Ré-ordonnancement des systèmes de production flexibles avec contraintes de blocage mixtes soumis à des aléas de commandes ou de production

Ayoub Tighazoui

École doctorale IAEM-Lorraine

(Informatique, Automatique, Électronique-Électrotechnique et Mathématiques)

Thèse

Présentée pour l'obtention du grade de

Docteur de l'Université de Lorraine

Spécialité Automatique, Traitement du Signal et des Images, Génie Informatique

par

Ayoub TIGHAZOUI

Ré-ordonnancement des systèmes de production flexibles avec contraintes de blocage mixtes soumis à des aléas de commandes ou de production

---

Soutenue publiquement, le 12 novembre 2021 devant le jury composé de :

Rapporteurs :	Alice YALAOUI, Maître de conférences HDR, Université de Technologie de Troyes, France
	Ameur SOUKHAL, Professeur, Université de Tours, France
Examinateurs :	Christelle GUERET, Professeur, Université d'Angers, France (Présidente de jury)
	Sylvie NORRE, Professeur, Université Clermont Auvergne, France
Directrice de thèse :	Nathalie SAUER, Professeur, Université de Lorraine, France
Co-directeur de thèse :	Christophe SAUVEY, Maitre de conférences HDR, Université de Lorraine, France

Dédicaces

À ma mère, source de vie

À mon père, source d'amour

À ma sœur, source de joie

À toute ma famille et mes amis

À tous mes collègues de travail

À mes enseignants depuis la maternelle

À tous ceux qui me sont chers et que j'ai omis de citer

À vous chers lecteurs

Remerciements

Cette thèse est le fruit d'un travail qui a été supporté et encouragé par de nombreuses personnes auxquelles je tiens à exprimer ma gratitude.

Mes premiers remerciements vont tout naturellement à mes deux directeurs de thèse. Je tiens à remercier Professeur **Nathalie SAUER** de m'avoir fait confiance et de m'avoir sélectionné pour réaliser cette thèse. Cette opportunité est pour moi un atout d'une valeur inestimable. Je la remercie également pour sa bienveillance, pour sa patience, sa rigueur, et sa sagesse. Je remercie Dr HDR **Christophe SAUVEY** pour ses encouragements, ses conseils précieux, et son soutien moral. Il a toujours trouvé les mots pour me donner confiance en moi et pour me guider dans l'acquisition de mes compétences. Mes deux directeurs de thèse ont, tout au long de cette période, été disponibles pour m'aider à faire avancer le travail, à corriger mes travaux, à écouter mes propositions, à me proposer les axes d'amélioration les plus pertinents, et à me guider sur le plan scientifique et humain. Je vous remercie pour votre suivi et votre supervision de cette thèse. Sachez que sans vous la thèse ne se déroulerait pas dans de si bonnes conditions.

Mes remerciements s'adressent aussi au Professeur **Nidhal REZG**, Directeur du LGIPM, de m'avoir accueilli si chaleureusement dans le laboratoire depuis mon stage de recherche en Master. Il a toujours su veiller sur le bon déroulement de mon travail au LGIPM tant sur le plan scientifique qu'administratif. Je remercie également Mme **Christel WIEMERT**, responsable administrative du LGIPM, pour son accueil chaleureux et pour l'attention qu'elle dédie, en particulier, aux nouveaux stagiaires et doctorants. Mes vifs remerciements s'adressent aussi au Docteur **Sadok TURKI** qui m'a co-encadré avec le Pr. Nathalie SAUER dans mon stage d'initiation à la recherche en Master. Je le remercie pour son encadrement et pour son soutien. J'adresse ma gratitude aussi au Dr HDR **Sofiene DELLAGI** et Dr **Hoai Minh LE** d'avoir été membre du comité de suivi de cette thèse, et d'avoir participé au progrès de ce travail par leurs propositions pertinentes. Je remercie aussi Dr **Jérémie SCHUTZ** avec qui j'ai collaboré lors de mes missions d'enseignement, ainsi que Dr HDR **Didier Anciaux** notre chef d'équipe, tout comme mes collègues du laboratoire, **Nadia NDHAIEF**, **Zouhour GUIRAS**, **Zakaria CHEKOUBI**, **Aminu Sahabi ABUBAKAR**, **Ronald Mauricio MARTINOD RESTREPO**, **Sadok REZIG**, **Mohamed Ali KAMMOUN**, et bien d'autres, pour leur soutien et leur convivialité.

Ensuite, je tiens à exprimer mes remerciements aux membres du jury qui ont accepté d'évaluer ce travail de thèse. Je remercie Dr HDR **Alice YALAOUI** et Professeur **Ameur SOUKHAL** pour le temps qu'ils ont bien voulu consacrer en acceptant d'être rapporteurs. Je remercie également les professeurs **Christelle GUERET** et **Sylvie NORRE** pour avoir accepté d'examiner ce travail de recherche.

Pendant cette période de thèse, le soutien de mes amis ne manquait jamais. Ils étaient toujours à mes côtés pour donner leurs encouragements, leur réconfort, et leur aide. Leur soutien est l'un des piliers de la réussite de cette aventure. Je pense à **Hicham JABRAOUI**, **Aamre KHALIL**, **Ilyes KADRI** et **Rida FARHAN**.

Enfin, j'adresse mes remerciements les plus chaleureux à mes parents qui m'ont aimé depuis le jour où j'ai ouvert les yeux. Je suis fier de leur montrer que leur sacrifice a porté ses fruits. Je remercie également ma sœur **Meriem** qui m'a toujours soutenu, ainsi que tous les membres de ma famille.

Ces remerciements sont une forme de reconnaissance envers toutes les personnes qui ont contribué de près comme de loin à l'aboutissement de cette thèse.

Ayoub TIGHAZOUI

Liste des Publications

Articles publiés dans des revues internationales

[AP1] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Predictive-reactive Strategy for Flowshop Rescheduling Problem: Minimizing the Total Weighted Waiting Times and Instability. *Journal of Systems Science and Systems Engineering*. 30, 253-275.

[AP2] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Predictive-reactive strategy for identical parallel machine rescheduling. *Computers & Operations Research*, 105372.

[AP3] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Minimizing the Total Weighted Waiting Times and Instability in a Rescheduling Problem with Dynamic Jobs Weight. *Applied Sciences* 11, no. 15: 7040. https://doi.org/10.3390/app11157040.

Article rédigé à soumettre dans une revue internationale

[AS1] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Heuristics for flow shop rescheduling with mixed blocking constraints.

Articles publiés dans des conférences internationales

[CP1] Tighazoui, A., Sauvey, C., & Sauer, N. (2020). New efficiency-stability criterion in a rescheduling problem with dynamic jobs weights. *7th International Conference on Control, Decision and Information Technologies (CoDIT)* (Vol. 1, pp. 475-480). IEEE.

[CP2] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). New efficiency-stability criterion for flow shop rescheduling problem with mixed blocking constraints. *17th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2021,* Budapest, Hungary.

[CP3] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Heuristics based on MILP for solving a flow shop rescheduling problem simultaneously considering efficiency and stability. *17th International Conference on Automation Science and Engineering, CASE 2021,* IEEE, Lyon, France.

[CP4] Tighazoui, A., Sauvey, C., & Sauer, N. (2020). Predictive-reactive strategy for operating rooms rescheduling. *Smart Health International Conference 2020, SHEIC 2020*, Troyes, France

Chapitre de livre soumis pour publication dans une revue internationale

[CL1] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Operating rooms rescheduling with a predictive-reactive strategy. Wiley Book: "*Recent Advances in Smart Healthcare*".

Résumé

La transformation numérique que vit l'entreprise aujourd'hui a définitivement changé le comportement des consommateurs. En effet, via les nouveaux outils d'information et de communication, un client peut désormais à tout moment créer, modifier ou annuler une commande. Ces évènements fortuits ont un impact sur l'organisation au sein des unités de production et engendrent une perturbation de l'ordonnancement planifié. Par conséquent, un processus de réordonnancement est nécessaire pour réviser efficacement le planning déjà établi, de préférence avec le moins de modifications possibles. Cette thèse propose des modèles mathématiques et des méthodes d'optimisation pour résoudre des problèmes de réordonnancement, afin de parer des perturbations dans des environnements machines différents, et évaluer la performance des solutions obtenues, grâce à un critère combinant l'efficacité et la stabilité de l'ordonnancement.

La stratégie prédictive-réactive a été adoptée dans cette étude. Elle consiste, dans la phase prédictive, à résoudre un problème d'ordonnancement classique ayant comme objectif de minimiser la somme des temps d'attente des jobs pondérés par leurs poids, représentant le critère d'efficacité. Dès l'apparition d'une perturbation, vient ensuite la phase réactive, qui consiste à mettre à jour le problème initial en modifiant ses données, puis à résoudre le nouveau problème de réordonnancement, ayant cette fois comme objectif de minimiser le critère d'efficacité décrit auparavant, combiné avec le critère de stabilité. Ce dernier est défini par la somme des différences entre les dates de fin des jobs avant et après l'apparition de la perturbation, pondérées par le poids des jobs. Cette combinaison des deux critères est significative, et peut être très utile dans des applications industrielles, où le temps d'attente des jobs représente le temps d'attente des produits devant le poste de travail, tandis que les poids des jobs représentent l'importance des clients. La mesure de stabilité permet de limiter la déviation par rapport au planning déjà établi, car celle-ci engendre des coûts supplémentaires.

Cette approche a été appliquée à différents environnements de machines, en commençant par une machine unique, représentant un seul poste de travail, puis, sur des machines parallèles représentant des postes de travail identiques. Et enfin, dans un atelier de type Flowshop, où l'ensemble des jobs doivent passer sur un ensemble de machines dans le même ordre. Ce dernier cas a aussi été étudié en considérant des contraintes de blocage mixtes entre les machines.

Ces différents problèmes de réordonnancement ont été modélisés en premier lieu sous forme de modèles mathématiques, adaptés à une Programmation Linéaire en Nombres Entiers (*PLNE*). Cependant, leur complexité *NP-difficile* n'a permis leur résolution que pour un nombre limité de jobs. Par conséquent, des méthodes heuristiques ont été développées, permettant de parcourir plus de jobs en un temps raisonnable. Les performances des méthodes développées ont été discutées et analysées, à la fois en termes de qualité de solution et de temps de calcul.

Mots clés

Réordonnancement, Stratégie prédictive-réactive, Efficacité, Stabilité, Contraintes de blocage mixtes, Heuristiques, Modélisation mathématique.

Abstract

Today's digital transformation has definitely changed the customers practices. In fact, through the new information and communication tools, a customer can at any time create, modify, or cancel an order. These unexpected events have a direct impact on the work organization of the production unit, generating a disruption of the already established schedule. Therefore, a rescheduling process is necessary for efficiently revising the existing schedule, preferably with less movements. In this Ph.D, mathematical models and optimization methods are developed for rescheduling problems, under different types of disruptions, in several machine environments. The performance of the obtained solutions is measured with a new criterion, simultaneously combining the schedule efficiency and stability.

The predictive-reactive scheduling strategy has been adopted in this work. It consists, in the predictive phase, to solve a classical scheduling problem minimizing the *Total Weighted Waiting Time* (*TWWT*) of the jobs, regarded as the schedule efficiency criterion. After the disruption appearance, the reactive phase starts. It consists in updating the initial problem by modifying its data, then solving the new rescheduling problem with the objective of minimizing the *TWWT* combined with a stability criterion. The schedule stability is measured with the *Total Weighted Completion Time Deviation* (*TWCTD*). This association of criteria is significant, and it can be very helpful in industrial applications, where the job waiting time estimates the duration that the job has waited in front of a workstation. The stability criterion is then used for limiting the deviation from the already established schedule, since this matter generates supplementary costs.

This approach has been applied for different machine environments. Firstly, on a single machine, illustrating the case of a single workstation. Secondly, on parallel machines, describing the case of identical workstations. Finally, on a flowshop system where a set of jobs are treated in the same order by a set of machines. The flowshop rescheduling problem is also considered with mixed blocking constraints.

These rescheduling problems have firstly been modeled as *Mixed Integer Linear Programing* (*MILP*) models. Due to their *NP-hard* complexity, the resolution is only possible for a limited number of jobs. Thus, heuristic methods have been designed, exploring more jobs in a reasonable time. The proposed methods have been discussed and analyzed, both in terms of solution quality and computing time.

Keywords

Rescheduling, Predictive-reactive strategy, Efficiency, Stability, Mixed blocking constraints, Heuristics, Mathematical modelling.

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

DMDFP :	Déviation Moyenne des Dates de Fin Pondérée.
DMI :	Durée Maximale d'une Itération.
EDD :	Earliest Due Date.
ER :	Erreur Relative.
ET :	Ecart Type.
FIFO :	First In First Out.
JRV :	Jobs Release Variation.
M-SPT :	Modified Shortest Processing Time.
M-LPT :	Modified Longest Processing Time.
MER :	Moyenne des Erreurs Relatives.
MILP :	Mixed Integer Linear Programing.
NEH :	Nawaz, Enscore, and Ham.
NJCM :	Nombre de Jobs qui Changent de Machine après le réordonnancement.
NMJOL :	Nombre Moyen des Jobs Optimisés par itération à travers la Lexicographie.
NSGA :	Non-dominated Sorting Genetic Algorithm.
NTJA :	Nombre Total des Jobs Arrivés.
NTJR :	Nombre Total des Jobs Retirés.
PLNE :	Programmation Linéaire en Nombres Entiers.
RCb :	Release when Completing blocking.
RLGV :	Recherche Locale à Grand Voisinage.
RSb :	Release when Starting blocking.
SPT :	Shortest Processing Times.
TAMP :	Temps d'Attente Moyen Pondéré.
TEM :	Temps d'Ecoulement Moyen
TLBO :	Teaching-Learning Based Optimization.
TR :	Temps de Résolution.
TWCTD :	Total Weighted Completion Times Deviation.
TWWT :	Total Weighted Waiting Times.
Wb :	Without blocking.
wSPT :	Weighted Shortest Processing Times.
wSRPT :	Weighted Shortest Remaining Processing Times.
SRPT :	Shortest Remaining Processing Time.

Introduction générale

Ces dernières décennies ont été marquées par une progression constante des nouvelles technologies d'information et de communication au sein des entreprises, à savoir la transformation numérique. Celle-ci a définitivement changé l'interaction entre les clients et l'entreprise. Désormais, un client peut à tout moment créer, modifier ou annuler sa commande via des nouveaux outils technologiques, notamment les applications des téléphones mobiles ou les plateformes web. En revanche, pour rester compétitives sur le marché, les entreprises doivent faire face à ces aléas de commande. En effet, l'introduction des e-commandes au sein de leurs services a des conséquences sur l'ordonnancement. Bien qu'un ordonnancement traditionnel permette d'organiser valablement les systèmes de production de biens et de services, ses hypothèses deviennent invalides lors de l'apparition d'une nouvelle commande. La possibilité de créer, de modifier, ou d'annuler une commande en temps réel change les approches adoptées pour ce processus. En outre, les problèmes de production peuvent également avoir un impact sur l'ordonnancement établi. Les pannes des machines ou le retard de l'arrivée de matières premières sont aussi des évènements fortuits qui peuvent empêcher la réalisation de certaines opérations. Les planificateurs sont par conséquent obligés de très rapidement réagir pour ne pas arrêter complètement la production.

Face à ces environnements dynamiques, la réactivité de l'ordonnancement devient un enjeu majeur. En effet, Les décideurs doivent non seulement générer un ordonnancement efficace, mais également très vite réagir face aux perturbations en le mettant à jour. Ce processus de révision d'un ordonnancement existant face aux aléas de commandes ou de production, dit réordonnancement, préoccupe les preneurs de décisions au sein des entreprises, mais aussi il constitue une thématique intéressante pour les chercheurs scientifiques qui lui dédient une attention particulière. Vieira *et al.*, (2003) est parmi les célèbres travaux à avoir apporté une définition détaillée de ce processus et avoir décrit ses approches et ses méthodes. Les auteurs ont défini le réordonnancement comme étant le processus de mise à jour d'un ordonnancement existant en réponse à des perturbations ou d'autres changements. Ceci inclut l'arrivée de nouveaux jobs, l'annulation des jobs, les pannes des machines, le manque de matières premières, etc. Dans la littérature, la performance d'un ordonnancement est souvent mesurée par des critères classiques d'efficacité, à savoir le Makespan, la moyenne des dates de fin, le retard moyen ou le retard maximal, etc. Or, lorsqu'il s'agit d'étudier des problèmes de réordonnancement, d'autres mesures peuvent être considérées. C'est le cas des mesures de stabilité qui permettent de limiter la déviation par rapport à l'ordonnancement existant. En effet, la modification d'un planning déjà établi due à une perturbation peut générer des coûts supplémentaires, incluant les coûts de réaffectation des outils et des équipements, ou bien les coûts de réapprovisionnement des matières premières. A cet égard, la performance de l'ordonnancement ne sera pas évaluée qu'à l'aune de son efficacité, mais aussi en termes de stabilité, c'est-à-dire agir efficacement avec le moins de mouvements possibles. Pour avoir un compromis entre ces deux aspects influents, il est judicieux de les considérer simultanément.

Dans le cadre de cette thèse, une nouvelle mesure de performance a été définie et étudiée. Celle-ci combine simultanément l'efficacité et la stabilité de l'ordonnancement. Premièrement, en termes d'efficacité de l'ordonnancement, le temps d'attente moyen pondéré par les poids des jobs a été considéré comme mesure. Le temps d'attente représente la durée d'attente du job

avant son exécution par l'outil de production à partir de l'instant où il est disponible. Cette mesure d'efficacité est très significative et utile dans plusieurs applications. Dans les systèmes de production, celle-ci correspond à l'attente des jobs devant un poste de travail ou des produits pour les clients, en considérant les poids comme étant les priorités des clients. Dans les systèmes hospitaliers, celle-ci correspond à l'attente des patients avant d'être opérés, en considérant les poids comme étant les niveaux d'urgence des patients. Deuxièmement, en termes de stabilité de l'ordonnancement, la déviation moyenne des dates de fin pondérées par les poids des jobs a été considérée comme mesure. Ce critère évalue la différence entre la date de fin d'un job lorsqu'il est ordonnancé la première fois et la date de fin de celui-ci après l'apparition de la perturbation, les poids permettent alors de pénaliser encore plus les mouvements des jobs prioritaires. En effet, lorsque que le job est planifié pour la première fois, la date peut être transmise au client ou au patient et il est donc important de ne la modifier le moins possible. De plus, l'ordre dans lesquels les jobs sont planifiés permet d'organiser la production ou les opérations, et le fait de retarder un job peut désorganiser complètement le système. Ces deux critères décrits sont associés par un coefficient α, nommé coefficient d'*efficacité-stabilité* qui représente la pondération liée à chaque critère.

La stratégie prédictive-réactive est très courante dans les systèmes de réordonnancement, elle est basée sur deux principales étapes. La première consiste à générer un ordonnancement initial. Puis, dans la deuxième, de le mettre à jour à chaque fois qu'une perturbation arrive. Cette stratégie est itérative, et permet à chaque étape du processus de réordonnancement d'avoir une décision locale. Dans notre approche, cette stratégie a été adoptée, consistant dans la phase prédictive à résoudre un problème d'ordonnancement classique, ayant comme objectif de minimiser le critère d'efficacité seul, représenté par le temps d'attente moyen pondéré. A chaque arrivée d'une perturbation, intervient la phase réactive qui consiste à réordonnancer les jobs, ayant cette fois-ci comme objectif de minimiser les deux critères simultanément, le critère d'efficacité de l'ordonnancement décrit auparavant, et le critère de stabilité représenté par la déviation moyenne des dates de fin pondérées. Le fait d'associer le critère de stabilité au critère d'efficacité dans la phase réactive permet d'évaluer l'impact dû au changement du planning déjà établi, et de décider l'action à mener suite à cela. Par exemple, un nouveau job sera planifié en dernier dans une séquence si la modification de celle-ci génère un coût trop élevé.

