_k dans S (et donc aussi dans S' selon l'hypothèse (3)).

Ainsi par l'hypothèse (3), nous avons : $\sigma^{(S)}_{(k,N_2)} = \sigma^{(S')}_{(k,N_2)} \Rightarrow S^{(S)}_{(1,k)} = S^{(S')}_{(1,k)}$. Si $S^{(S)}_{(2,k)} \geq S^{(S')}_{(2,k)}$, alors nous avons : $C^{(S)}_{max} \geq C^{(S')}_{max}$ (ce qui est contradictoire avec l'hypothèse (1)). Nous avons donc : $S^{(S)}_{(2,k)} < S^{(S')}_{(2,k)}$. Ainsi : $C^{(S)}_{(2,k)} < C^{(S')}_{(2,k)}$. Donc, selon l'hypothèse (5) : $X^{(S)}_{(N_1)} = X^{(S')}_{(N_1)}$, alors $X^{(S)}_{(N_1,k)} < X^{(S')}_{(N_1,k)}$. Ainsi, nous avons $X^{(S_1)}_{(N_1,l)} < X^{(S'_1)}_{(N_1,l)}$. **(f)**

À partir de (a), et (b), on montre que $C^{(S_2)}_{(1,N_1)} = C^{(S'_2)}_{(1,N_1)}$ et $C^{(S_2)}_{(2,N_1)} = C^{(S'_2)}_{(2,N_1)}$.

Selon corollaire de la propriété (*2*) avec $(d_1 - d_2) =$

$C^{(S_1)}_{(2,N_1)} - C^{(S_1)}_{(1,N_1)} = C^{(S'_1)}_{(2,N_1)} - C^{(S'_1)}_{(1,N_1)}$ et $(d_3 - d_4) = S^{(S_2)}_{(2,N_1)} - C^{(S_2)}_{(1,N_1)} = C^{(S'_2)}_{(2,N_1)} - C^{(S'_2)}_{(1,N_1)}$, nous déduisons que $X^{(S_2)}_{(N_1,l)} \leq X^{(S'_2)}_{(N_1,l)}$.

Selon les résultats (a),(b) et $\psi^{(S_2)}_l = \psi^{(S_2)}_k + X^{(S_2)}_{(j,N_1)} + X^{(S_2)}_{(N_1,l)}$ (de même pour S'_2), nous avons donc $\psi^{(S_2)}_l \leq \psi^{(S'_2)}_l$.

L'inéquation est toujours vérifiée dans le cas où une sous-sequence σ est ajoutée à la fin de la sous-séquence considérée (i.e. après la fin du dernier travail de la sous-séquence actuelle et avant le travail J_l selon les séquences S_2 et S'_2).

Ces mouvements seront donc effectués afin d'avoir $\sigma^{S_2}_{(l,N_1)} = \sigma$. Selon l'hypothèse (3), σ est aussi identique à $\sigma^{S'_2}_{(l,N_1)}$. Par conséquence, $X^{(S_2)}_{(N_2)} \leq X^{(S'_2)}_{(N_2)}$.□

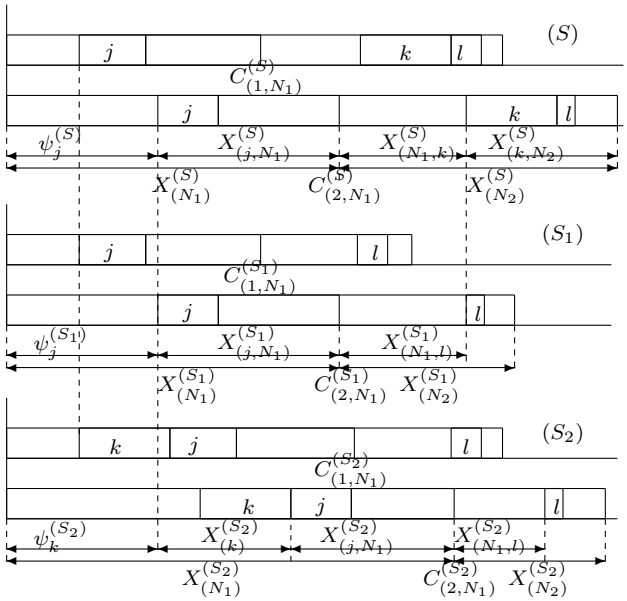


Figure 4: Propriété 3

Propriété 4

Considérons ici le même contexte que celui présenté pour la propriété 3, à l'exception de la condition **(1)**, nous supposons que : $C^{(S)}_{max} = C^{(S')}_{max}$.