Pour résoudre le problème décrit, des modèles mathématiques sous forme de Programmation Linéaire en Nombres Entiers (*PLNE*) ont été développés. Cependant, leur complexité *NP-difficile* a permis de les résoudre que pour un nombre limité de jobs. Par conséquent, des méthodes heuristiques ont été développées, permettant de parcourir plus de solutions en un temps raisonnable. Par ailleurs, ces problèmes ont été traités sur plusieurs environnements machines, en premier lieu, sur une machine unique représentant un seul poste de travail. Le problème de réordonnancement sur une machine unique est générique, et peut être utilisé comme une base pour d'autres problèmes plus compliqués dans des environnements contenant des machines multiples. En deuxième lieu, ils ont été traités sur un environnement de machines parallèles représentant des postes de travail identiques. Le problème de réordonnancement sur des machines parallèles est très souvent rencontré dans les applications réelles. En particulier, dans l'ordonnancement des blocs opératoires, où les opérations électives et urgentes doivent être planifiées dans des salles d'opérations identiques. Et enfin, dans un atelier de type Flowshop où l'ensemble des jobs doivent passer sur un ensemble de machines dans le même ordre. C'est le cas des lignes de production dans les usines. Ce dernier cas a été également

étudié en considérant des contraintes de blocages mixtes entre les machines, modélisant le cas où les capacités de stock entre les machines sont limitées.

Le premier chapitre de ce mémoire est dédié à la présentation et l'étude de l'état de l'art des travaux abordant les problèmes de réordonnancement. Dans un premier temps, une définition détaillée de ce concept est apportée, ainsi que ses différents champs d'application. Plusieurs termes liés à la thématique de réordonnancement sont ensuite présentés, à savoir les types de perturbations, les mesures de performance, les stratégies et les méthodes de réordonnancement. Une revue de la littérature des travaux liés au réordonnancement a été établie, avec pour objectif de positionner notre problématique par rapport aux travaux existants. Deuxièmement, une description détaillée du problème a été présentée, décrivant la nouvelle mesure de performance et son application dans les cas réels, ainsi qu'une explication de la stratégie prédictive-réactive adoptée pour les problèmes que nous allons étudier.

Dans le chapitre deux, nous avons abordé le problème de minimisation du temps d'attente moyen pondéré et de la déviation de la date de fin pondérée dans un problème de réordonnancement sur une machine unique, en considérant des poids variables. En premier lieu, le problème a été décrit, en précisant bien les motivations de ce travail. Ensuite, une formulation mathématique basée sur la PLNE a été conçue. Puis, dans la partie des résultats expérimentaux, nous avons analysé l'impact de la variation du coefficient d'efficacité-stabilité sur la performance de l'ordonnancement, ainsi que l'influence de la variation des poids des jobs sur le système. Enfin, le temps de résolution du modèle est étudié pour découvrir ses limites en termes de nombre de jobs.

Dans le troisième chapitre, nous avons étudié un problème de minimisation du temps d'attente moyen pondéré et de la déviation moyenne des dates de fin pondérée dans un problème de réordonnancement sur des machines parallèles. Comme dans le deuxième chapitre, nous avons commencé par une description du problème en motivant l'inspiration de notre idée. Nous avons ensuite fourni une nouvelle formulation mathématique adaptée à ce problème précis. La fonction objectif considérée dans ce cas est celle décrite auparavant, qui combine le critère d'efficacité et de stabilité, en plus d'une troisième mesure pour minimiser le nombre de jobs qui procèdent sur d'autres machines après la perturbation. Ce troisième critère a été optimisé à l'aide de la méthode d'optimisation lexicographique. Des tests numériques ont été élaborés pour étudier l'impact du coefficient d'efficacité-stabilité sur le système et explorer la limitation du modèle en termes de nombre de jobs.

Dans le quatrième chapitre, nous avons traité un problème de minimisation du temps d'attente moyen pondéré et de la déviation de la date de fin pondérée dans un problème de réordonnancement pour un atelier de type Flowshop avec des contraintes de blocages mixtes entre les machines. Une description détaillée du problème a été fournie. Puis, un modèle mathématique a été développé. Les résultats numériques ont ensuite été discutés. Ensuite, deux méthodes heuristiques basées sur la PLNE ont été proposées. Ces méthodes décomposent le problème de réordonnancement en sous-problèmes en divisant la partie concernée par le réordonnancement, et ensuite en décalant cette partie. Les heuristiques proposées ont été évaluées en faisant varier la fréquence d'apparition des perturbations, le nombre de jobs que contient la partie à réordonnancer, ainsi que le nombre de positions à décaler. Après l'analyse des résultats en termes de temps, et vu la complexité NP-difficile de ce type problème, quatre autres heuristiques ont été conçues. La première heuristique réagit en insérant le nouveau job

dans toutes les positions possibles. Les trois autres sont basées sur l'heuristique NEH. Enfin, une étude de sensibilité a été entamée pour comparer les heuristiques en termes de qualité de solution et de temps de calcul.

Finalement, une conclusion synthétisant l'apport de l'ensemble de nos travaux est présentée, ainsi que des perspectives proposant les axes d'amélioration.

Chapitre 1 : Etat de l'art et présentation du problème

Ce chapitre est dédié à la présentation et l'étude de l'état de l'art des travaux abordant les problèmes du réordonnancement. Dans un premier temps, une définition détaillée de ce concept est apportée, ainsi que ses différents champs d'application. Plusieurs termes liés à la thématique de réordonnancement sont ensuite présentés, à savoir les types de perturbation, les mesures de performance, les stratégies et les méthodes de réordonnancement. Une revue de la littérature des travaux liés au réordonnancement est établie, avec pour objectif le positionnement de notre problématique par rapport aux travaux existants. Dans un second temps, une description détaillée du problème est présentée, décrivant la nouvelle mesure de performance et son application dans les cas réels, ainsi qu'une explication de la stratégie prédictive-réactive adoptée pour ce problème.

1. Introduction

L'ère du numérique que nous vivons a changé complètement nos habitudes de consommation. En tant que clients, nous avons désormais la possibilité de créer, de modifier ou même d'annuler une commande à tout moment via des outils technologiques, notamment les applications mobiles, les sites web, etc. La progression constante de ces technologies de pointe change non seulement nos habitudes en tant que consommateurs, mais aussi la façon de traiter les commandes par les entreprises qui les reçoivent. En effet, ce changement de situation a un impact sur l'organisation de la production au sein des entreprises, plus précisément sur l'ordonnancement de la production. Un ordonnancement permet d'organiser valablement les systèmes de production des biens et des services, mais en présence d'évènements fortuits, ces hypothèses deviennent invalides. En plus, les pannes des machines ou le retard de l'arrivée de la matière première peuvent aussi engendrer des perturbations au planning déjà établi. Par conséquent, les planificateurs sont obligés de réagir très rapidement pour ne pas arrêter complètement la production. Face à ces environnements dynamiques, la réactivité de l'ordonnancement devient un enjeu majeur pour les décideurs. En effet, ils doivent non seulement générer un ordonnancement efficace, mais également très vite réagir face aux perturbations en le mettant à jour. Le réordonnancement est un processus qui permet de réviser un ordonnancement existant face aux aléas de commande ou de production. Depuis de nombreuses années, ce processus préoccupe de plus en plus les preneurs de décisions au sein des entreprises, mais il constitue aussi une thématique intéressante pour les chercheurs qui lui dédient une attention particulière.

Dans ce chapitre, nous nous intéressons à la présentation et l'étude de l'état de l'art des travaux abordant les problèmes de réordonnancement. Dans un premier temps, une définition détaillée de ce concept est apportée, ainsi que ses différents champs d'application. Plusieurs termes liés à la thématique de réordonnancement sont ensuite présentés, à savoir les types de perturbation, les mesures de performance, les stratégies et les méthodes de réordonnancement. Une revue de la littérature des travaux liés au réordonnancement a été établie, avec pour objectif le positionnement de notre problématique par rapport aux travaux existants. Ensuite, le problème est présenté, avec une description de la nouvelle mesure de performance et son application dans les cas réels, ainsi qu'une explication de la stratégie prédictive-réactive adoptée pour les problèmes qui seront étudiés.

2. Concept du réordonnancement

Dans les milieux industriels, l'ordonnancement permet d'organiser valablement la production des biens et des services. Cela engendre une meilleure coordination entre les services, pour augmenter la productivité et minimiser les coûts d'exploitation. Un planning de production permet une meilleure affectation des ressources, un contrôle des flux, et assure l'approvisionnement en matières premières nécessaires pour la fabrication. Un ordonnancement de production optimal ou quasi-optimal fournit alors aux personnels de l'atelier une planification efficace pour tenir la promesse de livraison. Dans la littérature, l'ordonnancement classique consiste à déterminer la séquence optimale suivant laquelle un ensemble de N tâches (jobs) doit être exécuté sur un ensemble de M ressources (machines) et soumis à des contraintes. Entre autres, il permet de déterminer les dates de début et de fin d'exécution des différentes

tâches sur chacune des ressources. Après la génération d'un ordonnancement initial, des évènements imprévus peuvent se produire et perturber ce calendrier déjà établi, tels que : l'arrivée, la modification ou l'annulation de commandes, les pannes de machines, ou le retard de la matière première, etc. Les preneurs de décision doivent dans ce cas très rapidement réagir pour modifier cet ordonnancement, en réponse à ces aléas, afin de ne pas arrêter complètement la production. C'est ainsi qu'apparaît le concept du réordonnancement.

2.1 Réordonnancement des systèmes : définitions et applications

2.1.1 Définition du processus de réordonnancement

Selon Vieira *et al*. (2003), le réordonnancement est le processus de mise à jour d'un ordonnancement de production existant en réponse à des perturbations. Ceci inclut :

- l'arrivée ou l'annulation de jobs
- le changement des dates de disponibilité des jobs
- la panne ou la réparation d'une machine
- le retard de la matière première
- les problèmes de qualité

Ces évènements imprévus sont connus dans la littérature sous le nom de perturbations ou d'incertitudes (Li et Ierapetritou, 2008). Contrairement à l'ordonnancement statique, dont les informations sont connues à l'avance, dans le cas du réordonnancement, les informations sont inconnues, le processus surveille l'exécution du planning, et réagit lorsqu'un évènement apparaît. La révision d'un ordonnancement est donc un processus réactif. Dans la littérature, le réordonnancement est aussi appelé ordonnancement réactif ou ordonnancement dynamique (Vieira *et al*., 2000b). Herrmann (2006) définit le réordonnancement comme étant un processus qui est effectué périodiquement pour planifier les tâches d'une période en fonction de l'état du système ou ponctuellement quand une perturbation arrive.

Dans la littérature, en fonction des données disponibles des tâches à exécuter, trois principaux types de problèmes d'ordonnancement ont été définis, à savoir, l'ordonnancement *offline, online,* et *semi-online*.

2.1.1.1 Ordonnancement *offline*

Lorsque toutes les informations concernant l'ensemble des tâches à exécuter sont connues à l'avance, les décideurs peuvent déterminer, à l'instant zéro, l'ensemble de l'ordonnancement, ce type de problème est appelé « ordonnancement *offline* ». Dans ce type de problème, les décideurs connaissent tout sur le futur.

2.1.1.2 Ordonnancement *online*

En pratique, les décideurs ne disposent pas, à l'avance, de toutes les informations concernant les tâches à exécuter. Ils sont donc obligés de créer un ordonnancement en se basant sur les informations de l'instant présent, puis le modifier au fur et à mesure, ce genre de problème est appelé « ordonnancement *online* ». Dans ce type de problème, les décideurs ne connaissaient rien sur le futur. Si dès qu'une tâche est disponible, elle doit être ordonnancée immédiatement avant l'arrivée d'une autre tâche, on parle d'un modèle *over-list*. Par contre, si la tâche qui vient d'être disponible peut attendre d'autres tâches avant son ordonnancement, on parle d'un modèle *over-time*.

2.1.1.3 Ordonnancement *semi-online*

Dans certains cas, une ou plusieurs informations sur les tâches à venir sont disponibles au préalable. Cette information peut être exploitée par les planificateurs pour une meilleure décision. Par exemple, les dates de disponibilité des tâches peuvent être connues sans connaitre leur durée. Dans ce type de problème, les décideurs connaissent des informations partielles sur le futur et on parle « d'ordonnancement *semi-online* ».

Les problèmes de réordonnancement sont souvent rencontrés dans les milieux industriels, notamment en production où les jobs représentent une tâche à exécuter. Mais ils apparaissent également dans d'autres secteurs d'activités, comme les blocs opératoires des systèmes hospitaliers dans lesquels les jobs représentent les patients à opérer et les salles d'opérations sont vues comme des ressources. Dans les deux paragraphes qui suivent, nous présenterons de manière plus détaillée des applications du réordonnancement dans ces deux secteurs.

2.1.2 Réordonnancement dans les systèmes de production industriels

L'industrie vit actuellement sa 4ème révolution, dite Industrie 4.0. Désormais, les entreprises sont de plus en plus réactives, elles ont la possibilité de récupérer en temps réel des données grâce à de nouveaux outils technologiques. Ces données peuvent correspondre, par exemple, à de nouvelles commandes ou à des commandes annulées ou modifiées, et sont exploitées en temps réel par l'outil de production pour garantir une meilleure productivité. Les entreprises ont donc intérêt à optimiser leurs processus de réordonnancement (Rossit *et al*., 2019). Plusieurs travaux ont abordé la problématique de réordonnancement, inspirés de cas de systèmes de production industriels. Uhlmann et Frazzon (2018) ont établi une revue de la littérature de ces travaux et leurs applications dans des cas réels. Par exemple, Guo *et al*. (2016) ont traité un problème de réordonnancement inspiré du cas de la fabrication du verre de Quartz. Les auteurs ont considéré le temps d'attente maximale des produits comme critère pour mesurer l'efficacité de l'ordonnancement. En réalité, cette mesure représente la durée d'attente maximale de certains matériaux devant un four de réchauffage. Dans leur cas, la minimisation de ce critère permet d'économiser la consommation d'énergie. Rahmani et Ramezanian (2016) ont étudié un problème de réordonnancement inspiré du cas de la production des vannes en acier pour les industries du pétrole. L'atelier de fabrication a été modélisé comme un atelier de type flowshop hybride. Les auteurs ont considéré le retard moyen des jobs comme une mesure de l'efficacité de l'ordonnancement, ainsi que l'écart absolu des dates de fin des jobs comme mesure de stabilité. Ils ont également introduit la notion de résistance aux changements des opérateurs, qui impacte la durée des jobs après chaque changement de planning. Guo et Tang (2019) ont traité un problème de réordonnancement d'un cas réel rencontré dans l'industrie sidérurgique. Dans leur situation, chaque commande doit être planifiée en tenant compte des contraintes de production, puis affectée à l'unité de production appropriée. Des commandes urgentes arrivent régulièrement, constituant une perturbation du système de production.

2.1.3 Réordonnancement dans les blocs opératoires

Le bloc opératoire constitue le secteur le plus couteux et le plus important dans la majorité des hôpitaux (Bouguerra *et al*. 2015). Selon Macario *et al*. (1995), il représente plus de 10% du budget de l'hôpital. La gestion d'un bloc opératoire est une tâche complexe car il s'agit de planifier et ordonnancer les opérations dans les salles opératoires et de satisfaire les besoins des patients. La planification des opérations se fait en deux étapes. La première est la planification hebdomadaire, pour déterminer quel jour de la semaine les opérations auront lieu, en tenant

compte des disponibilités des salles et des chirurgiens. La deuxième est l'ordonnancement journalier, pour déterminer la séquence des opérations dans chaque salle (Lust *et al*., 2012). En réalité, il existe deux types d'opérations. Les opérations électives sont connues à l'avance, alors que les opérations urgentes peuvent arriver à tout moment. Ces dernières constituent une perturbation de l'ordonnancement déjà établi. Dans ce cas, un réordonnancement des opérations doit être mis en œuvre. Dans la littérature, plusieurs travaux se sont intéressés à cette problématique. Par exemple, Addis *et al*. (2016) ont considéré un problème d'ordonnancement dans lequel l'objectif est de sélectionner un ensemble de patients électifs parmi une liste d'attente donnée et les affecter à un ensemble de blocs de salles opératoires disponibles. De nouveaux patients entrent en continu sur la liste d'attente. Un réordonnancement des patients est alors effectué, ayant comme objectif de minimiser le temps d'attente des patients et le retard par rapport aux dates de fin prévue des opérations. Ballestín *et al*. (2019) ont traité le problème de construction d'un ordonnancement des patients en chirurgie élective dans une unité hospitalière. Deux phases sont effectuées. Dans la première phase, un ordonnancement provisoire est établi deux semaines avant, ayant comme objectif de minimiser le pourcentage de retard des patients. Dans la deuxième phase, un dernier ordonnancement est réalisé en tenant compte de l'évolution des informations disponibles. Une méthodologie générale est proposée pour calculer ce dernier calendrier. Erdem *et al*. (2012) ont développé un modèle basé sur la PLNE pour le réordonnancement des patients électifs à l'arrivée des patients urgents en considérant deux types d'unités cliniques, à savoir les salles d'opérations et les unités de soins post-anesthésiques. Le modèle développé consiste à minimiser le coût des heures supplémentaires des salles d'opération et des unités de soins intensifs, le coût de report ou de la replanification des opérations électives, et le coût du refus des patients urgents. Bouguerra *et al*. (2016) ont proposé une stratégie d'affectation en ligne pour réordonnancer les opérations existantes en réponse à l'arrivée des opérations urgentes. Leur stratégie consiste à minimiser trois objectifs à savoir, le temps d'attente des patients, la déviation des dates de début des jobs par rapport à l'ordonnancement original, et les coûts des heures supplémentaires. Les auteurs ont développé un modèle basé sur la PLNE pour résoudre ce problème.

Les problèmes de réordonnancement sont aussi rencontrés dans d'autres secteurs d'activités, notamment les systèmes de circulation ferroviaire (Altazin *et al*., 2017, Josyula *et al*., 2018), les systèmes cloud computing (Lee et Zomaya 2010, Yao *et al*., 2016), ou les véhicules de transport (Li *et al*., 2009, Spliet *et al*., 2014).

2.2 Types de perturbations dans l'ordonnancement des systèmes de production

Les perturbations sont des évènements inattendus qui peuvent modifier l'état du système et affecter ses performances. À l'arrivée de l'un de ces évènements, un réordonnancement se déclenche. Pour cette raison, les perturbations sont aussi appelées les facteurs de réordonnancement (Dhingra *et al*., 1992). Plusieurs types de perturbations ont été étudiés dans la littérature. Dans les paragraphes qui suivent, nous présentons les plus fréquemment considérées.

2.2.1 Perturbations dues à l'arrivée ou l'annulation de jobs

La progression constante du e-commerce a changé le rapport entre l'entreprise et les consommateurs. Désormais, un client peut à tout moment, créer, modifier, ou annuler une commande. Ce changement de situation a un impact sur l'organisation au sein de l'unité de production. En effet, l'entreprise doit mettre à jour le planning de production pour satisfaire les

besoins des clients et assurer une meilleure productivité, d'où la nécessité du réordonnancement. Dans la littérature, plusieurs travaux se sont intéressés aux problèmes de réordonnancement en réponse à des perturbations dues à l'arrivée ou l'annulation de jobs. Par exemple, Gao *et al*. (2015) ont étudié un problème d'ordonnancement de type job shop flexible avec l'insertion de nouvelles commandes. Ce problème est inspiré du cas de la logistique en boucle fermée, dont les produits retournés devront être remis à neuf et traités sur les mêmes machines. Ces produits peuvent être insérés dans l'ordonnancement en cours d'exécution, ils seront alors combinés avec les produits non encore exécutés pour être réordonnancés. Zhang et Wong (2017) ont considéré un problème d'ordonnancement de type flowshop hybride considérant trois types de perturbations, l'arrivée de nouvelles commandes, l'annulation de commandes et les pannes de machines. Pour faire face à cela, les auteurs ont développé deux modèles de réordonnancement, l'un est partiel et consiste à ne réordonnancer que les jobs concernés sans changer l'affectation des anciens jobs aux machines, et l'autre est total et consiste à réordonnancer tous les jobs qui n'ont pas encore commencé leur exécution.