Nous allons donc comparer les deux valeurs de $C^{(S_2)}_{max}$ et de $C^{(S'_2)}_{max}$.

Notons qu'on peut déduire les mêmes résultats pour les équations **(a)**, **(b)**, mais pas pour **(c)**. Pour déterminer la relation entre les deux valeurs du makespan définies respectivement par S et S', nous analysons la relation qui existe entre $X^{(S)}_{(k,N2)}$ et $X^{(S')}_{(k,N_2)}$.

Supposons que $X^{(S)}_{(k,N_2)} = X^{(S')}_{(k,N_2)}$. Alors, $X^{(S)}_{(N_1,k)} = X^{(S')}_{(N_1,k)}$. Donc, $X^{(S_1)}_{(N_1,l)} = X^{(S'_1)}_{(N_1,l)}$.

Selon la propriété (*2*), nous avons donc : $X^{(S_2)}_{(N_1,l)} = X^{(S'_2)}_{(N_1,l)}$. Par conséquent, l'ajout de la même séquence à la fin de la séquence actuelle n'aura aucune conséquence sur la valeur du makespan, i.e. $C^{(S_2)}_{max} = C^{(S'_2)}_{max}$.

Dans le cas où : $X^{(S)}_{(k,N_2)} > X^{(S')}_{(k,N_2)} \Rightarrow X^{(S)}_{(N_1,k)} < X^{(S')}_{(N_1,k)}$. Nous avons donc : $X^{(S_1)}_{(N_1,l)} < X^{(S'_1)}_{(N_1,l)}$.

Suite la démonstration similaire à partir de (f) de propriété *3*, on obtiendra $X^{(S_2)}_{(N_1,l)} \leq X^{(S'_2)}_{(N_1,l)}$ et puis $C^{(S_2)}_{max} \leq C^{(S'_2)}_{max}$. On peut donc déduire que la séquence S est meilleure.

Dans le cas contraire, c-à-d $X^{(S)}_{(k,N_2)} < X^{(S')}_{(k,N_2)} \Rightarrow C^{(S_2)}_{max} \geq C^{(S'_2)}_{max}$. C'est la séquence S' qui est meilleure.

En résumé, dans le cas de $C^{(S)}_{max} = C^{(S')}_{max}$ avec les conditions **(2), (3), (4), (5), (6) et (7)** définies précédemment, si : $X^{(S)}_{(k,N_2)} \geq X^{(S')}_{(k,N_2)}$ alors la séquence S est meilleure.□

Toutes ces propriétés restent également valides pour le cas $F2|C^N_{max} \leq \epsilon|C^{N_1}_{max}$. Par conséquent, nous pouvons déduire un algorithme permettant de calculer un ordonnancement optimal en $O(n^2\epsilon^4)$.

4.4. ALGORITHME PSEUDO-POLYNOMIAL

D'abord, et selon la propriété 1, on détermine une solution initiale S_0. Si $C^{(S_0)}_{(N_1)} > \varepsilon$ alors il n'existe pas de solution satisfait la contrainte *varepsilon* ; si $C^{(S_0)}_{(N_1)} = \varepsilon$ alors la solution S_0 est optimale ; sinon, il existe une solution optimale définie par le déplacement d'un sous-ensemble de travaux de N_2 dans la première partie de l'ordonnancement avant *varepsilon*. Les travaux à déplacer sont ordonnancés selon la propriété 1.

Pour déterminer l'ensemble des travaux à déplacer, on procède de la manière suivante :

Les travaux de l'ensemble N_2 sont rangés dans l'ordre de Johnson. Une solution courante est représentée par le 5-tuplet suivant : $(t,\ X^{(S)}_{(j,N_1)},\ X^{(S)}_{(N_1)},\ C^{(S)}_{(1,N_1)},\ C^{(S)}_{(2,N_1)})$ où t est l'indice du travail de N_2 à insérer parmi les travaux de N_1 en respectant la propriété 1 ; J_j est le premier travail de l'ensemble N_1 précède immédiatement J_t.

Ainsi, pour chaque travail J_t de N_2, on calcule le 5-tuplet qui sera stocké. Remarquons que les valeurs de $X^{(S)}_{(j,N_1)}$, $X^{(S)}_{(N_1)}$, $C^{(S)}_{(1,N_1)}$ et $C^{(S)}_{(2,N_1)}$ sont inférieures ou égales à ε. Donc, le nombre des solutions déterminées à chaque itération est borné par ε^4.