2.2.2 Perturbations dues aux pannes des machines

Dans de nombreux processus industriels, les pannes des machines affectent en permanence les activités planifiées. La maintenance préventive permet de réduire le taux de panne. Cependant, il est presque impossible d'éradiquer complètement l'apparition, par définition inattendue, d'une panne. De même, d'autres paramètres tels que l'indisponibilité des pièces de rechange et/ou des techniciens de maintenance sont très susceptibles d'arriver. Par conséquent, il est essentiel de réagir très rapidement afin de produire un nouvel ordonnancement en réponse à ces perturbations. Katragjini *et al*. (2013) ont considéré un problème de réordonnancement de type flowshop subissant trois types de perturbations, les pannes des machines, l'arrivée de nouveaux jobs, et le changement des dates de disponibilité des jobs. Les auteurs ont proposé une nouvelle méthodologie pour gérer simultanément ces trois types de perturbations. Molaee *et al*. (2021) ont étudié un problème de réordonnancement sur une machine unique qui subit des pannes aléatoires. Les auteurs ont considéré que les pannes de la machine peuvent arriver tout au long de l'horizon de simulation, avec une date et une durée aléatoire. Leur but est de réordonnancer le planning de production, avec pour objectif la minimisation du retard maximal, tout en tenant compte du temps de préparation des commandes « *setup time* ».

2.2.3 Perturbations dues au manque de matière première

Le retard dans l'approvisionnement en matière première a des conséquences sur la planification de la production, à savoir la modification des dates de disponibilité des jobs « *Release dates* ». En effet, l'ordonnancement initial établi tient compte de la date de disponibilité du job. Si la matière première nécessaire à la fabrication de celui-ci est indisponible, la date de début de la production planifiée est impossible à respecter. La planification du job concerné doit alors être reportée à une date ultérieure. Ce type de perturbation est connue dans la littérature sous le nom de « *Jobs Release Variation* », c'est-à-dire la variation des dates de disponibilité des jobs. Plusieurs travaux se sont intéressés à ce type de perturbation. Par exemple, Peng *et al*. (2019) ont considéré trois types de perturbations dans un problème d'atelier de type flowshop hybride, à savoir les pannes des machines, l'arrivée de nouveaux jobs et la variation des dates de disponibilité des jobs. Les auteurs ont adapté une métaheuristique basée sur la recherche à voisinage variable pour résoudre ce problème, en tenant compte de la possibilité pour ces trois perturbations d'arriver simultanément, et avec

comme objectif la minimisation du Makespan et de l'instabilité du système. Li *et al*. (2015) ont considéré cinq types de perturbations dans un problème de réordonnancement d'un atelier de type flowshop : l'arrivée et l'annulation de jobs, les pannes des machines, le changement des durées et des dates de disponibilité des jobs. Ces perturbations sont gérées simultanément, avec l'objectif de minimiser le Makespan et l'instabilité du système. Les auteurs ont développé un algorithme d'optimisation basé sur l'apprentissage TLBO (Teaching-Learning Based Optimization) pour résoudre ce problème.

2.3 Mesures de performance dans les problèmes de réordonnancement

Pour mesurer la performance d'un ordonnancement, les mesures classiques d'efficacité sont souvent utilisées, comme le Makespan, la somme des dates de fin ou le retard moyen. Dans les environnements dynamiques, d'autres mesures peuvent être considérées. Par exemple, les mesures de stabilité ou de robustesse de l'ordonnancement qui permettent de limiter l'écart entre l'ordonnancement déjà établi et le nouvel ordonnancement. Dans les paragraphes qui suivent, nous présentons les différentes mesures de performance fréquemment utilisées dans les problèmes de réordonnancement.

2.3.1 Efficacité de l'ordonnancement

Les mesures d'efficacité sont souvent utilisées pour mesurer la performance d'un ordonnancement. Ces mesures consistent généralement à minimiser un critère qui dépend du temps, notamment le Makespan *i.e*. la date de fin maximale de tous les jobs (Zakaria et Petrovic, 2012), la somme des dates de fin ou des dates de fin pondérées par les poids des jobs (Liu *et al*., 2013 ; Hall et Potts, 2010), le retard moyen (Dou *et al*., 2020), le temps de parcours moyen (Ozlen et Azizoğlu, 2011), ou la latence maximale (Hall and Potts, 2004). Angel-Bello *et al*. se sont intéressés en 2021 à la minimisation du Makespan dans un problème de réordonnancement sur une machine unique, avec comme perturbation l'arrivée de nouveaux jobs. Les auteurs ont implémenté deux nouvelles stratégies de réordonnancement pour résoudre leur problème. La première consiste à réordonnancer tous les jobs disponibles au début de l'intervalle en minimisant le Makespan. La deuxième consiste à ne réordonnancer que les jobs pouvant être terminés avant la prochaine période de réordonnancement. Gao *et al*. ont traité en 2015 un problème de réordonnancement de type job shop en considérant une fonction multi-objectif qui évalue l'efficacité de l'ordonnancement. Celle-ci mesure simultanément le Makespan, le retard moyen, la précocité moyenne, la charge de travail maximale des opérateurs et la charge de travail moyenne des opérateurs. Les auteurs ont développé quatre heuristiques pour résoudre leur problème.

2.3.2 Stabilité de l'ordonnancement

Dans les environnements dynamiques, la stratégie de réordonnancement implique une mise à jour de l'ordonnancement déjà établi pour adapter celui-ci à la perturbation apparue. Ainsi, le nouvel ordonnancement est donc vraisemblablement différent du précèdent. Cela signifie que la plupart, sinon la totalité, des jobs planifiés précédemment et qui n'ont pas encore commencé leur exécution sur l'outil de production, peuvent avoir une date de début différente de celle qui était planifiée avant l'apparition de la perturbation. Dans la pratique, cette situation est gênante, en particulier si le planning en cours d'exécution nécessite une préparation des jobs à l'avance. Des coûts supplémentaires de changement de planning doivent dans ce cas être appliqués, qui incluent les coûts des réapprovisionnements en matière première, ou les coûts de réaffectation

des ressources. Par conséquent, des mesures de stabilité de l'ordonnancement sont prises en compte lorsqu'il s'agit de réordonnancer les jobs.

Dans la littérature, nombreux sont les travaux qui se sont intéressés à ce type de mesure. Par exemple, Akkan (2015) a étudié un problème de réordonnancement dans lequel la fonction objectif mesure simultanément l'efficacité et la stabilité de l'ordonnancement. L'auteur a considéré le retard maximal comme mesure d'efficacité et la somme des déviations absolues des dates de début des jobs comme mesure de stabilité. Cette mesure de stabilité détermine la différence absolue entre la date de début d'un job avant et après le réordonnancement. Il a aussi développé un algorithme qui permet d'améliorer la stabilité de l'ordonnancement, sans en impacter l'efficacité. Liu et Zhou (2013) se sont intéressés à un problème de réordonnancement sur des machines parallèles et identiques. Les auteurs ont considéré le nombre de jobs qui changent de machine après chaque réordonnancement comme mesure de stabilité, ainsi que la somme des dates de fin des jobs comme mesure d'efficacité. D'autres mesures de stabilité sont présentées dans le Tableau 1.

Tableau 1. Mesures de stabilité d'un ordonnancement

Référence	Mesure de stabilité	Description
Rangsaritratsamee *et al*. (2004)	Pénalité de la déviation totale	Mesure la déviation des dates de début des jobs par rapport à l'existant.
Yang (2007)	Coûts du réordonnancement	Mesure les coûts administratifs et les coûts des perturbations associés à la coordination avec les fournisseurs et les clients.
Yuan *et al.* (2007)	Déviation maximale absolue	Mesure la déviation maximale absolue entre la position d'un job avant et après le réordonnancement (en unités de temps).
Hoogeveen *et al*. (2012)	Déviation absolue des positions	Mesure pour chaque job la différence absolue entre sa position avant et après le réordonnancement (en classement).
Liu et Zhou (2013)	Nombre d'opérations qui changent de machine	Mesure le nombre d'opérations qui changent de machine, après le réordonnancement.
Liu et Ro (2014)	Changement de la date d'échéance	Mesure la pénalité due aux changements des dates communiquées aux clients à chaque fois.
Akkan (2015)	Déviation absolue des dates de début	Détermine la différence absolue entre la date de début d'un job avant et après le réordonnancement
Peng *et al*. (2018)	Nombre d'opérations déviées	Mesure le nombre d'opérations dont la date de début a changé.
Wang, D *et al*. (2020)	Déviation du Makespan	Mesure la différence entre la valeur du Makespan de l'ordonnancement avant et après le réordonnancement.

2.3.3 Robustesse de l'ordonnancement

Plusieurs définitions ont été fournies pour définir un ordonnancement robuste.

Selon Jorge Leon *et al*. (1994), « *Un ordonnancement robuste est un ordonnancement insensible aux perturbations imprévues* ».

Selon Jensen (2003), « *Un ordonnancement est dit robuste si sa performance reste inchangée malgré les perturbations qu'il subit* ».

Selon Canon et Jeannot (2009), « *Un ordonnancement est dit robuste s'il est capable d'absorber un niveau d'incertitude au niveau des durées des tâches, tout en maintenant une solution stable* ».

Une autre définition a été récemment fournie par Salido *et al*. (2017) : « *Un ordonnancement est robuste s'il est capable d'absorber la perturbation sans affecter les jobs existants* ».

Comme les mesures de stabilité, les mesures de robustesse calculent l'écart de performance de l'ordonnancement après la perturbation par rapport à l'ordonnancement établi, en d'autres termes la proximité entre les deux ordonnancements, l'ordonnancement modifié et établi. Cependant, l'approche utilisée pour obtenir un ordonnancement robuste est différente. Un

ordonnancement robuste est obtenu par une approche proactive, c'est-à-dire la création de plusieurs scénarios possibles grâce à des estimations afin d'obtenir, dès le premier coup, un ordonnancement initial robuste qui ne va pas changer malgré les perturbations qu'il va subir. Rahmani et Heydari (2014) ont développé pour un problème de réordonnancement de type flowshop une approche d'optimisation dite robuste. Celle-ci consiste à obtenir un ensemble de solutions pour un problème d'ordonnancement afin qu'il reste robuste malgré les changements de durées des jobs qui peuvent se produire, cela en prévoyant des durées de jobs et en créant plusieurs scénarios possibles. Quand un changement de durée de job se produit, l'ordonnancement va l'absorber car il laisse des dates inactives dans le calendrier. Pour mesurer la robustesse, les auteurs ont calculé la somme des différences absolues entre les dates de fin des jobs dans les ordonnancements, l'ordonnancement établi et modifié. Chaari *et al*. (2011) ont proposé pour un problème de réordonnancement de type flowshop hybride, une fonction multi objectif pour mesurer la robustesse de l'ordonnancement. Celle-ci minimise à la fois le Makespan et la déviation entre le Makespan de la solution établie et de l'ordonnancement modifié. Les auteurs ont développé un algorithme génétique pour résoudre leur problème.

3. Méthodes, stratégies, et modèles de réordonnancement

Dans ce paragraphe, nous présentons les différentes méthodes et stratégies de réordonnancement, ainsi que des travaux de réordonnancement dans différents environnements machines.

3.1 Méthodes de réordonnancement

Selon Vieira *et al*. (2003), il existe deux principales méthodes de réordonnancement. La première est la génération d'un ordonnancement robuste. La deuxième est la réparation d'un ordonnancement existant. Dans les deux paragraphes qui suivent, nous discutons plus en détails ces deux méthodes.

3.1.1 Génération d'un ordonnancement robuste

Dans les environnements dynamiques, l'ordonnancement en cours d'exécution va probablement subir des perturbations. Si ces perturbations n'impactent pas sa performance, l'ordonnancement est dit robuste. La robustesse d'un ordonnancement est alors sa capacité à absorber des perturbations sans nuire à sa performance (Jensen, 2003). La méthode de génération d'un ordonnancement robuste consiste à prévoir les incertitudes qu'il peut subir et les prendre en considération pour créer un ordonnancement initial dit robuste (Wu *et al*., 1999). Les étapes de la méthode robuste, selon Daniels et Kouvelis (1995), sont les suivantes :

- Etape 1 : Identifier les scénarios possibles de perturbation.
- Etape 2 : Définir une mesure de robustesse.
- Etape 3 : Incorporer les scénarios et la mesure choisie pour créer un ordonnancement robuste.
- Etape 4 : Exécuter l'ordonnancement sans changement, malgré ce qui peut se produire à l'avenir.

L'avantage de cette méthode est qu'elle permet de fournir un ordonnancement stable, puisque les incertitudes sont prises en compte dans la génération de l'ordonnancement. Son inconvénient est qu'elle nécessite une connaissance approfondie des incertitudes, c'est-à-dire d'avoir toutes les informations nécessaires pour traiter tous les scénarios possibles. Dans

certains cas, il est difficile de générer tous les scénarios, et la solution peut nécessiter des réparations.

3.1.2 Réparation d'un ordonnancement existant

En raison des perturbations imprévues que subit l'ordonnancement initial, il est nécessaire de le mettre à jour pour assurer la productivité. Cette action de mise à jour d'un ordonnancement en cours d'exécution est appelée « *Réparation* ». Selon Vieira *et al*. (2003), il existe trois méthodes courantes de réparation d'un ordonnancement : la régénération, le réordonnancement partiel, et le décalage vers la droite.

La régénération (Church et Uzsoy, 1992 ; Wu *et al*., 1993) construit un nouvel ordonnancement en réordonnançant tous les jobs, sans prendre en considération l'ordonnancement initial. Elle est aussi appelée réordonnancement total. Elle nécessite un temps d'exécution élevé car tous les jobs doivent être réordonnancés à chaque arrivée d'une perturbation. Cette méthode produit généralement des solutions efficaces, mais instables par rapport à l'ordonnancement initial.

Le réordonnancement partiel (Li *et al*., 1993 ; Wu et Li, 1995) tient compte de l'ordonnancement initial, et construit un ordonnancement qui ne dévie pas trop les jobs existants avant la perturbation. Cela réduit l'efficacité, mais augmente la stabilité de l'ordonnancement.

Le décalage vers la droite (Abumaizar et Svestka, 1997) consiste à décaler, après la date de la perturbation, toutes les dates de début des jobs non exécutés vers la droite dans un diagramme de Gantt. Cette méthode est souvent utilisée lors des perturbations dues à des pannes de machines. Elle fournit la plus stable des solutions dans ce cas. Mais, quand il s'agit de l'arrivée ou de l'annulation de jobs, cette méthode n'est pas souvent utilisée car elle fournit de mauvaises solutions en termes d'efficacité.

Le Tableau 2 récapitule les méthodes discutées, ainsi que les avantages et inconvénients de chacune.

Tableau 2. Les méthodes de réordonnancement

Méthode de réordonnancement		**Avantage**	**Inconvénient**
Génération d'un ordonnancement robuste		Fournit un ordonnancement stable	Nécessite une connaissance approfondie des incertitudes
Réparation d'un ordonnancement existant	Régénération	Efficace	Instable et nécessite un temps d'exécution élevé
	Réordonnancement partiel	Plus stable par rapport à la régénération	Efficace, mais moins que la régénération
	Décalage vers la droite	Fournit la plus stable des solutions	Pas adaptée à tous les types de perturbations

3.2 Stratégies de réordonnancement

Après avoir défini une méthode de réordonnancement, une stratégie doit aussi être définie. La stratégie de réordonnancement répond à la question : *Comment l'ordonnancement réagit-il face à l'apparition d'un évènement ?* Cette section présente les trois stratégies les plus couramment utilisées pour résoudre ces problèmes, à savoir : l'ordonnancement complètement réactif, l'ordonnancement robuste proactif et l'ordonnancement prédictif-réactif. Ces stratégies ont été répertoriées, en particulier, dans les travaux de Herroelen et Leus (2004), Aytug *et al* (2005), et Ouelhadj et Petrovic (2009).

3.2.1 Ordonnancement complètement réactif

L'ordonnancement complètement réactif, aussi appelé ordonnancement en temps réel, ne nécessite pas de créer un ordonnancement de production initial. Les choix du job à exécuter et des ressources à utiliser, sont des décisions prises en temps réel. Parmi une liste de jobs à exécuter, un job est choisi en fonction de sa priorité grâce à un critère prédéfini. En fait, le système crée des règles de répartition ou des règles de priorités « *Dispatching rules* » pour trier les jobs. Quand la ressource est disponible, elle utilise ces règles pour choisir le job à exécuter (Haupt, 1989 ; Bhaskaran et Pinedo, 1991). Une extension de cette stratégie a été développée par Wu et Wysk (1989), permettant de sélectionner d'une façon dynamique, une règle de répartition parmi plusieurs, en fonction des conditions actuelles du système. L'avantage de cette stratégie est son temps d'exécution extrêmement faible, en plus des règles de priorité qui sont faciles à expliquer aux utilisateurs du système de planification. Certaines règles de répartition classiques sont listées dans Nie *et al*. (2013). Ce sont des extensions des règles d'ordonnancement statique, comme *SPT* (le délai de traitement le plus court), *EDD* (la date d'échéance la plus proche), *FIFO* (premier arrivé, premier servi), etc. L'inconvénient de ces méthodes est qu'elles ne tiennent pas compte de la stabilité de l'ordonnancement et qu'elles ne sont pas adaptées à tous les types de problèmes. Cependant, d'autres règles de réparation peuvent être adaptées à un problème spécifique, mais celles-ci nécessitent une grande quantité d'informations, et les priorités doivent être calculées à chaque apparition d'une nouvelle perturbation.

3.2.2 Ordonnancement robuste proactif

Cette stratégie consiste à étudier les perturbations possibles que peut subir l'ordonnancement et les intégrer pour créer un ordonnancement robuste. Cette prise en compte des perturbations prévisibles permet de créer un ordonnancement robuste, qui peut absorber les changements qui apparaissent dans le futur. C'est pour cette raison que la stratégie est dite robuste proactive. La génération d'un ordonnancement robuste se fait selon les étapes listées dans le paragraphe 3.1.1. Elle s'appuie sur l'évaluation de plusieurs ordonnancements dits ordonnancements candidats, afin de sélectionner celui qui va pouvoir fournir la solution la plus robuste. Cette évaluation est généralement basée sur des mesures qui quantifient la capacité d'un ordonnancement à rester efficace dans diverses circonstances. Pour créer plusieurs scénarios possibles, les auteurs admettent des hypothèses (Ghezail *et al*., 2010) qui sont caractérisées par :

- Une cause : outils non disponible, probabilité d'une panne.
- Un contexte : saison des ventes, situation environnementale.
- Un impact : retard prévu.
- Une inclusion : façon de gérer la perturbation.

Comme listé ci-dessus, la création d'un ordonnancement robuste proactif nécessite une connaissance approfondie des incertitudes. En revanche, si celles-ci sont maîtrisées, cette stratégie fournit un ordonnancement stable, qui est capable d'absorber les perturbations sans nuire à sa performance.

3.2.3 Ordonnancement prédictif-réactif

C'est la stratégie la plus utilisée pour résoudre les problèmes de réordonnancement (Ouelhadj et Petrovic, 2009). Le principe de la stratégie prédictive-réactive repose sur deux étapes. La première étape, la phase prédictive, consiste à créer un ordonnancement initial. La seconde étape, la phase réactive, consiste à réviser à chaque fois l'ordonnancement en réponse à une perturbation afin de minimiser son impact sur les performances du système (Vieira *et al*., 2003). Lorsque l'ordonnancement est en cours d'exécution, le processus surveille l'apparition d'une perturbation. Si celle-ci apparaît, il met à jour le planning. C'est à dire qu'à chaque apparition d'une perturbation (itération), le processus réagit en réparant l'ordonnancement. La stratégie prédictive-réactive est donc un processus itératif. Wu et Li (1995) ont décrit le processus de cette stratégie prédictive-réactive en trois étapes :

- L'étape d'évaluation : consiste à évaluer l'impact causé par la perturbation.
- L'étape de résolution : consiste à déterminer la meilleure solution de réordonnancement.
- L'étape de révision : consiste à appliquer la solution choisie et donc à réviser l'ordonnancement existant.

Plusieurs travaux ont appliqué cette stratégie pour résoudre les problèmes de réordonnancement. Par exemple, Yang et Geunes (2008) ont implémenté une stratégie d'ordonnancement prédictive-réactive pour un problème de réordonnancement sur une machine unique qui subit des perturbations dues à l'arrivée de nouveaux jobs. Leur stratégie consiste à créer, dans la phase prédictive, un ordonnancement initial avec des temps d'inactivité sur la machine. Après qu'un nouveau job arrive, dans la phase réactive, la stratégie réagit en incluant les nouveaux jobs dans les dates d'inactivité. Wang *et al*. (2015) ont utilisé la stratégie prédictive-réactive pour un problème de réordonnancement des opérations chirurgicales dans une salle d'opération. Les perturbations sont dues à l'arrivée des opérations urgentes. Les auteurs génèrent un ordonnancement prédictif avec certaines plages horaires libres. Puis, dans la phase réactive, les opérations urgentes sont insérées, après leur apparition, dans les plages horaires dédiées.

Par ailleurs, des politiques de réordonnancement sont nécessaires pour implémenter la stratégie prédictive-réactive (Vieira *et al*., 2003). La politique de réordonnancement répond à la question : *Quand est ce que l'ordonnancement réagit* ? Trois types de politiques sont connues dans la littérature, à savoir la politique périodique, la politique évènementielle « *event-driven* », et la politique hybride.

3.2.3.1 Politique de réordonnancement périodique

Dans cette politique, l'horizon temporel est divisé en périodes. Le processus de réordonnancement est lancé à chaque période, quelle que soit la situation (*i.e.* apparition d'une perturbation ou pas), et quel que soit le nombre des perturbations (Sabuncuoglu et Karabuk, 1999). Dans de nombreuses situations industrielles, le réordonnancement se fait d'une manière périodique, en particulier dans les environnements où il n'y a pas d'acquisition de données en

ligne depuis l'atelier pour surveiller l'état de l'usine en temps réel. Dans ce cas, le responsable du planning rassemble toutes les informations pour élaborer des mises à jour de l'ordonnancement à des intervalles définis. Cette politique fournit plus de stabilité, mais face à d'importantes perturbations, elle peut fournir des résultats moins efficaces. De plus, la détermination de la période optimale de réordonnancement est un sujet difficile.

3.2.3.2 Politique de réordonnancement évènementielle

Dans la politique évènementielle, le processus de réordonnancement est déclenché à chaque arrivée d'un évènement perturbateur en temps réel (Church et Uzsoy, 1992). À chaque fois qu'une perturbation arrive, l'ordonnancement initial est révisé. Par conséquent, le temps de calcul peut devenir excessif, car la réparation de l'ordonnancement se produit tant qu'il y a des perturbations. Dans les milieux industriels, cette politique nécessite un système de capture des données très rapide et fiable pour collecter rapidement les nouveaux évènements, en particulier dans les grandes installations industrielles. Cependant, cette politique reste la plus utilisée par les chercheurs scientifiques, car elle fournit des résultats performants tant en termes d'efficacité que de stabilité de l'ordonnancement, grâce à des fonctions objectif qui optimisent les deux aspects simultanément.

3.2.3.3 Politique de réordonnancement hybride

La politique hybride réordonnance la séquence périodiquement, mais également lorsque des évènements spéciaux (majeurs) apparaissent (Chacon, 1998). Les évènements majeurs sont généralement les pannes des machines, l'arrivée de tâches urgentes, l'annulation d'une tâche, ou le changement des priorités des tâches. L'avantage de cette politique est qu'elle permet de choisir, parmi les deux politiques décrites auparavant, la meilleure selon la situation. L'enjeu est de développer la méthodologie optimale, qui combinera les deux politiques en un seul système. Par exemple, Pfeiffer *et al*. (2008) ont considéré une politique de réordonnancement hybride pour la planification de la production sur une machine unique. Leur méthodologie consiste à définir un intervalle dans lequel le réordonnancement s'établit périodiquement. Puis, lors de l'apparition d'une perturbation, cet intervalle est modifié en fonction de la date d'apparition de la perturbation.

Le Tableau 3 récapitule les avantages et les inconvénients des stratégies étudiées.

Tableau 3. Les stratégies de réordonnancement

Stratégie de réordonnancement	Politique de réordonnancement	Avantage	Inconvénient
Réactive		Faible temps d'exécution	Fournit un ordonnance-ment qui n'est pas stable. Les règles de répartition spécifiques nécessitent une grande quantité d'informations.
Robuste proactive		Fournit un ordonnancement stable	Nécessite une connaissance approfondie des incertitudes.
Prédictive-réactive	Périodique	Fournit plus de stabilité	Moins efficace face à des perturbations importantes.
	Evènementielle	La plus performante	Temps de calcul élevé. Nécessite des capteurs puissants pour être appliquée en pratique.
	Hybride	Permet de choisir la politique optimale selon la situation	La méthodologie de combinaison des deux politiques reste un enjeu.

3.3 Réordonnancement dans des environnements de machines différents

Les problèmes de réordonnancement ont été traités dans tous les types d'environnements machines. En premier lieu, sur une machine unique représentant un seul poste de travail. Le problème de réordonnancement sur une machine unique est générique, et peut être utilisé comme une base pour d'autres problèmes plus compliqués, dans des environnements contenant des machines multiples. En deuxième lieu, sur un environnement de machines parallèles, représentant des postes de travail identiques. Ce type de problème est très souvent rencontré dans les applications réelles. Par exemple, dans l'ordonnancement des blocs opératoires, où les opérations électives et urgentes doivent être planifiées dans des salles d'opérations identiques. Enfin, pour des machines en série où l'ensemble des jobs doivent passer sur un ensemble de machines selon le type d'atelier, c'est le cas des lignes de production dans les usines. Dans ce qui suit, nous présentons des travaux de réordonnancement dans ces différents types d'environnements de machines.

3.3.1 Réordonnancement sur une machine unique

Plusieurs auteurs se sont intéressés aux problèmes de réordonnancement sur une machine unique. On peut citer, entre autres, les travaux suivants.

Hall *et al*. (2007) ont considéré, sur une machine unique, un problème de réponse à des perturbations dues à l'arrivée de nouveaux jobs. L'objectif est de minimiser le retard maximal des jobs, sous réserve d'une limite sur le changement de date de début des jobs d'origine. Les auteurs ont développé un algorithme Branch and Bound pour résoudre ce problème.

Luo *et al*. (2018) se sont intéressés au réordonnancement des jobs sur une machine unique qui subit des temps d'inactivité inattendus. L'objectif est de minimiser simultanément la moyenne des dates de fin des jobs pondérées par les poids des jobs et la déviation maximale des dates de fin des jobs. Cette dernière mesure la différence entre la date de fin d'un job avant et après la perturbation. Les auteurs ont développé un algorithme en temps pseudo-polynomial et un schéma d'approximation en temps polynomial pour résoudre ce problème.

Cheng *et al*. (2018) ont traité un problème sur une machine unique, avec l'arrivée de nouveaux jobs comme perturbation. L'objectif est de minimiser le Makespan et la somme des dates de fin des jobs, sous un nombre limité de changements par rapport à l'ordonnancement original, sinon des coûts de perturbations sont appliqués. Les auteurs ont aussi introduit la notion de l'effet d'apprentissage, c'est-à-dire que les opérateurs acquièrent de l'expérience au cours des exécutions des tâches et cela réduit le temps d'exécution des tâches en fonction du temps. Les auteurs ont développé un algorithme en temps pseudo-polynomial et un algorithme en temps polynomial pour résoudre ce problème.

Le Tableau 4 présente une bibliographie des travaux de réordonnancement sur une machine unique.

Tableau 4. Bibliographie des travaux de réordonnancement sur une machine unique

Référence	Perturbations	Mesure de performance			Stratégie et/ou méthode	Contrainte de blocage	Méthode de résolution
		Efficacité	Stabilité	Robustesse			
Vieira *et al*. (2000a)	NJ	Temps d'écoulement moyen, Utilisation de la machine, fréquence de réordonnancement	N/A	N/A	Évènementielle, Périodique	N/A	Modèle analytique et simulation
Yang (2007)	NJ	Somme des dates de fin, retard pondéré	Déviation absolue des dates de fin, coûts administratifs	N/A	Réordonnancement partiel	N/A	Algorithme polynomial basé sur SPT, RLGV
Hoogeveen *et al*. (2012)	NJ	Somme des temps setup, Makespan	Changement des positions et des dates de fin des jobs	N/A	Réordonnancement partiel	N/A	Algorithme polynomial optimal
Akkan (2015)	NJ	Retard maximal	Somme des déviations absolues des dates de début des jobs	Somme des déviations absolues des dates de début des jobs	Prédictive-Réactive, Évènementielle avec des temps inactifs	N/A	Branch and bound, recherche locale
Guo *et al*. (2016)	NJ	Temps d'attente maximal	N/A	N/A	Réordonnancement partiel	N/A	Heuristique adaptée
Cheng *et al*. (2018)	NJ	Makespan, somme des dates de fin des jobs	Déviation absolue des dates de fin et des positions	N/A	Réordonnancement partiel	N/A	Algorithmes polynomial et pseudo polynomial
Da Silva *et al*. (2019)	NJ, JA	Makespan	N/A	N/A	Évènementielle	N/A	PLNE
Luo *et al*. (2020)	JA	Somme pondérée des dates de fin des jobs, retard maximal, coût d'annulation des jobs	N/A	N/A	Complétement réactive	N/A	Programmation dynamique, schéma d'approximation
Sahraeian *et al*. (2020)	DI, PM	Retard moyen	Déviation des dates de fin	Différence entre les sommes des dates de fin de jobs	Prédictive-Réactive	N/A	Heuristique prédictive
Zhao et Wang (2020)	NJ	Latence maximale	N/A	N/A	Complétement réactive	N/A	Algorithme évolutionnaire dynamique
Angel-Bello *et al*., (2021)	NJ	Makespan	N/A	N/A	Périodique	N/A	Métaheuristique basée sur l'algorithme glouton
Jun et Lee (2021)	NJ	Retard moyen pondéré	N/A	N/A	Complétement réactive	N/A	Méthode basée sur l'algorithme génétique

NJ : Nouveaux jobs, JA : Jobs annulés, PM : Pannes machines, DI : Durées des jobs incertaines, RLGV : Recherche locale à grand voisinage, N/A : Non applicable.

3.3.2 Réordonnancement sur des machines parallèles

D'autres travaux dans la littérature se sont intéressés aux problèmes de réordonnancement sur des machines parallèles. Nous présentons quelques-uns de ces travaux dans la suite.

Alagöz et Azizoğlu (2003) ont traité un problème sur des machines parallèles et identiques, sous contraintes d'éligibilité des machines. Les auteurs ont considéré la somme des temps de passage dans le système comme critère d'efficacité de l'ordonnancement et le nombre de jobs qui sont traités sur une machine différente après le réordonnancement comme mesure de stabilité. Plusieurs heuristiques ont été développées pour établir un ensemble de solutions, afin de minimiser un critère combinant l'efficacité et la stabilité de l'ordonnancement.

Yin *et al*. (2016) ont étudié un problème sur des machines parallèles et identiques. Les auteurs ont considéré la somme des dates de fin des jobs comme critère d'efficacité. En parallèle, ils ont proposé deux critères de stabilité, la déviation maximale des dates de fin des jobs et la somme des retards virtuels. Lorsqu'un job est ordonnancé pour la première fois, une date est communiquée au client. Quand cette date change, un retard virtuel est compté. Les perturbations dans leur cas sont dues aux pannes des machines. Un algorithme en temps pseudo-polynomial est développé pour résoudre ce problème.

Hamzadayi et Yildiz (2016) se sont intéressés à un problème sur des machines parallèles et identiques, dans lesquelles les séquences dépendent du temps de *setup*, avec un seul serveur. La politique de réordonnancement évènementielle a été utilisée avec comme objectif la minimisation du Makespan. Les auteurs ont utilisé le recuit simulé comme méthode de résolution approchée.

Le Tableau 5 présente une bibliographie des travaux de réordonnancement sur des machines parallèles.

Tableau 5. Bibliographie des travaux de réordonnancement sur des machines parallèles

Référence	Perturbations	Mesure de performance			Stratégie et/ou méthode	Contrainte de blocage	Méthode de résolution
		Efficacité	Stabilité	Robustesse			
Alagöz et Azizoğlu (2003)	EM	Temps d'écoulement total	Nombre de jobs qui changent de machine	N/A	Décalage vers la droite	N/A	Heuristiques adaptées
Duenas et Petrovic (2008)	JA	Makespan	Somme des déviations absolues des dates de début des jobs	N/A	Prédictive-Réactive, Décalage vers la gauche, Génération d'un nouvel ordonnancement	N/A	Système d'aide à la décision basé sur la logique floue
Tang et Zhang (2009)	PM	Somme pondérée des dates de fin des jobs	Somme des déviations des dates de fin des jobs	N/A	Réparation de l'ordonnancement existant, Evènementielle	N/A	PLNE, Relaxation lagrangienne
Aktürk *et al*., (2010)	PM	Temps match-up	N/A	N/A	Décalage vers la droite avec des durées de jobs contrôlables	N/A	Heuristique adaptée
Hamzadayi et Yildiz (2016)	NJ, PM, RM	Makespan	N/A	N/A	Evènementielle, Régénération	N/A	Recuit simulé, règles de répartition
Yin *et al*. (2016)	PM	Somme des dates de fin des jobs	Déviation maximale des dates de fin des jobs, somme des retards virtuels	N/A	Réordonnancement partiel	N/A	Algorithme pseudo polynomial
Baykasoğlu *et al*. (2018)	PM, NJ, JA, CDE, CTL	Makespan, latence maximale	N/A	N/A	Évènementielle, Périodique	N/A	Algorithme multi start et constructif
Qiao *et al*. (2018)	PM, CDS	Taux d'utilisation des équipements	Déviation absolue des dates de début des jobs	N/A	Prédictive-Réactive, Réordonnancement partiel, Décalage vers la droite	N/A	Simulation
Wang, D *et al*. (2020)	JA	Somme des dates de fin des jobs	Déviation du Makespan	N/A	Evènementielle	N/A	PLNE, algorithme évolutionnaire
Wang, S *et al*. (2020)	NJ	Makespan, Somme des dates de fin des jobs	N/A	N/A	Complètement réactive, proactive, hybride	N/A	PLNE, Heuristiques basée sur les règles de priorité
Letsios *et al.* (2021)	VDJ, PM	Makespan	N/A	Rapport entre Makespan initial et le nouveau	Robuste	N/A	Optimisation lexicographique, Branch and bound

NJ : Nouveaux jobs, JA : Jobs annulés, PM : Pannes machines, RM : Réparation machines, EM : Eligibilité machines, CDE : Changement de la date d'échéance, CTL : Changement de la taille du lot, CDS : changement de la date de disponibilité des jobs, VDJ : Variation des durées des jobs, N/A : Non applicable

3.3.3 Réordonnancement dans des ateliers de type Flowshop

Dans un environnement de machines en série, un ensemble de jobs doivent être exécutés sur un ensemble de machines placées en série. L'ordre de passage des jobs sur les machines est défini selon le type d'atelier. Trois types d'atelier existent :

- L'atelier Flowshop : Tous les jobs doivent suivre le même ordre de passage sur les machines.
- L'atelier Jobshop : Chaque job a son propre ordre de passage sur les machines.
- L'atelier Openshop : L'ordre de passage des jobs sur les machines n'est pas défini, et peut donc varier d'une solution à l'autre.

Parmi ces trois types d'atelier, nous nous sommes intéressés, dans le cadre de cette thèse, à l'atelier flowshop car il est très souvent rencontré dans les milieux industriels. Nombreux sont les travaux qui ont étudié les problèmes de réordonnancement dans un atelier de type Flowshop.

Par exemple, Katragjini *et al*. (2013) ont considéré ce problème de type flowshop. Trois types de perturbations sont simultanément gérées, l'arrivée de nouveaux jobs, les pannes des machines et la variation des dates de disponibilité des jobs. Les auteurs ont considéré le Makespan comme mesure d'efficacité de l'ordonnancement, et le nombre de jobs dont la date de début a changé comme mesure de stabilité. Ils ont proposé un algorithme glouton pour résoudre ce problème.

Certaines entreprises cherchent à augmenter leur flexibilité en multipliant le nombre de machines qui exécutent la même opération. Ces machines, considérés comme identiques, sont regroupées en étages. Le type d'atelier résultant de ce modèle est connu dans la littérature sous le nom de Flowshop hybride. Plusieurs travaux traitants le réordonnancement dans un atelier de type flowshop hybride ont aussi été analysés comme présenté dans la suite.

Rahmani and Ramezanian (2016) ont étudié un atelier de type flowshop hybride, dans lequel les perturbations sont dues à l'arrivée de nouveaux jobs. La stratégie prédictive-réactive a été adoptée. Dans un premier temps, ils génèrent un ordonnancement initial avec pour objectif la minimisation de la somme des retards pondérés comme critère d'efficacité. Puis lorsqu'une perturbation arrive, l'ordonnancement est mis à jour en incluant le nouveau job dans l'ordonnancement initial, avec l'objectif d'optimiser l'efficacité et la stabilité simultanément. La stabilité de l'ordonnancement est mesurée par la somme des déviations absolues des dates de fin des jobs.

He *et al*. (2020) ont traité, dans un atelier de type flowshop hybride, un problème dont les perturbations sont dues à l'arrivée de nouvelles commandes. L'efficacité de l'ordonnancement est mesurée par le Makespan et le temps de transport d'un job d'un étage à l'autre. Puisque les machines du même étage sont disposées en parallèle, la stabilité de l'ordonnancement est mesurée par le nombre de jobs qui changent de machine après chaque réordonnancement. Les auteurs ont aussi considéré la contrainte du temps de *setup*. Ils ont appliqué la politique évènementielle, et développé une méthode de résolution basée sur un algorithme évolutionnaire pour résoudre ce problème.

Les problèmes d'ordonnancement de type flowshop diffèrent aussi selon les technologies utilisées et les contraintes auxquelles le système est soumis. La plupart des travaux considèrent un espace de stockage illimité entre les machines. Or, dans les situations réelles, cet espace est

limité, voir nul. Cette réalité crée une situation de blocage, c'est-à-dire que la ressource reste bloquée par un job tant que l'espace de stockage pour le déposer est insuffisant. Dans la littérature, peu de travaux ont considéré les contraintes de blocages dans les problèmes de réordonnancement (Kecman *et al.*, 2013 ; Tao et Liu, 2019). En revanche, dans les problèmes d'ordonnancement statique, plusieurs types de contraintes de blocages ont été introduits aux systèmes flowshop. Dans les paragraphes qui suivent, nous décrivons en détail ces différentes contraintes de blocage.

3.3.3.1 Contrainte de blocage *RSb*

Cette contrainte de blocage classique illustre le cas où une machine reste bloquée par un job tant que celui-ci n'a pas commencé son exécution sur la machine suivante (Wang *et al.*, 2006). Elle est connue dans la littérature sous le nom *RSb* (*Release when Starting blocking*). Celle-ci illustre le cas de plusieurs applications industrielles, comme les chaînes robotisées des usines de production d'acier (Hall *et al.*, 1998). Pour illustrer cette contrainte, nous présentons dans la Figure 1 un exemple de quatre jobs et cinq machines. La matrice suivante *P* donne les durées des jobs sur les machines où p_{jm} correspond au temps d'exécution du job *j* sur la machine *m*.

$$p_{jm} = \begin{pmatrix} 1 & 1 & 2 & 1 & 2 \\ 1 & 3 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 1 \\ 3 & 2 & 1 & 1 & 1 \end{pmatrix} \; j\epsilon\{1,..,4\}, m\epsilon\{1,..,5\} .$$

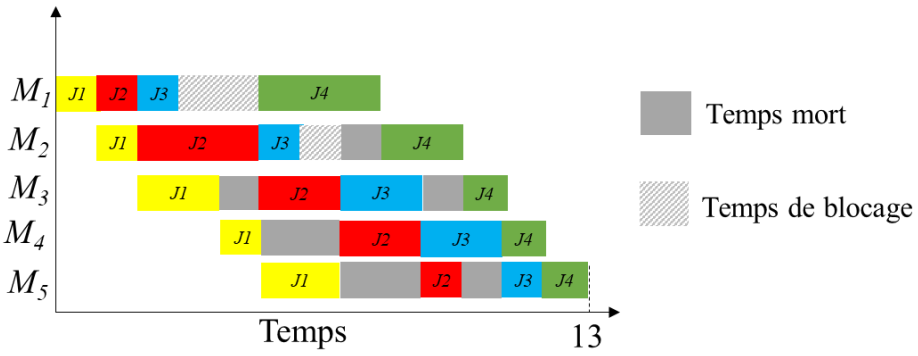

Figure 1. Illustration du blocage de type *RSb*

Dans la Figure 1, le job *J3* reste bloqué sur la machine M_1 tant que la machine M_2 exécute le job *J2*. Dans ce cas, le job *J4* ne peut pas commencer son exécution sur la machine M_1 tant que le job *J3* bloque M_1. Le Makespan de la solution est 13.