Pour chaque solution S de l'itération t stockée, à l'itération $t+1$ qui correspond au déplacement d'un autre travail de l'ensemble N_2, noté J_{t+1}, on met à jour le deuxième argument du 5-tuplet $X^{(S)}_{(j',N_1)}$ où $J_{j'}$ correspond au travail de N_1 qui précède immédiatement le travail J_{t+1}. Ainsi, de nouvelles solutions S_1 sont créées à partir de l'ensemble des solutions S définies par le déplacement du travail $t+1$.

Seules les solutions faisables nouvellement créées non dominées de S_1 seront stockées. Une solution S'_1 domine S_1 si le 5-tuplet de S'_1 est identique au 5-tuplet de S_1 avec $C^{S'_1}_{max} < C^{S_1}_{max}$ (suite à la propriété 4); ou si $C^{S'_1}_{max} = C^{S_1}_{max}$ et $X^{(S'_1)}_{\sigma} \geq X^{(S_1)}_{\sigma}$ (suite à la propriété 5), σ représente la sous séquence des travaux de N_2 ordonnancés selon Johnson après le travail J_{t+1}.

Toutes les solutions stockées sont, par conséquent, toujours non dominées. Nous avons donc, $|N_2| < n$ itérations au total. En outre, à l'itération $(t+1)$, pour chaque séquence S trouvée à l'itération t, nous devons calculer les valeurs suivantes : $X^{(S)}_{(j',N_1)}$, et $X^{(S)}_{\sigma'}$. Ce calcul s'effectue en $O(n)$. Ainsi, le temps de calcul de l'algorithme est bornée par $O(n^2\varepsilon^4)$ (cf. l'algorithme de la figure 5).

5. CONCLUSION

Dans cet article, on s'intéresse à l'ordonnancement des travaux interférants dans un flowshop à deux machines. L'étude de complexité montre la NP-complétude au sens ordinaire du cas $\varepsilon(C^{(N_1)}_{\max}/C_{\max})$. Un algorithme pseudo-polynomial est proposé pour résoudre les deux problèmes considérés en $O(n^2\epsilon^4)$. Le premier problème correspond au cas où on cherche à minimiser la date d'achèvement des travaux sous la contrainte que le makespan d'un sous-ensemble donné soit optimal et inversement pour le second. Dans le cas où aucune marge sur la date de fin du dernier travail d'un sous-ensemble n'est tolérée ($C^{(N_1)}_{\max} = \varepsilon$), la minimisation du C_{max} global sous cette contrainte peut être déterminé en $O(n\log(n))$.

La recherche du futur est d'étudier ce type de problème d'ordonnancement en considérant d'autres critères réguliers, mais leur NP-complétude est acquise d'avance. Les cas de machines parallèles seront également à étudier.

RÉFÉRENCES

[Agnetis et al., 2000] Agnetis A., Mirchandani P.B., Pacciarelli D., et Pacifici A. (2000). Nondominated schedules for a job-shop with two competing users. *Computational & Mathematical Organization Theory* **6**, pp. 191–217.

[Agnetis et al, 2004] Agnetis A., Mirchandani P.B., Pacciarelli D., et Pacifici A. (2004). Scheduling problems with two competing agents. *Operations Research* **52**, pages 229–242.

[Baker et Smith, 2003] Baker K., et Smith J.C. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling* **6**, pp. 7–16.

[Brucker et Knust, 2006] Brucker P., Knust S. (2006) Complexity results for scheduling problems. *http://www.mathematik.uni-osnabrueck.de.*

[Ehrgott, 2005] Ehrgott M. (2005) *Multicriteria Optimization.* 2nd edition, Springer, Berlin-Heidelberg.

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

[Garey et al, 1976] Garey M.R., Johnson D.S., et Sethi R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research* **1**, pages 117–129.

[Garey et Johnson, 1979] Garey M.R. et Johnson D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman & Company, San Francisco.

[Hoogeveen, 2005] Hoogeveen H. (2005). Multicriteria Scheduling. *European Journal of Operational Research* **167**, pages 592–623.

[Johnson, 1954] Johnson S.M. (1954). Optimal two-and-three-stage production schedules with set-up times included. *Naval Research Logistics Quarterly* **1**, pages 61–68.

[Lee, 1997] Chung-Yee Lee (1997) Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters* **20**, pages 129–139.