3.3.3.2 Contrainte de blocage *RCb*

Cette contrainte particulière nommée *RCb* (*Release when Completing blocking*) illustre la situation où le job reste bloqué sur une machine jusqu'à ce que son opération sur la machine suivante soit terminée et qu'il ait quitté cette machine. Cette date correspond à la date de début de l'opération du job sur la troisième machine. Cette contrainte a été considérée pour la première fois par Dauzère-Pérès *et al.* (2000). Des applications industrielles de ce type de contrainte ont été présentées par Martinez (2005). Elles concernent, par exemple, la fabrication de pièces métalliques qui doivent d'abord être chauffées dans un four, puis embouties sur une presse. Le four reste bloqué par une pièce ou un lot de pièces, jusqu'à ce que toutes les pièces qu'il contient soient embouties par la presse. Pour illustrer cette contrainte, nous reprenons

l'exemple du flowshop à 4 jobs et 5 machines en appliquant la contrainte de blocage *RCb* entre toutes les machines.

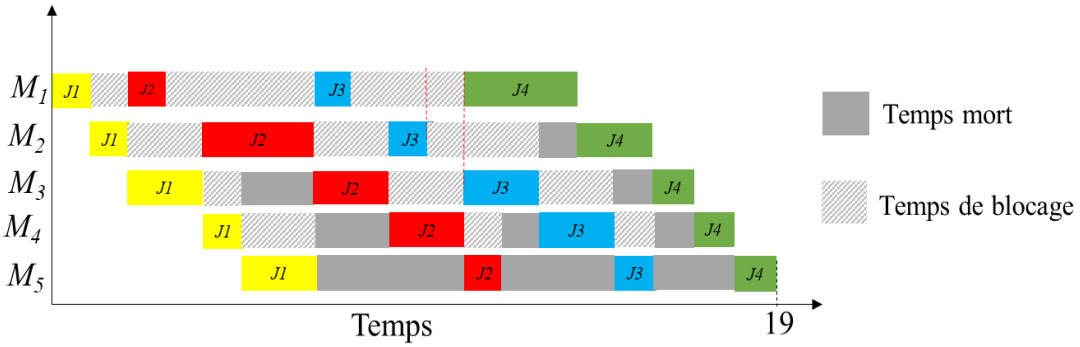

Figure 2. Illustration du blocage de type *RCb*

Sur la Figure 2, la valeur du Makespan de la solution augmente à 19, cela est dû à l'accroissement des temps de blocage. Par exemple, le job *J3* reste bloqué sur la machine M_1 jusqu'à ce que son opération sur la machine M_2 soit terminée, et qu'il ait quitté cette machine. En fin de compte, cette date correspond à la date de début de l'opération du job *J3* sur la machine M_3. Cette date est illustrée avec un trait en tirets rouge sur la figure.

3.3.3.3 Contrainte de blocage *RCb**

La contrainte de blocage *RCb** est une variante de la contrainte de blocage *RCb*, proposée pour la première fois dans Trabelsi *et al*. (2010). Elle illustre la situation où le job reste bloqué sur une machine jusqu'à ce que son opération sur la machine suivante soit terminée, sans tenir compte du fait qu'il ait quitté ou pas cette machine. Une application industrielle de ce type de contrainte est présentée dans Trabelsi (2012). Elle concerne le brassage du cidre. Des pommes doivent être versées dans un bain, puis pressées pour en tirer le jus. Les pommes d'un client ne peuvent pas être versées dans le bain avant que la totalité des pommes du client précédent ne soit pressées, car on ne peut pas mélanger les pommes de différents clients. Pour illustrer cette contrainte, nous reprenons l'exemple du flowshop à 4 jobs et 5 machines, en appliquant la contrainte de blocage *RCb** entre toutes les machines.

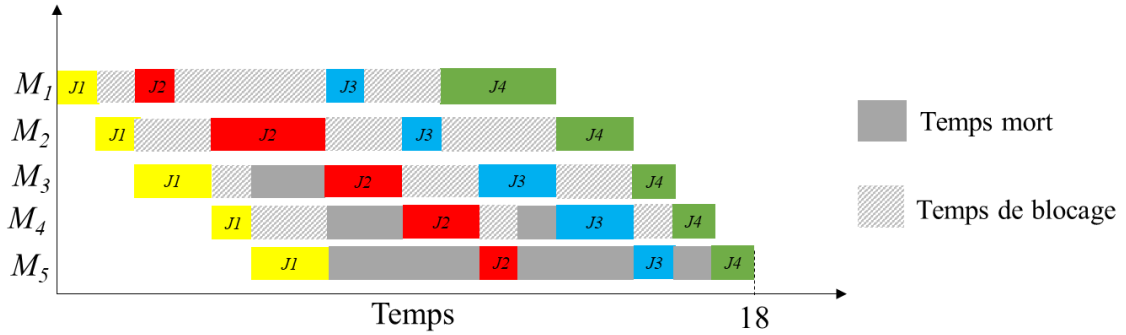

Figure 3. Illustration du blocage de type *RCb**

Dans la Figure 3, la valeur du Makespan de la solution a diminué d'une unité de temps par rapport au même exemple avec la contrainte *RCb*. Cette différence est due à l'avancement de

la date de début de la première opération du job *J4* sur la machine M_1. En effet, la machine M_1 est disponible pour traiter l'opération du job *J4* immédiatement après que le job *J3* a fini son opération sur la machine M_2. Dans ce cas, le job *J4* n'attend pas que le job *J3* quitte la machine M_2.

3.3.3.4 Contraintes de blocage mixtes

Ce type de contrainte décrit la situation dans laquelle plusieurs types de contraintes peuvent être mixés dans un seul système flowshop. C'est-à-dire un cas général dans lequel nous pouvons choisir le type de contrainte que nous souhaitons appliquer entre deux machines successives. La complexité du problème flowshop considérant les contraintes de blocage mixtes a été étudiée par Martinez (2005) puis Trabelsi (2012). Plus tard, des heuristiques ont été proposées par Trabelsi *et al*. (2012a). Le cas sans blocage peut, lui aussi, être combiné avec les autres contraintes de blocage. Dans ce qui suit, nous allons noter *Wb* (*Without blocking*) le cas du flowshop classique sans contrainte de blocage. Pour illustrer les contraintes de blocage mixtes, nous reprenons l'exemple du flowshop à 4 jobs et 5 machines, en appliquant des contraintes de blocage mixtes entre les machines.

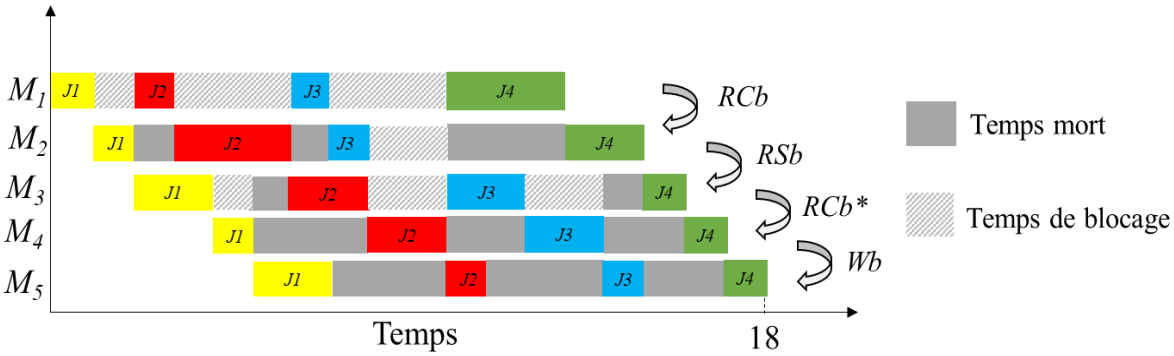

Figure 4. Illustration du blocage mixte

Dans Trabelsi *et al*. (2012b), les auteurs ont modélisé cette situation en intégrant un vecteur *V* qui représente une séquence de contraintes de blocage. Ce vecteur est adapté selon le nombre de machines que contient le système flowshop. Les éléments de ce vecteur représentent les contraintes de blocage entre deux machines successives. C'est-à-dire que V_k est la contrainte de blocage entre la machine M_k et M_{k+1}. Par exemple, V = (*RCb*, *RSb*, *RCb**, *Wb*) est le vecteur de l'exemple présenté dans le Figure 4. Si le système flowshop contient *M* machines, le vecteur *V* contient *M*-1 éléments.

Le Tableau 6 présente une bibliographie des travaux de réordonnancement dans des systèmes flowshop.

.

Tableau 6. Bibliographie sur les travaux de réordonnancement dans des systèmes flowshop

Référence	Perturbations	Mesure de performance			Stratégie et/ou méthode	Contrainte de blocage	Méthode de résolution
		Efficacité	Stabilité	Robustesse			
Chaari *et al*. (2011)	VDJ	Makespan	N/A	Déviation du Makespan	Robuste proactive	N/A	Algorithme génétique
El-Bouri (2012)	NJ	Retard moyen	N/A	N/A	Complétement réactive (règles de répartition)	N/A	Répartition coopérative
Katragjini *et al*. (2013)	NJ, PM, CDS	Makespan	Déviation absolue des dates de début des jobs	N/A	Prédictive-Réactive, Evènementielle	N/A	Algorithme glouton itératif
Rahmani et Heydari (2014)	NJ, VDJ	Makespan	Somme des déviations des dates de fin des jobs	Déviation du Makespan	Robuste proactive,	N/A	Heuristique, FIFO, M-SPT, M-LPT, règles de Johnson
Li *et al*. (2015)	PM, NJ, JA, VDJ, CDS	Makespan	Nombre de jobs perturbés	N/A	Réparation de l'ordonnancement existant, Evènementielle	N/A	Optimisation basée sur l'enseignement et l'apprentissage
Nagasawa *et al*. (2015)	VDJ	Pic de consommation électrique et coût de stock	N/A	Déviation du pic de consommation électrique	Robuste avec des temps d'inactivité	N/A	Simulation (logiciel Gurobi)
Tang *et al*. (2016)	PM, NJ	Makespan, consommation d'énergie	N/A	N/A	Évènementielle	N/A	Optimisation par essaims particulaires
Han *et al*. (2017)	PM	Makespan, Retard	Somme des déviations des dates de fin des jobs	Déviation de la fonction objectif	Robuste	N/A	Algorithme évolutionnaire multi objectif robuste
Liu *et al*. (2017)	NJ, PM	Temps d'écoulement total	Insatisfaction client	Déviation des dates de fin des jobs	Robuste proactive, réactive	N/A	Algorithme évolutionnaire multi objectif
Liu (2019)	NJ	Makespan, coût d'externalisation	Nombre de jobs perturbés	N/A	Réordonnancement partiel	N/A	PLNE, Descente de voisinage
Peng *et al*. (2019)	PM, NJ, CDS	Makespan	Nombre de jobs qui ont changé de position	N/A	Évènementielle	N/A	Descente de voisinage
Framinan *et al*. (2019)	VDJ	Makespan	N/A	N/A	Prédictive-Réactive, Évènementielle	N/A	Algorithme glouton itératif
Zhang *et al*. (2019)	PM, JA	Makespan	Déviation absolue des dates de début des jobs, nombre de jobs qui changent de machine	N/A	Prédictive-Réactive, Évènementielle	N/A	Métaheuristique, algorithme des oiseaux migrateurs
He *et al*. (2020)	NJ	Makespan, coût de production, coût de transport	Déviation maximale des jobs qui changent de machine	N/A	Évènementielle	N/A	Algorithme génétique de tri non dominé (NSGA)
Yang *et al*. (2021)	PM, NJ, CDS	Temps d'écoulement total	Insatisfaction client	Déviation des dates de fin des jobs	Robuste proactive, réactive	N/A	Algorithme génétique de tri non dominé (NSGA),

											Algorithme évolutionnaire multi objectif

NJ : Nouveaux jobs, JA : Jobs annulés, PM : Pannes machines, CDS : changement de la date de disponibilité des jobs, VDJ : Variation des durées des jobs, M-SPT : Modified Shortest Processing Time, M-LPT : Modified Longest Processing Time, N/A : Non applicable.

4. Synthèse bibliographique et présentation du problème

Dans ce paragraphe, nous établissons une synthèse bibliographique et présentons le problème que nous allons aborder dans cette thèse.

4.1 Synthèse bibliographique

Dans les paragraphes précédents, nous avons défini plusieurs termes liés au concept de réordonnancement, notamment les types de perturbations, les mesures de performance, les modèles, les stratégies, et les différents environnements machines qui ont été étudiés. Nous avons aussi présenté des travaux qui se sont intéressés au réordonnancement, catégorisés par type d'environnement machine. Dans ce paragraphe, nous fournissons une synthèse bibliographique. L'objectif est d'établir une analyse critique des travaux existants dans la littérature, et de pouvoir en extraire une nouvelle problématique de recherche dans laquelle nous pouvons faire de nouvelles contributions dans le domaine du réordonnancement.

La synthèse bibliographique nous a révélé plusieurs constats. Premièrement, malgré le grand nombre de travaux qui existent dans la littérature sur le réordonnancement, les contraintes de blocages ont reçu peu d'attention de la part des chercheurs. La majorité des travaux traitant des systèmes flowshop considèrent une capacité de stockage infinie entre les machines. En réalité, cette capacité de stockage est limitée, voire nulle dans certaines situations. Ainsi, il est intéressant de considérer les contraintes de blocage dans nos travaux pour analyser leur effet sur le réordonnancement des jobs.

Par ailleurs, la plupart des travaux considèrent le Makespan ou la somme des dates de fin des jobs comme critères pour mesurer l'efficacité de l'ordonnancement. Ces critères ne tiennent compte que de la durée des jobs pour répondre aux perturbations. Or, dans plusieurs applications réelles, lorsqu'il s'agit d'arrivée de nouvelles commandes, le facteur de priorité entre également en jeu, à savoir l'importance des clients dans les milieux industriels ou bien le niveau d'urgence des patients dans les milieux hospitaliers. Certaines opérations, malgré leur courte durée, sont plus importantes que d'autres vu leurs priorités ou leurs niveaux d'urgence. Dans ce cas, le fait de les retarder à chaque apparition d'une nouvelle perturbation, afin de minimiser un critère qui ne dépend que des durées des opérations, semble peu réaliste.

De même, pour les critères de stabilité, les auteurs considèrent souvent la déviation ou la déviation absolue des dates de fin, ou des dates de début des jobs, comme un critère pour mesurer la stabilité de l'ordonnancement. Ce critère mesure la différence entre la date de début (ou de fin) d'un job, avant et après la perturbation. C'est-à-dire que, dès qu'un job change de position, une pénalité est comptée. Cependant, le mouvement des jobs importants n'est pas pris en compte. En appliquant les poids des jobs à la mesure de stabilité, la pénalité du mouvement d'un job important ou plus urgent sera plus élevée. Ainsi, il sera plus difficile de déplacer les jobs qui ont un poids important. À notre connaissance, peu de travaux ont considéré les poids des jobs dans la mesure de stabilité.

En vertu des observations susmentionnées, une nouvelle mesure de performance a été proposée dans le cadre de cette thèse. Cette dernière combine simultanément un nouveau critère d'efficacité de l'ordonnancement avec un nouveau critère de stabilité. Cette mesure est inspirée des cas d'ordonnancement des blocs opératoires, mais elle peut s'appliquer également aux cas industriels. Le paragraphe ci-dessous présente une description du problème ainsi que la motivation qui nous a conduit à l'étudier.

4.2 Présentation du problème

Parmi les services hospitaliers, le bloc opératoire représente 33% des coûts de l'hôpital. Il est ainsi parmi les services les plus coûteux (Macario *et al*., 1995 ; Zhu *et al*., 2019). Pour cela, l'ordonnancement des blocs opératoires a été identifié comme un champ très important pour les preneurs de décision au sein de l'hôpital. D'autre part, les trop longues attentes avant d'être opérés sont parmi les réclamations les plus reçues de la part des patients (Cardoen *et al*., 2010). Par conséquent, il est pertinent de réagir sur ces deux points. La minimisation du temps d'attente des patients devant les blocs opératoires par l'ordonnancement des opérations, permettra de réduire les coûts hospitaliers et d'améliorer la satisfaction des patients. En pratique, il existe deux types d'opérations : les opérations électives qui sont connues à l'avance, et les opérations urgentes qui ne sont pas connues à l'avance et qui peuvent apparaître lors de l'arrivée des patients urgents (Bouguerra et *al*. 2020). Celles-ci provoquent une perturbation au planning déjà établi. D'autre part, selon (Abedini *et al*., 2016), les priorités chirurgicales peuvent être divisées en cinq grands groupes. Ainsi, nous allons considérer pour chaque opération un poids représentant son niveau d'urgence. L'objectif sera de modéliser ce cas pratique comme étant un problème de réordonnancement, en utilisant la stratégie prédictive-réactive. La résolution de ce problème se fera en deux phases. La première phase est la phase prédictive, qui consiste à ordonnancer les opérations électives avec comme objectif la minimisation du temps d'attente des patients, pondéré par leur poids (niveau d'urgence) représentant le critère d'efficacité de l'ordonnancement. La deuxième phase est la phase réactive, qui consiste à réordonnancer ces opérations à chaque fois qu'une opération urgente arrive. Dans cette deuxième phase, l'objectif est d'optimiser l'efficacité combinée avec la stabilité pour ne pas trop dévier par rapport à ce qui a été planifié. La stabilité dans ce cas est mesurée par la différence entre la date de fin de l'opération lorsque celle-ci est ordonnancée pour la première fois et sa date de fin dans l'ordonnancement après la perturbation. Cette quantité est pondérée par le niveau d'urgence de l'opération. L'opération est identifiée au job, le niveau d'urgence au poids, et les salles opératoires aux machines.

Ce problème peut aussi être une illustration des cas industriels, comme celui décrit dans Guo *et al*. (2016), où le temps d'attente représente la période d'attente des matériaux devant un four de réchauffage, avec l'arrivée de nouvelles commandes considérées comme des perturbations. Les poids des jobs peuvent représenter, dans ce cas, les priorités des clients. Les contraintes de blocage mixtes modélisent le cas où les capacités de stockage entre les machines d'un problème de type flowshop sont limitées.

En conclusion, le problème auquel nous allons nous attaquer dans le cadre de cette thèse, s'intéressera au réordonnancement des jobs en réponse à des perturbations dues à l'arrivée ou l'annulation des jobs. Avant l'apparition des perturbations, un ensemble de jobs dont les informations sont connues à l'avance, doivent être traités sur une ou plusieurs machines, avec l'objectif d'optimiser le critère d'efficacité. Après l'apparition des perturbations, l'ordonnancement déjà établi est mis à jour et un nouvel ordonnancement est établi avec l'objectif d'optimiser simultanément l'efficacité et la stabilité de l'ordonnancement.

5. Conclusion

Dans ce chapitre, deux objectifs ont été visés. Premièrement, nous avons défini le concept du réordonnancement et ses applications réelles, puis nous avons présenté les différents termes liés à ce concept, à savoir les types de perturbation, les mesures de performance, les méthodes

et les stratégies de réordonnancement, ainsi qu'aux environnements machines dans lesquels les problèmes de réordonnancement ont été étudiés. Cette analyse approfondie de l'état de l'art nous a permis d'établir une synthèse bibliographique pour contextualiser notre travail. Deuxièmement, sur la base de la synthèse bibliographique, de nouvelles problématiques ont été révélées qui ont permis la proposition d'un nouveau sujet de recherche.

Dans l'état de l'art, nous avons défini le concept du réordonnancement en fournissant les cas pratiques où ce problème a été rencontré, notamment dans les milieux industriels en réponse aux perturbations dues à l'arrivée de nouvelles commandes, aux pannes des machines ou au manque de matière première, etc. Dans les milieux hospitaliers aussi, ce problème a été rencontré en réponse aux perturbations dues à l'apparition des patients urgents. Nous avons aussi identifié les mesures utilisées pour évaluer la performance de l'ordonnancement dans les milieux dynamiques, à savoir les mesures d'efficacité, les mesures de stabilité et les mesures de robustesse. Ensuite, les différentes méthodes et stratégies de réordonnancement ont été présentées, en précisant les avantages et les inconvénients de chacune. Puis, nous avons présenté des applications du réordonnancement sur différents environnements de machines, en particulier pour le cas d'une machine unique, pour des machines parallèles et pour des problèmes de type flowshop. Finalement, une synthèse bibliographique des travaux récents a été fournie.

À travers cette synthèse bibliographique, nous avons pu constater que les contraintes de blocages entre les machines ont été très peu étudiées par les chercheurs qui travaillent sur les problèmes de réordonnancement et qu'en général ils considèrent une capacité de stockage infinie. D'autre part, la plupart des travaux considèrent le Makespan ou la somme des dates de fin des jobs comme critères pour mesurer l'efficacité de l'ordonnancement. Ces critères ne tiennent compte que de la durée des jobs pour construire un nouvel ordonnancement en réponse aux perturbations. Or, il est judicieux de considérer pour chaque job un poids représentant sa priorité. De même, pour les critères de stabilité, les auteurs considèrent souvent la déviation ou la déviation absolue des dates de fin, ou des dates de début des jobs, comme critère pour mesurer la stabilité de l'ordonnancement. De plus, le mouvement des jobs importants n'est pas pris en compte. En affectant les poids des jobs à la mesure de stabilité, la pénalité du mouvement d'un job important ou plus urgent sera plus élevée et il sera plus difficile de déplacer les jobs de poids important.

Dans le chapitre suivant, nous présentons notre premier travail, dans lequel nous nous intéressons au problème de minimisation du temps d'attente moyen pondéré et la déviation des dates de fin pondérées, avec des poids variables dans un problème de réordonnancement dans le cas d'une machine unique.

Chapitre 2 : Optimisation de l'efficacité et de la stabilité dans un problème de réordonnancement : cas d'une machine unique

Dans ce chapitre, nous avons étudié le problème de minimisation du temps d'attente moyen pondéré et de la déviation de la date de fin pondérée dans un problème de réordonnancement sur une machine unique, en considérant des poids des jobs variables. Après avoir décrit le problème en précisant les motivations qui nous ont conduit à le considérer, une formulation mathématique basée sur la PLNE a été proposée. Dans la partie des résultats expérimentaux, nous avons ensuite analysé l'impact de la variation du coefficient d'efficacité-stabilité sur la performance de l'ordonnancement, l'influence de la variation des poids des jobs sur le système, ainsi que l'évolution des temps de résolution du modèle en fonction du nombre de jobs pour identifier les limites de notre formulation.

1. **Introduction**

Le problème de réordonnancement sur une machine unique est un problème générique, et peut être utilisé comme une base pour d'autres problèmes plus compliqués dans des environnements contenant des machines multiples (Pinedo, 2012). En effet, les problèmes d'ordonnancement dans des environnements plus complexes sont souvent décomposés en sous-problèmes à machine unique. Par exemple, Denton *et al*. (2007) ont développé un modèle d'optimisation stochastique pour la planification quotidienne d'une salle d'opération unique, puis ils ont étendu leur modèle au cas à plusieurs ressources. La décision de choisir l'environnement à machine unique est aussi motivée par le fait que cet environnement représente plusieurs cas réels. Par exemple, une salle d'opération d'un hôpital où les opérations électives et urgentes doivent être traitées, ou bien le cas d'une machine goulot dans un atelier de production.

Ce chapitre s'intéresse à un problème de réordonnancement sur une machine unique qui subit des perturbations dues à l'arrivée de nouveaux jobs. La stratégie prédictive-réactive a été adoptée dans ce travail, consistant dans la phase prédictive à résoudre un problème d'ordonnancement classique, ayant comme objectif de minimiser le critère d'efficacité seul représenté par le temps d'attente moyen pondéré. À chaque arrivée d'une perturbation, intervient la phase réactive qui consiste à réordonnancer les jobs, avec cette fois comme objectif de minimiser les deux critères simultanément, le critère d'efficacité de l'ordonnancement, et le critère de stabilité représenté par la déviation moyenne des dates de fin pondérées par les poids des jobs. Ces deux critères sont associés dans la phase réactive par un coefficient d'efficacité-stabilité, noté α.

Dans l'analyse des résultats numériques, il a été constaté que souvent les jobs qui ont un faible poids sont reportés, même s'ils apparaissent tôt. Ceci est dû au fait qu'à chaque fois qu'un job avec un poids élevé apparaît, il est priorisé vu son importance. Ce job sera planifié au début de la séquence, et cela fera reporter les jobs moins importants. Si cette situation se produit à chaque fois, il y a un grand risque d'ignorer les jobs qui ont un poids faible peu importe leur date de disponibilité. Pour remédier à cette situation, nous avons conçu un nouveau concept qui consiste à faire augmenter les poids des jobs en fonction du temps. Grâce à ce concept, quand un job reste longtemps dans le système, son poids va augmenter. Cela va l'aider à être considéré dans la séquence au fil du temps.

Un modèle basé sur la PLNE a été développé pour résoudre ce premier problème. Il contribue à la problématique du réordonnancement par :

- L'étude d'une nouvelle mesure de performance inspirée d'une situation réelle, où le temps d'attente moyen pondéré est considéré comme un critère d'efficacité, et la déviation moyenne des dates fin des jobs pondérée, est considérée comme un critère de stabilité.
- L'implémentation d'un modèle basé sur la PLNE et d'une stratégie prédictive-réactive pour le réordonnancement des jobs en réponse aux perturbations dues à l'arrivée de nouveaux jobs.
- Le développement d'un nouveau concept qui consiste à augmenter les poids des jobs en fonction du temps pour aider les jobs de faibles poids à ne pas être ignorés.

2. Description du problème

Nous étudions un problème de réordonnancement sur une machine unique avec des perturbations causées par l'arrivée de nouveaux jobs. Ce problème sera résolu en deux phases. La phase prédictive, avant l'apparition d'une perturbation, consiste à minimiser le Temps d'Attente Moyen Pondéré par les poids des jobs, que l'on notera « *TAMP* ». La phase réactive, qui commence après l'apparition d'une perturbation, consiste à minimiser simultanément le *TAMP* ainsi que la Déviation Moyenne des Dates de Fin Pondérées par les poids des jobs, que l'on notera « *DMDFP* ». Dans ce qui suit, nous présentons une description détaillée de chaque critère.

2.1 Critère d'efficacité de l'ordonnancement : Temps d'attente moyen pondéré

Dans un problème d'ordonnancement, un ensemble de n jobs, $N = \{1, 2, ..., n\}$, doivent être traités sur une machine unique. A chaque job j est associé un poids w_j, une durée p_j et une date de disponibilité (date de début au plus tôt) r_j. Quand le job j débute son exécution à la date S_j, il va procéder sur la machine jusqu'à sa date de fin C_j sans interruption, *i.e.* $C_j = S_j + p_j$. Le temps d'attente « *Waiting Time* » d'un job j, noté W_j, représente la période durant laquelle le job attend avant son exécution sur la machine, c'est-à-dire la période entre sa date de disponibilité r_j et sa date d'exécution S_j. Par conséquent, $W_j = S_j - r_j = C_j - p_j - r_j$. La Figure 5 illustre le temps d'attente d'un job j sur une machine unique.

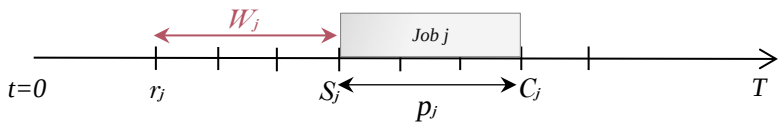

Figure 5. Temps d'attente W_j du job j

Dans cette phase, dite la ***phase prédictive***, toutes les informations concernant les jobs (w_j, r_j, p_j) sont connues à l'avance. Au cours de cette phase hors ligne, l'objectif est de déterminer la séquence qui va minimiser la moyenne des temps d'attente des jobs, pondérés par leur poids, $min \sum_{j \in N} w_j W_j$, représentant le ***critère d'efficacité***. Ce problème peut être représenté dans la notation standard des problèmes d'ordonnancement (Graham *et al.*, 1979) par $1|r_j|\sum w_j W_j$. Il a été défini la première fois par Rinnooy Kan (1976), qui a classifié sa résolution comme *NP-Difficile* sur une machine unique.

2.2 Critère de stabilité de l'ordonnancement : Déviation des dates de fin pondérées

Lorsqu'un nouveau job *j'* arrive, l'ordonnancement établi doit être mis à jour pour répondre à cette nouvelle perturbation. Cette phase est appelée la phase réactive ou la phase online. Tous les jobs qui ont commencé leur exécution sur la machine avant la date d'apparition de la perturbation vont être supprimés de l'ensemble des jobs. On note *No* l'ensemble des jobs restants. L'ensemble des jobs N est alors mis à jour, tel que $N \leftarrow No \cup \{j'\}$. Cela signifie que le nouveau job *j'* est combiné avec les jobs existants qui n'ont pas encore commencé leur exécution. Ce nouvel ensemble N va alors être réordonnancé. Dans le cas d'annulation d'un job dont l'exécution n'a pas débuté, ce job sera également supprimé de l'ensemble des jobs restants et ne sera plus dans l'ensemble *No*. Le nouvel ensemble N sera donc le même que *No*, $N \leftarrow No$.

Dans cette seconde phase, *un* ***critère de stabilité*** est considéré. On note Co_j la date de fin du job j lorsque celui-ci est ordonnancé pour la première fois. Quand un nouveau job arrive, l'ordonnancement est mis à jour et on note alors C_j la date de fin du job j après cette mise à jour. La différence entre C_j et Co_j est alors utilisée pour évaluer la déviation du job j après la perturbation. $D_j = \max\{0, C_j - Co_j\}$ représente ainsi la déviation du job j, mesurée entre sa date de fin lorsqu'il est planifié pour la première fois et sa date de fin après le réordonnancement. Lorsque ce job est retardé, une pénalité d'instabilité est comptée, tandis que lorsqu'il est avancé, rien n'est considéré. L'objectif est de réduire l'instabilité du système à travers la minimisation de la *DMDFP*, i.e. $min\ \sum_{j\in No} w_j D_j$. Le critère de stabilité n'évalue que les jobs qui appartiennent à l'ensemble *No*, puisque le nouveau job n'existait pas avant la perturbation. Le fait d'associer des poids à la mesure de stabilité rend plus difficile le déplacement des jobs qui ont un poids élevé. C'est par exemple le cas des patients qui ont un niveau d'urgence élevé dans le cas du réordonnancement des blocs opératoires.

2.3 Association du critère d'efficacité et de stabilité

Dans la phase réactive, l'objectif est non seulement d'optimiser l'efficacité de l'ordonnancement, mais également sa stabilité. Ces deux aspects sont donc regroupés dans une seule fonction objectif, et associés par un coefficient d'efficacité-stabilité $\alpha \in [0,1]$. Ce coefficient est une pondération de chaque partie de la fonction objectif, telle que α est associé à l'efficacité et (1 - α) à la stabilité, avec α un nombre réel qui est compris entre zéro et un. La fonction objectif de la phase réactive est la suivante :

$$f_1 = \alpha \sum_{j\in N} w_j W_j + (1-\alpha) \sum_{j\in No} w_j D_j$$

Le coefficient α est également introduit pour aider les preneurs de décision à trouver un compromis entre les deux aspects du problème, l'efficacité et la stabilité de l'ordonnancement. Si nous attribuons plus d'importance à l'efficacité, la stabilité aura alors moins d'importance et vice-versa. Dans le cadre de ce travail, nous supposons que l'efficacité est le critère principal. C'est pour cette raison que dans les tests numériques, la valeur de α n'ira pas au-dessous de 0,5. En effet, si α est trop petit, l'efficacité de l'ordonnancement devient négligeable et la situation est irréaliste. Dans la phase réactive, le problème peut être représenté dans la notation standard des problèmes d'ordonnancement par $1|r_j|\alpha \sum w_j W_j + (1-\alpha) \sum w_j D_j$.

3. Stratégie prédictive-réactive pour le problème $1|r_j|\ \alpha\ \Sigma w_j W_j + (1-\alpha)\ \Sigma w_j D_j$

Comme dit précédemment, la stratégie prédictive-réactive proposée consiste, dans la phase prédictive, à résoudre un problème classique d'ordonnancement ayant comme objectif la minimisation du *TAMP*, i.e. $min\ \sum_{j\in N} w_j W_j$, puis, dans la phase réactive, à parcourir un horizon de simulation étape par étape, itérativement. À chaque étape, le processus vérifie si un job arrive. Si c'est le cas, l'ensemble *N* des jobs est mis à jour et un nouvel ordonnancement est établi. Dans cette deuxième phase, l'objectif est de minimiser la fonction f_1 qui combine l'efficacité et la stabilité de l'ordonnancement.

Pour ce faire, nous considérons un horizon de simulation fini [0, *T*]. Cet horizon est discrétisé en des périodes Δt (voir Figure 6), représentant la durée de chaque période ou le pas de la simulation. Les modèles à flux discret sont souvent considérés comme des modèles plus réalistes pour la production discrète. Ils sont utilisés pour l'évaluation des performances, pour

suivre les pièces individuellement, ou dans le contrôle du flux en temps réel (Tighazoui *et al*., 2019).

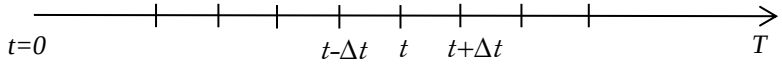

Figure 6. Discrétisation de l'horizon de simulation

Dans ce travail, il est supposé que l'arrivée d'un job ne puisse se produire qu'aux instants multiples de Δt. Quand un nouveau job arrive à une date t, celle-ci sera sa date de disponibilité. Pour simplifier les calculs, nous supposons que $\Delta t = 1$ unité de temps. La même unité de temps est utilisée pour les paramètres p_j et r_j. Nous avons également défini un paramètre binaire $\theta(t)$ qui est égal à 1 si un job arrive à l'instant t, et 0 sinon. Nous supposons qu'un seul job arrive lorsque $\theta(t) = 1$. Le logigramme de la Figure 7 décrit la stratégie prédictive-réactive proposée pour résoudre ce problème.

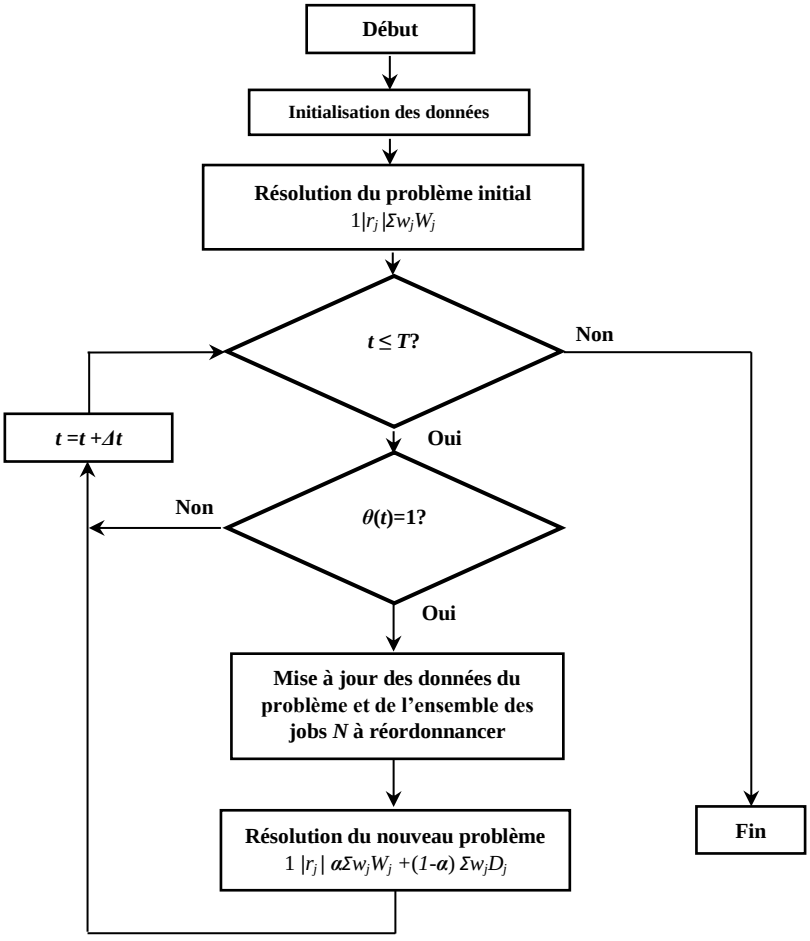

Figure 7. Stratégie prédictive-réactive pour le problème $1|r_j|\ \alpha\ \Sigma w_j W_j+(1-\alpha)\ \Sigma w_j D_j$

La stratégie consiste à parcourir l'horizon de simulation T étape par étape, et à vérifier, à chaque étape Δt si un job est arrivé grâce à l'état du paramètre $\theta(t)$. Si un job est arrivé, le processus met à jour les données du problème et l'ensemble des jobs N à réordonnancer, puis il résout le nouveau problème. Sinon, il passe à l'étape suivante.

4. Modèles mathématiques

Dans cette section, deux modèles mathématiques basés sur la PLNE sont proposés. Le premier est dédié au modèle mathématique avant la perturbation, et le second au modèle mathématique après la perturbation. Dans cette section, nous présentons d'abord la première partie du modèle. Ensuite, nous présentons une étude de complexité, ainsi qu'une borne inférieure pour le problème $1|r_j| \Sigma w_j W_j$. Enfin, nous présentons la seconde partie du modèle mathématique.

4.1 Modèle mathématique avant la perturbation

Le modèle mathématique proposé est implémenté selon une formulation basée sur les positions, i.e. affection d'un job à une position. Plusieurs chercheurs ont opté pour cette formulation pour résoudre les problèmes d'ordonnancement sur une machine unique, comme par exemple, Eren (2009), Baker et Keller (2010), Kooli et Serairi (2014), ainsi que Guo et Xie (2017). Cependant, la majorité de ces travaux considèrent le Makespan comme critère. Pour correspondre au problème de la minimisation du *TAMP*, d'autres variables et paramètres ont été ajoutés, tels que les poids des jobs, les dates de sortie et le temps d'attente. La formulation du problème est décrite dans la sous-section suivante.

4.1.1 Formulation mathématique pour $1|r_j| \Sigma w_j W_j$

Dans cette sous-section, nous présentons la formulation mathématique dédiée au problème $1|r_j| \Sigma w_j W_j$.