[Lenstra et al, 1977] Lenstra J.K., Rinnooy Kan A.H.G., et Brucker P. (1977) Complexity of machine scheduling problems. *Annals of Discrete Mathematics* **1** 343–362.

[T'kindt et Billaut, 2006] T'kindt V. et Billaut J-C. (2006) *Multicriteria Scheduling: Theory, Models and Algorithms.* 2nd edition, Springer.

Algorithm $F2|C_{max}^{(1)} \leq \epsilon|C_{max}$

Soit σ_1 l'ensemble des travaux de $N = N1 \cup N2$ ordonné selon Johnson
Soit σ_2 (resp. σ_3) l'ensemble des travaux de $N1$(resp. $N2$) ordonné selon Johnson
$S_0 = (\sigma_2\sigma_3)$
Calculer $C_{(2,N1)}^{(S_0)}$ et $C_{max}^{(S_0)}$
Si $(C_{(2,N1)}^{(S_0)} > \epsilon)$ **alors**
  S^* = NULL *//Il n'y a pas de solution faisable*
  C^* = -1
Si $(C_{(2,N1)}^{(S_0)} = \epsilon)$ **alors**
  $S^* = S_0$ *//La solution initiale est optimale*
  $C^* = C_{max}^{(S_0)}$
Sinon *//*$C_{(2,N1)}^{(S_0)} < \epsilon$
  $S^* = S_0$
  $C^* = C_{max}^{(S_0)}$
  $F(k, t_1, t_2, t_3, t_4) = C_{max}^{(S_0)} + 1$, $\forall k \in 0...|N2|$ et $t_1, t_2, t_3, t_4 \in 0...\epsilon$
  $F(0, 0, 0, 0, 0) = C_{max}^{(S_0)}$
  Pour $k = 1$ **à** $|N2|$ **faire** *//Considérons J_k le travail d'indice k*
    Déterminer J_j : le premier travail $\in N1$ et ordonnancé après J_k selon Johnson sur l'ensemble N.
    Pour chaque tuplet $(t_1, t_2, t_3, t_4) = (0, 0, 0, 0)$ **à** $(\epsilon, \epsilon, \epsilon, \epsilon)$ **faire**
      Si $F(k, t_1, t_2, t_3, t_4) \leq C_{max}^{(S_0)}$ **alors**
        Définir la séquence S' en ordonnançant J_k selon Johnson
        Calculer le 5-tuplet correspondant à S': $(k, t'_1, t'_2, t'_3, t'_4) = (k, X_{(j,N1)}^{(S')}, X_{(N1)}^{(S')}, C_{(1,N1)}^{(S')}, C_{(2,N1)}^{(S')})$
        Si $C_{(2,N1)}^{(S')} \leq \epsilon$ **alors**
          S'il existe déjà une solution au tuplet $(k, t'_1, t'_2, t'_3, t'_4)$ **alors**
            Si $C_{(2,N1)}^{(S')} < F(k, t'_1, t'_2, t'_3, t'_4)$ **alors** *//selon la propriété 3*
              Stocker S' au tuplet $(k, t'_1, t'_2, t'_3, t'_4)$
            Sinon
              Si $C_{(2,N1)}^{(S')} = F(k, t'_1, t'_2, t'_3, t'_4)$ **alors** *//selon la propriété 4*
                Soit S_1 la solution trouvée à partir du tuplet $(k, t'_1, t'_2, t'_3, t'_4)$
                Selon l'ordre de Johnson, déterminer J_l le premier travail de $N2$ ordonnancé après J_k
                Si $X_{(l,N2)}^{(S')} \geq X_{(l,N2)}^{(S1)}$ **alors**
                  Stocker S' au tuplet $(k, t'_1, t'_2, t'_3, t'_4)$
                FinSi
              FinSi
            FinSi
          Sinon *//Il n'existait aucune solution à ce tuplet*
            Stocker S' au tuplet $(k, t'_1, t'_2, t'_3, t'_4)$
          FinSi
        FinSi
        Si $C^* > C_{max}^{(S')}$ **alors** *//Mettre à jours la meilleure solution temporaire*
          $C^* = C_{max}^{(S')}$
          $S^* = S'$
        FinSi
      FinSi
    FinBoucle
  FinBoucle
FinSi
Retourner S^* et C^*

Figure 5: Un algorithme pseudo-polynomial pour le $F2|C_{max}^{(1)} \leq \epsilon|C_{max}$