Ensembles et indices

N : ensemble de jobs $\{1,2, \ldots, n\}$

K : ensemble de positions $\{1,2, \ldots, n\}$

j : indice des jobs, $j = 1,2, \ldots, n$

k : indice des positions, $k = 1,2, \ldots, n$

Paramètres

w_j : poids du job j

r_j : date de disponibilité du job j

p_j : durée du job j

$bigM$: grande valeur, $bigM = \sum_{j \in N} p_j + \max r_j$

Variables de décision

$$x_{jk} = \begin{cases} 1, & \text{Si le job } j \text{ est affecté à la } k^{ème} \text{ position} \\ 0, & \text{Sinon} \end{cases}$$

W_j : temps d'attente du job j

C_j : date de fin du job j

S_k : date de début du job placé en la $k^{ème}$ position

Cp_k : date de fin du job placé en $k^{ème}$ position

Fonction objectif

$$min \sum_{j \in N} w_j W_j$$

Contraintes

$$\sum_{j \in N} x_{jk} = 1, \ \forall k \epsilon K \quad (1.1)$$

$$\sum_{k \in K} x_{jk} = 1, \ \forall j \epsilon N \quad (1.2)$$

$$S_k \geq \sum_{j \in N} r_j x_{jk}, \ \forall k \epsilon K \quad (1.3)$$

$$S_{k+1} \geq S_k + \sum_{j \in N} p_j x_{jk}, \ \forall k \epsilon 1, \dots, n-1 \quad (1.4)$$

$$Cp_k = S_k + \sum_{j \in N} p_j x_{jk}, \ \forall k \epsilon K \quad (1.5)$$

$$C_j \geq Cp_k - bigM(1 - x_{jk}), \ \forall j \epsilon N, \ \forall k \epsilon K \quad (1.6)$$

$$C_j \leq Cp_k + bigM(1 - x_{jk}), \ \forall j \epsilon N, \ \forall k \epsilon K \quad (1.7)$$

$$W_j = C_j - r_j - p_j, \ \forall j \epsilon N \quad (1.8)$$

$$S_k, \ Cp_k, W_j \text{ et } C_j \geq 0, \ \forall j \epsilon N, \ \forall k \epsilon K \quad (1.9)$$

$$x_{jk} \in \{0,1\}, \ \forall j \epsilon N, \ \forall k \epsilon K \quad (1.10)$$

Signification des contraintes

- Contrainte (1.1) : Elle permet de n'avoir qu'un seul job par position.
- Contrainte (1.2) : Elle permet de n'avoir qu'une seule position par job.
- Contrainte (1.3) : Elle oblige que la date de début du job en $k^{ème}$ position soit supérieure ou égale à sa date de disponibilité.
- Contrainte (1.4) : Elle impose que la date de début du job en $(k+1)^{ème}$ position soit supérieure ou égale à la date de fin du job dans la position précédente ($k^{ème}$ position).
- Contrainte (1.5) : Elle impose que la date de fin du job en $k^{ème}$ position soit égale à sa date de début plus sa durée.
- Contraintes (1.6) et (1.7) : Si le job j est en $k^{ème}$ position, ces contraintes permettent d'imposer que la date de fin du job j est égale à la date de fin du job en position k. Comme *bigM* doit être une valeur suffisamment grande et en particulier supérieure à toutes les dates de fin des jobs, on a pris $bigM = \sum_{j \in N} p_j + \max r_j$.
- Contrainte (1.8) : Elle définit le temps d'attente du job j en fonction sa date de fin, sa date de disponibilité et sa durée.
- Contraintes (1.9) : Ce sont des contraintes de non-négativité qui contraignent toutes les variables à être supérieures ou égales à zéro.

- Contrainte (1.10) : Elle impose que les variables x_{jk} soient binaires.

4.1.2 Exemple

Dans cet exemple, les durées p_j, les dates de disponibilité r_j, et les poids w_j de 5 jobs sont fournis dans le Tableau 7.

Tableau 7. Données du problème initial sur une machine unique

Job	*A*	*B*	*C*	*D*	*E*
p_j	1	2	2	3	4
r_j	1	1	0	0	2
w_j	5	1	4	2	3

La séquence optimale obtenue pour la minimisation du *TAMP* est CAEDB. La valeur obtenue est $\sum_{j=1}^{5} w_j W_j = 31$. Les valeurs de la solution sont présentées dans la Tableau 8.

Tableau 8. La solution optimale du problème initial sur une machine unique

Job	*C*	*A*	*E*	*D*	*B*
p_j	2	1	4	3	2
r_j	0	1	2	0	1
w_j	4	5	3	2	1
C_j	2	3	7	10	12
$w_j W_j$	0	5	3	14	9

4.2 Complexité et borne inférieure du problème $1|r_j| \Sigma w_j W_j$

Dans cette section, nous présentons la complexité du problème $1|r_j| \Sigma w_j W_j$ et nous proposons une borne inférieure.

Théorème 1 : $1|r_j| \Sigma w_j W_j$ est fortement *NP-Difficile.*

Preuve : Selon Rinnooy Kan (1976), le critère $\Sigma w_j W_j$ est équivalent au critère $\Sigma w_j C_j$ car $W_j = C_j - r_j - p_j$, et r_j et p_j sont des constantes quelle que soit la séquence. Ainsi, $min \sum_{j=1}^{n} w_j (C_j - r_j - p_j)$ est équivalent à $min \sum_{j=1}^{n} w_j C_j + constante$. D'autre part, Lenstra *et al*. (1977) ont montré que le problème $1|r_j|\Sigma w_j C_j$ est fortement *NP-Difficile*. Labetoulle *et al*. (1984) ont prouvé que ce problème reste fortement *NP-Difficile* même si la préemption est autorisée. D'où, le problème $1|r_j| \Sigma w_j W_j$ est fortement *NP-Difficile*. ▫

Selon Nessah *et al*. (2006), quand toutes les dates de disponibilité des jobs sont égales, la règle *wSPT* (*Weighted Shortest Processing Time*) qui consiste à séquencer les jobs par ordre croissant de p_j/w_j, est optimale pour le problème $1||\Sigma w_j C_j$. Cependant, la règle *wSPT* ne permet pas de résoudre à l'optimal le problème lorsque les dates de disponibilité sont considérées. Dans ce qui suit, une borne inférieure pour le problème $1|r_j| \Sigma w_j W_j$ est présentée.

Théorème 2 : La solution obtenue avec *wSRPT* est une borne inférieure pour $1|r_j| \Sigma w_j W_j$.

Preuve : Ahmadi (1990) obtient une borne inférieure pour le problème $1|r_j|\Sigma C_j$ en autorisant la préemption et puis en utilisant la règle *SRPT* (*Shortest Remaining Processing Time*). Dans notre cas, nous considérons le problème de minimisation du *TAMP* sur une machine unique en autorisant la préemption, $1|r_j, pmtn|\Sigma w_j W_j$. Nous utilisons la règle *wSRPT* (*Weighted Shortest Remaining Processing Time*) appliquée au problème avec préemption pour obtenir une borne inférieure pour le problème initial $1|r_j| \Sigma w_j W_j$. En utilisant la même démarche que Ahmadi

(1990), on peut ainsi montrer que toutes les solutions admissibles pour le problème $1|r_j|\Sigma w_j W_j$ sont supérieures ou égales à la solution obtenue avec la règle *wSRPT* appliquée au problème avec préemption et que cette dernière est donc une borne inférieure du problème $1|r_j|\Sigma w_j W_j$. □

Nous considérons l'exemple du Tableau 7. La liste *wSPT* est donnée par {A, C, E, D, B}. Le Tableau 9 présente la solution obtenue par *wSRPT* appliquée au cas avec préemption.

Tableau 9. La solution obtenue avec *wSRPT*

Job	C_1	*A*	C_2	*E*	*D*	*B*
C_j	1	2	3	7	10	12
w_jW_j	-	0	4	3	14	9

Le job A est le premier dans la liste *wSPT*. Cependant, il n'est pas disponible à l'instant t = 0, nous avons alors ordonnancé le job C qui est disponible et qui est le deuxième de la liste, jusqu'à l'instant t = 1 où le job A est disponible. Le job C a été repris après l'exécution du job A. La valeur obtenue en utilisant *wSRPT* est $\sum_{j=1}^{5} w_j W_j = 30$, au lieu de 31 qui a été obtenue en résolvant le problème sans préemption.

4.3 Modèle mathématique après chaque perturbation

Ce second modèle est généré après l'apparition de chaque perturbation. Ainsi, les nouveaux jobs ont les mêmes paramètres que les anciens. A chaque job j, est associé un poids w_j, une durée p_j et une date de disponibilité r_j qui correspond à sa date d'arrivée ou sa date d'apparition dans le système. Le second modèle est présenté dans la sous-section qui suit.

4.3.1 Formulation mathématique pour $1|r_j|\ \alpha\ \Sigma w_jW_j+(1-\alpha)\ \Sigma w_jD_j$

Mise à jour de l'ensemble des jobs

No : L'ensemble des jobs existants qui n'ont pas encore commencé leur exécution.

j' : Le nouveau job.

$N \leftarrow No \cup \{j'\}$: L'ensemble des jobs à réordonnancer.

Nouveaux paramètres

Co_j : La date de fin originale du job j, calculée lorsque ce job est ordonnancé pour la première fois.

α : Coefficient d'efficacité-stabilité.

Variables de décision et fonction objectif

Les variables de décision x_{jk}, W_j, C_j, S_k, Cp_k sont aussi utilisées pour ce deuxième modèle, ainsi que D_j la déviation de la date de fin du job j. Par conséquent, l'objectif sera de minimiser la fonction f_1, telle que :

$$f_1 = \alpha \sum_{j \in N} w_j W_j + (1-\alpha) \sum_{j \in No} w_j D_j$$

Contraintes

Les contraintes (1.1) jusqu'à (1.10) sont aussi utilisées pour ce deuxième modèle, ainsi que deux contraintes additionnelles.

$$D_j \geq C_j - Co_j, \ \forall j \in No \tag{1.11}$$

$$D_j \geq 0, \ \forall j \in No \tag{1.12}$$

Les contraintes (1.11) et (1.12) permettent de calculer la valeur de la variable $D_j = \max\{0, C_j - Co_j\}$.

4.3.2 Exemple

Dans cet exemple, nous considérons les mêmes données que celles de l'exemple précèdent, en supposant que les jobs F et G arrivent quand la machine exécute la séquence déjà établie (voir Figure 8).

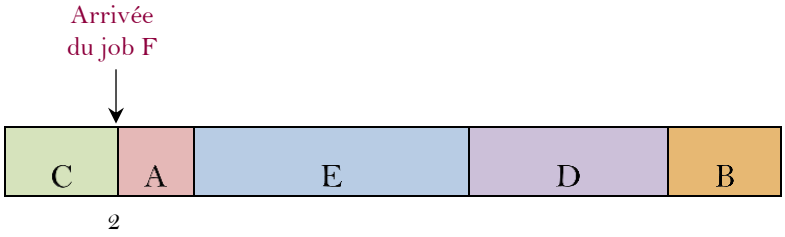

Figure 8. Perturbation causée par l'arrivée du job F

Le job F arrive à $t = r_F = 2$, avec une durée $p_F = 1$ et un poids $w_F = 5$. Le coefficient d'efficacité-stabilité est $\alpha = 0.5$. La solution optimale est CAFEDB avec une valeur $\alpha \sum_{j=1}^{6} w_j W_j + (1-\alpha) \sum_{j=1}^{5} w_j D_j = 24$. Les résultats sont présentés dans le Tableau 10.

Tableau 10. La solution optimale après l'apparition du job F

Job	*C*	*A*	*F*	*E*	*D*	*B*
p_j	2	1	1	4	3	2
r_j	0	1	2	2	0	1
w_j	4	5	5	3	2	1
C_j	2	3	4	8	11	13
$w_j W_j$	0	5	5	6	16	10
$w_j D_j$	0	0		3	2	1

Le job G arrive ensuite à $t = r_G = 3$, avec une durée $p_G = 1$ et un poids $w_G = 1$ (voir Figure 9).

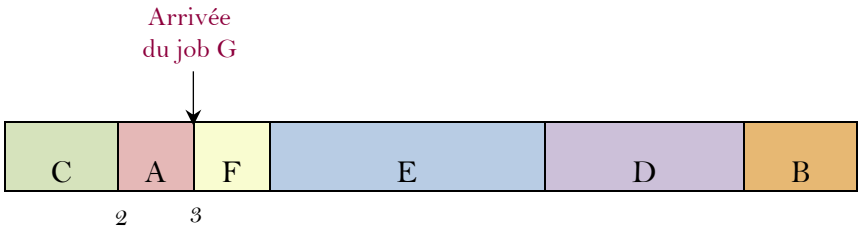

Figure 9. Perturbation causée par l'arrivée du job G

La solution optimale est CAFEDBG avec une valeur de la fonction objectif $\alpha \sum_{j=1}^{7} w_j W_j + (1-\alpha) \sum_{j=1}^{6} w_j D_j = 29$. Les résultats sont présentés dans le Tableau 11.

Tableau 11. La solution optimale après l'apparition du job G

Job	*C*	*A*	*F*	*E*	*D*	*B*	*G*
p_j	2	1	1	4	3	2	1
r_j	0	1	2	2	0	1	3
w_j	4	5	5	3	2	1	1
C_j	2	3	4	8	11	13	14
w_jWj	0	5	5	6	16	10	10
w_jD_j	0	0	0	3	2	1	

5. Etudes expérimentales

L'algorithme basé sur les PLNE a été programmé sur FICO Xpress IVE sur un PC Core i7, 2.90 GHz, RAM 8 Go. Il est composé de deux parties. La première partie consiste à résoudre le problème d'ordonnancement avec l'objectif de minimiser le *TAMP*. La deuxième partie consiste à réordonnancer les jobs en réponse aux perturbations avec l'objectif de minimiser le *TAMP* et *DMDFP*, en adoptant la stratégie prédictive-réactive décrite dans le logigramme de la Figure 7. Dans la sous-section qui suit, nous décrivons la génération des instances.

5.1. Génération des instances

Le Tableau 12 présente les valeurs des paramètres utilisés pour le problème $1|r_j|\ \alpha\ \Sigma w_jW_j+(1-\alpha)\ \Sigma w_jD_j$

Tableau 12. Valeurs des paramètres utilisés pour $1|r_j|\ \alpha\ \Sigma w_jW_j+(1-\alpha)\ \Sigma w_jD_j$

Paramètres	Valeurs
w_j	~ $U(1, 5)$
p_j	~ $U(1, 4)$ (*ut*)
$\theta(t)$	~ $B(p_\theta)$
T	48 (*ut*)

Les valeurs qui ont été choisies peuvent être adaptées à des cas réels, que ce soit dans le milieu hospitalier ou dans le milieu industriel. En effet, si on suppose que $\Delta t = 1$ unité de temps (*ut*), et 1 *ut* est équivalente à 10 minutes, i.e. chaque 10 minutes un évènement peut se produire (l'arrivée d'une nouvelle commande ou d'un patient urgent), alors la simulation correspond à un horizon de $T = 48$ *ut* = 480 *min* = 8*h*, représentant le temps d'ouverture d'une entreprise ou d'un bloc opératoire dans un hôpital. Les valeurs des poids peuvent représenter les 5 niveaux d'urgence dans un hôpital (Abedini *et al*. 2016) ou 5 niveaux de priorité des clients dans une usine. Les durées des jobs représentent les durées des opérations chirurgicales ou le temps de fabrication des produits. Dans notre cas, elles suivent une loi de distribution uniforme discrète avec un paramètre qui varie entre 1 et 4, afin d'obtenir des temps d'opération ou de fabrication entre 10 *minutes* (1 *ut*) et 40 *minutes* (4 *ut*). Les valeurs de la variable $\theta(t)$ sont générées aléatoirement suivant une loi de Bernoulli qui fournit, à chaque période *t*, une valeur 1 avec une probabilité p_θ et 0 avec une probabilité 1- p_θ. Par définition, $p_\theta \in [0, 1]$ et représente la fréquence d'apparition d'un nouveau job à un instant *t*. Dans les tests numériques, nous avons étudié le comportement du système sous différentes fréquences d'apparition. Nous avons commencé par une petite fréquence d'apparition, valant $p_\theta = 0{,}2$ (car au-dessous de 0,2 il n'y a pas assez de perturbations), puis une fréquence d'apparition moyenne $p_\theta = 0{,}5$, et ensuite, des fréquences d'apparition plus élevées jusqu'à ce que la PLNE n'arrive plus à fournir les solutions en un temps raisonnable.

Quatre études ont été menées pour ce premier modèle :

- L'analyse de l'impact du coefficient d'efficacité-stabilité α sur la performance de l'ordonnancement ;
- L'étude de la variation du poids en fonction du temps ;
- La comparaison du modèle basé sur la PLNE avec d'autres règles de répartition, en termes de qualité de solution et de temps de calcul ;
- Et l'analyse du temps de résolution de la PLNE sur différentes tailles d'instances, afin d'en explorer les limites en termes de nombre de jobs.

5.2 Impact du coefficient d'efficacité-stabilité α sur l'ordonnancement

Dans cette étude, nous avons fait varier la valeur de α de 0,5 à 1 pour analyser son impact sur la performance de l'ordonnancement. Comme précisé précédemment, nous avons considéré que l'efficacité de l'ordonnancement est le critère principal. Par conséquent, dans cette étude, la valeur de α n'est pas inférieure à 0,5. En effet, si α est trop petit, l'efficacité de l'ordonnancement devient négligeable et la situation n'est plus réaliste. Nous présentons d'abord une analyse de la variation de α sur l'exemple précèdent, puis sur différentes instances.

5.2.1 Variation de α pour un exemple

Après avoir résolu le problème initial de l'exemple précèdent (voir Tableau 8), nous allons faire subir à cet ordonnancement des perturbations avec une fréquence d'apparition $p_\theta = 0{,}5$, i.e. 20 perturbations sur un horizon $T = 48$ *ut*. À chaque arrivée de job, l'ensemble des jobs qui ne sont pas commencés, sont réordonnancés, et trois valeurs seront extraites : la valeur du critère d'efficacité *TAMP*, la valeur du critère de stabilité *DMDFP*, et la valeur de la fonction objectif f_1. Nous avons également numéroté chaque ordonnancement, tel que le « $2^{ème}$ ordo » correspond à l'ordonnancement établi après la première perturbation, et ainsi de suite. Le Tableau 13 présente les résultats obtenus.

Tableau 13. Variation de α pour une instance exemple

α	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
	$2^{ème}$ Ordo			$3^{ème}$ Ordo			$4^{ème}$ Ordo			$5^{ème}$ Ordo		
1	42	11	42	49	17	49	63	29	63	84	47	84
0,9	42	6	38,4	49	12	45,3	63	24	59,1	84	42	79,8
0,8	42	6	34,8	49	12	41,6	63	24	55,2	84	42	75,6
0,7	42	6	31,2	50	9	37,7	66	15	50,7	90	24	70,2
0,6	42	6	27,6	51	7	33,4	69	9	45	96	12	62,4
0,5	42	6	24	52	6	29	69	9	39	99	9	54
	$6^{ème}$ Ordo			$7^{ème}$ Ordo			$8^{ème}$ Ordo			$9^{ème}$ Ordo		
1	95	49	95	102	56	102	119	64	119	143	76	143
0,9	95	42	89,7	102	49	96,7	119	57	112,8	143	69	135,6
0,8	95	42	84,4	102	49	91,4	119	57	106,6	143	69	128,2
0,7	101	24	77,9	113	31	88,4	127	45	102,4	151	57	122,8
0,6	107	12	69	120	20	80	142	24	94,8	172	30	115,2
0,5	110	9	59,5	123	17	70	139	24	81,5	169	30	99,5
	$10^{ème}$ Ordo			$11^{ème}$ Ordo			$12^{ème}$ Ordo			$13^{ème}$ Ordo		
1	153	84	153	163	84	163	170	89	170	175	94	175
0,9	153	77	145,4	163	77	154,4	170	82	161,2	175	87	166,2
0,8	153	77	137,8	163	77	145,8	170	82	152,4	175	87	157,4
0,7	161	65	132,2	171	65	139,2	178	70	145,6	183	75	150,6
0,6	188	32	125,6	198	32	131,6	209	35	139,4	218	38	146
0,5	185	32	108,5	195	32	113,5	206	35	120,5	215	38	126,5
	$14^{ème}$ Ordo			$15^{ème}$ Ordo			$16^{ème}$ Ordo			$17^{ème}$ Ordo		
1	183	94	183	187	98	187	191	101	191	197	107	197
0,9	183	87	173,4	187	91	177,4	191	94	181,3	197	100	187,3
0,8	183	87	163,8	187	91	167,8	191	94	171,6	197	100	177,6
0,7	191	75	156,2	195	79	160,2	199	82	163,9	205	88	169,9
0,6	226	38	150,8	230	42	154,8	234	45	158,4	240	51	164,4
0,5	223	38	130,5	230	39	134,5	237	39	138	249	42	145,5
	$18^{ème}$ Ordo			$19^{ème}$ Ordo			$20^{ème}$ Ordo			$21^{ème}$ Ordo		
1	206	116	206	210	119	210	216	125	216	226	144	226
0,9	206	109	196,3	210	112	200,2	216	118	206,2	226	128	216,2
0,8	206	109	186,6	210	112	190,4	216	118	196,4	226	128	206,4
0,7	217	88	178,3	221	92	182,3	231	97	190,8	238	125	204,1
0,6	252	51	171,6	256	55	175,6	266	60	183,6	270	94	199,6
0,5	261	42	151,5	268	42	155	284	48	166	288	82	185

(1) : *TAMP* (2) : *DMDFP* (3) : f_1

Lorsque α = 1, la fonction objectif considère seulement le *TAMP* comme critère. Dans ce cas, lorsqu'un job arrive, il est ordonnancé sans prendre en compte la stabilité. Généralement, si les dates de disponibilité sont égales, la règle *wSPT* est optimale pour le problème dans ce cas.

Quand α diminue, la stabilité de l'ordonnancement est prise en compte. Une diminution du critère de stabilité *DMDFP* est observée, ainsi qu'une augmentation du critère d'efficacité *TAMP*. Cette augmentation est plus grande dans les dernières itérations, quand le nombre des jobs est plus important. La Figure 10 décrit ce comportement.

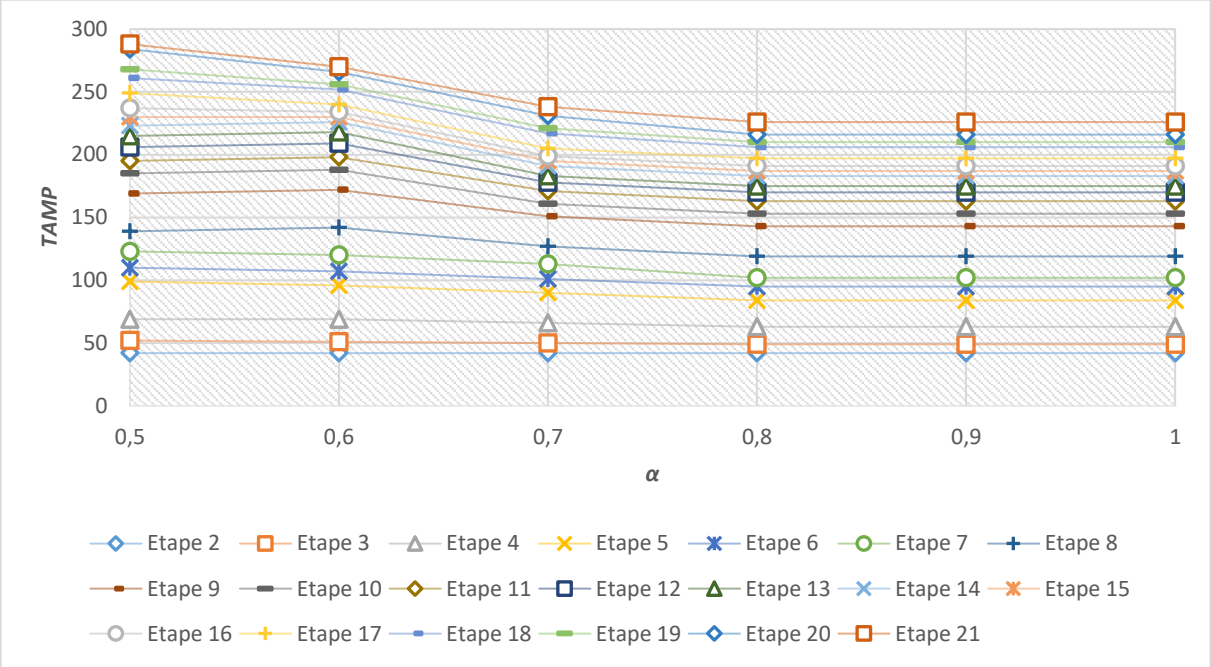

Figure 10. *TAMP* en fonction de *α*

Plus le nombre de jobs est important, plus la valeur du *TAMP* augmente, particulièrement quand *α* diminue. L'augmentation de la valeur du *TAMP* signifie qu'il y a des jobs qui attendent longtemps dans le système avant d'être exécutés. Pour illustrer cette situation, nous présentons dans le Tableau 14, la disposition des 5 premières itérations. Les jobs en gras sont les nouveaux jobs qui arrivent.

Tableau 14. Disposition des 5 premières itérations

α	Ordo initial	(1)	2ème Ordo	(1)	(2)	3ème Ordo	(1)	(2)
1	C-A-E-D-B	31	C-**F**-A-E-D-B	42	11	C-F-A-**G**-E-D-B	49	17
0,9			C-A-**F**-E-D-B	42	6	C-A-F-**G**-E-D-B	49	12
0,8			C-A-**F**-E-D-B	42	6	C-A-F-**G**-E-D-B	49	12
0,7			C-A-**F**-E-D-B	42	6	C-A-F-E-**G**-D-B	50	9
0,6			C-A-**F**-E-D-B	42	6	C-A-F-E-D-**G**-B	51	7
0,5			C-A-**F**-E-D-B	42	6	C-A-F-E-D-B-**G**	52	6

α	4ème Ordo	(1)	(2)	5ème Ordo	(1)	(2)
1	C-F-A-G-**H**-E-D-B	63	29	C-F-A-G-H-**I**-E-D-B	84	47
0,9	C-A-F-G-**H**-E-D-B	63	24	C-A-F-G-H-**I**-E-D-B	84	42
0,8	C-A-F-G-**H**-E-D-B	63	24	C-A-F-G-H-**I**-E-D-B	84	42
0,7	C-A-F-E-G-**H**-D-B	66	15	C-A-F-E-G-H-**I**-D-B	90	24
0,6	C-A-F-E-D-G-**H**-B	69	9	C-A-F-E-D-G-H-**I**-B	96	12
0,5	C-A-F-E-D-G-**H**-B	69	9	C-A-F-E-D-G-H-B-**I**	99	9

(1) : *TAMP* (2) : *DMDFP*

L'augmentation de la valeur du *TAMP* à chaque étape est due aux jobs à faible poids. Ces jobs sont repoussés vers la droite à chaque étape, car les nouveaux jobs sont inclus au sein de la séquence, et particulièrement au début de la séquence quand ils ont un poids élevé.

Cependant, lorsque α diminue, la stabilité de l'ordonnancement est beaucoup plus considérée. Ainsi, les jobs existants font moins de mouvement pour ne pas trop dévier la séquence précédente. Les nouveaux jobs sont alors placés en dernier dans la séquence. Cette situation impacte l'efficacité de l'ordonnancement. D'où, l'augmentation de la valeur du *TAMP* dans ce cas.

Remarque : Les jobs qui ont un poids élevé sont toujours priorisés. En présence de ces jobs, les jobs qui ont un poids faible sont reportés même s'ils arrivent tôt, comme le job B dans l'exemple précédent. Le job B, bien qu'il soit présent dans l'ordonnancement initial et que sa date de disponibilité soit $r_B = 1$, est toujours placé en dernier dans la séquence car son poids est faible, $w_B = 1$. Par conséquent, les jobs qui ont un poids faible sont reportés à chaque étape de réordonnancement et il y a un grand risque que ces jobs ne soient jamais exécutés.

Pour faire face à ce problème, un concept innovant a été conçu. Il consiste à augmenter le poids des jobs en fonction du temps. C'est l'objectif de la section 5.3.

5.2.2 Variation de α pour différentes instances

Pour analyser le comportement des critères en fonction du nombre de jobs, nous étudions dans ce paragraphe la variation des valeurs finales du *TAMP*, de la *DMDFP*, et de f_1 pour différentes valeurs de α et p_θ. 10 instances pour chaque type de problème ont été générées, soit ce qui donne pour 6 valeurs de α et 3 valeurs de p_θ, soit 10x6x3 = 180 problèmes différents. Ces instances sont générées selon la description de la section 5.1. Chaque instance commence par 5 jobs initiaux. Ensuite, ces jobs sont perturbés par l'arrivée de nouveaux jobs avec des fréquences d'apparition p_θ respectivement égales à 0,2, 0,5 et 0,6. Les moyennes des résultats sont présentées dans le Tableau 15.

Tableau 15. Moyennes de *TAMP*, *DMDFP* et f_1 pour différentes valeurs de α et p_θ

	$p_\theta = 0{,}2$			$p_\theta = 0{,}5$			$p_\theta = 0{,}6$		
α	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
0,5	60,9	9,4	35,15	240,4	61,7	151,05	305,3	79,4	192,35
0,6	59,3	11	39,98	228,2	72,8	166,04	283,6	93,4	207,52
0,7	59,2	12,4	45,16	218,5	91	180,25	277	115,6	228,58
0,8	58,5	14,5	49,7	207,8	106,9	187,62	265,5	137,1	239,82
0,9	58,4	15,3	54,09	206,8	108,2	196,94	265,3	137,9	252,56
1	58,4	16,8	58,4	205,5	119,5	205,5	261,2	154,6	261,2

(1) : Moyenne *TAMP* (2) : Moyenne *DMDFP* (3) : Moyenne f_1

L'augmentation de α aide l'efficacité de l'ordonnancement puisque la valeur moyenne du *TAMP* diminue. En contrepartie, la valeur moyenne de la *DMDFP* augmente avec α puisque la stabilité est moins considérée.

D'autre part, les valeurs moyennes de ces deux critères augmentent lorsque p_θ augmente. Quand le nombre de jobs devient plus important, la probabilité d'apparition de jobs avec un poids fort augmente. Dans ce cas, il y aura une forte chance que les jobs à faible poids soient

reportés. Il est donc judicieux d'augmenter les poids des jobs en fonction du temps pour que ces jobs ne soient pas ignorés en présence des nouveaux jobs à fort poids. D'autres tests ont été établis pour des valeurs de p_θ plus grandes, mais le PLNE n'arrive pas à résoudre le problème en un temps raisonnable.

5.3 Variation du poids en fonction du temps

Comme mentionné précédemment, les jobs qui ont un poids faible sont toujours placés en dernier dans la séquence et reportés après chaque réordonnancement. Nous en avons conclu qu'il serait intéressant de considérer des poids dynamiques. Dans cette section, nous considérons donc des poids qui varient en fonction du temps. La nouvelle formule est donnée comme suit : $w_{jt} = w_j(t - r_j + 1)^\rho$. Le poids sera alors une fonction du temps actuel *t*, de la date de disponibilité r_j, et de l'indice ρ qui permet au décideur de réguler l'effet de l'augmentation du poids en fonction du temps.

La valeur de (t - r_j + 1) représente la différence entre la date actuelle t et la date de disponibilité du job r_j. Plus un job reste longtemps dans le système, plus cette différence sera grande. Afin de ne pas avoir un poids nul quand $t = r_j$, nous avons ajouté 1 à cette quantité. ρ permet de réguler l'effet de la variation du poids. Il est supposé que la valeur de ρ soit entre 0 et 1 pour obtenir des augmentations de poids relativement petites. Lorsque $\rho = 0$, cela signifie que les poids sont statiques, et qu'il n'y a pas de variation de poids en fonction du temps. En revanche, lorsque $\rho = 1$, les valeurs des poids augmentent trop. Si la valeur de ρ dépasse 1, l'efficacité du système sera impactée. Donc, le domaine de variation de ρ sera dans l'intervalle [0,1]. Cette nouvelle formule du poids en fonction du temps est intégrée dans la fonction objectif, telle que :

$$f_1 = \alpha \sum_{j \in N} w_{jt} W_j + (1 - \alpha) \sum_{j \in No} w_{jt} D_j$$

Cette nouvelle formulation du poids aide les jobs à faible poids et qui ont une date de disponibilité précoce, à être exécutés, en augmentant leurs poids en fonction du temps. Pour analyser l'impact de la variation du poids en fonction du temps, nous avons étudié deux critères qui dépendent du temps d'écoulement F_j. Le temps d'écoulement F_j d'un job j mesure la durée entre sa date de disponibilité r_j et sa date de fin C_j, donnée par $F_j = C_j - r_j$. Le premier critère est le Temps d'Ecoulement Moyen *TEM*, $TEM = 1/n \sum_{j=1}^{n} (C_j - r_j)$. Le deuxième critère est l'Ecart Type des Temps d'Ecoulement *ETTE*. Il mesure la déviation entre le temps d'écoulement du job et le *TEM*. Le Tableau 16 présente les valeurs de ces deux critères en fonction de ρ pour l'exemple présenté dans la section 5.2.1.

Tableau 16. Valeurs de *TEM* et *ETTE* en fonction ρ de et α.

	α = 0.5		α = 0.6		α = 0.7		α = 0.8		α = 0.9		α = 1	
ρ	*TEM*	*ETTE*	*TEM*	*ETTE*	*TEM*	*ETTE*	*TEM*	*ETTE*	*TEM*	*ETTE*	*TEM*	*ETTE*
0	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,9	6,59
0,1	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5
0,2	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5
0,3	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5	7,8	6,5
0,4	7,8	5,87	7,8	5,87	7,8	5,87	7,8	5,87	7,8	5,87	7,8	5,87
0,5	8	5,21	8	5,21	8	5,21	8,1	5,5	8,1	5,5	8,1	5,5
0,6	8	5,21	8	5,21	8	5,21	8	5,21	8,1	5,5	8,1	5,5
0,7	8	5,21	8	5,21	8	5,21	8	5,21	8	5,21	8,1	5,5
0,8	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09
0,9	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09
1	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09	8,2	5,09

TEM : Temps d'Ecoulement Moyen, *ETTE* : Ecart Type des Temps d'Ecoulement

Comme présenté dans le Tableau 16, la valeur du *TEM* augmente avec ρ. En revanche, la valeur de l'*ETTE* diminue lorsque ρ augmente. Cela signifie que la dispersion autour du *TEM* diminue. En d'autres termes, les valeurs du temps d'écoulement F_j se rapprochent les unes des autres. Ainsi, nous pouvons conclure que l'augmentation de ρ aide les jobs de faible poids à être exécutés.

D'autre part, l'augmentation de α rend difficile la diminution de l'*ETTE*. Par exemple, la valeur de l'*ETTE* quand ρ = 0,6 est égale à 5,5 avec α = 1, au lieu de 5,21 avec α = 0,5. Quand α diminue, la stabilité de l'ordonnancement est plus prise en considération. Systématiquement, les nouveaux jobs sont placés en dernier dans la séquence pour ne pas trop perturber l'ordonnancement précédent. Par conséquent, les jobs existants sont exécutés même s'ils ont un poids faible. En revanche, lorsque la valeur de α augmente, l'efficacité de l'ordonnancement est plus considérée et l'ordonnancement déjà établi est beaucoup plus perturbé. Dans ce cas, de grandes valeurs de ρ sont nécessaires pour aider les jobs à faible poids. En conclusion, si α augmente, l'indice des poids ρ doit augmenter pour aider les jobs qui ont un poids faible à être exécutés.

Pour illustrer la différence entre le cas des poids statiques et dynamiques, nous présentons dans le Tableau 17, la disposition des 5 premières itérations dans le cas de ρ = 0 et ρ = 1, avec α = 0,8. Les jobs en gras sont les nouveaux jobs qui arrivent.

Tableau 17. Disposition des 5 premières itérations avec ρ = 0 et ρ = 1

ρ=0	Ordo initial	(1)		2ème Ordo	(1)	(2)	3ème Ordo	(1)	(2)
	C-A-E-D-B	31		C-A-**F**-E-D-B	42	6	C-A-F-**G**-E-D-B	49	12
	4ème Ordo	(1)	(2)	5ème Ordo	(1)	(2)			
	C-A-F-G-**H**-E-D-B	63	24	C-A-F-G-H-**I**-E-D-B	84	42			
ρ=1	Ordo initial	(1)		2ème Ordo	(1)	(2)	3ème Ordo	(1)	(2)
	C-A-E-D-B	31		C-A-**F**-E-D-B	42	6	C-A-F -E-D-B-**G**	52	6
	4ème Ordo	(1)	(2)	5ème Ordo	(1)	(2)			

C-A-F -E-D-B-G-**H**	72	6	C-A-F -E-D-B-G-H-**I**	105	6

(1) : *TAMP* (2) : *DMDFP*

L'augmentation de ρ a aidé le job B à ne pas être placé en dernier dans les séquences, et donc à être exécuté. Dans le 3ème Ordo par exemple, le job G s'est placé derrière le job B lorsque ρ = 1, contrairement au cas lorsque ρ = 0, où le job G s'est inséré après le job F. Cette nouvelle disposition montre bien que les jobs à poids faible et qui ont des dates de disponibilité précoces ne seront pas ignorés. Cependant, cette variation de poids a un impact sur les valeurs des critères.

Pour analyser l'impact de ρ sur la valeur des objectifs, nous avons calculé, pour chaque valeur de ρ, les valeurs finales de *TAMP*, *DMDFP* et f_1. Pour construire l'ordonnancement, nous avons utilisé des poids dynamiques w_{jt}, mais les valeurs des objectifs sont calculées avec des poids statiques w_j. Nous avons aussi calculé le pourcentage de la déviation de f_1 par rapport au cas des poids statiques. L'étude a été réalisée sur l'exemple précèdent pour assurer la comparabilité des résultats. Différentes valeurs de p_θ sont étudiées et il est supposé que α = 0.8. Les résultats sont présentés dans le Tableau 18.

Tableau 18. Variation de *TAMP*, *DMDFP*, et f_1 en fonction de ρ pour différentes valeurs de p_θ

	p_θ = 0,2				p_θ = 0,5				p_θ = 0,6			
ρ	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
0	62	8	51,2	0%	226	128	206,4	0%	349	210	321,2	0%
0,1	62	8	51,2	0%	268	104	235,2	14%	377	185	338,6	5%
0,2	62	8	51,2	0%	277	87	239	16%	416	151	363	13%
0,3	62	8	51,2	0%	297	59	249,4	21%	477	110	403,6	26%
0,4	62	8	51,2	0%	362	53	330,2	60%	517	84	430,4	34%
0,5	62	8	51,2	0%	393	43	323	56%	625	72	514,4	60%
0,6	68	6	55,6	9%	443	39	362,2	75%	727	52	592	84%
0,7	68	6	55,6	9%	493	23	399	93%	809	37	654,6	104%
0,8	68	6	55,6	9%	513	17	413,8	100%	848	25	683,4	113%
0,9	68	6	55,6	9%	513	17	413,8	100%	848	25	683,4	113%
1	68	6	55,6	9%	516	16	416	102%	848	25	683,4	113%

(1) : *TAMP* (2) : *DMDFP* (3) : f_1 (4) : pourcentage de la déviation de f_1

L'augmentation de ρ aide la stabilité du système, puisque la valeur de *DMDFP* diminue. En d'autres termes, les jobs qui arrivent tôt doivent garder les mêmes positions pour être exécutés. En revanche, la valeur du *TAMP* augmente avec ρ puisque la stabilité est plus importante dans ce cas. Une augmentation de la valeur de la fonction objectif f_1 est aussi observée, plus précisément lorsque p_θ = 0,6. Dans ce cas, le manager doit prendre une décision sur le fait d'utiliser ou pas des poids qui varient en fonction du temps, ainsi que sur la valeur de l'indice du poids ρ. D'autres tests ont été effectués pour des plus grandes valeurs de p_θ, mais la PLNE n'arrive pas toujours à résoudre le problème en un temps raisonnable.

5.4 Analyse comparative

Comme mentionné dans le premier chapitre, d'autres chercheurs ont opté pour des règles de répartition et des heuristiques pour implémenter leurs approches réactives, afin de réordonnancer les jobs en réponse aux perturbations. Certaines règles de répartition et heuristiques fréquemment utilisées sont mentionnées dans (Nie *et al.*, 2013). Parmi elles, nous avons choisi FIFO (First In First Out) pour la comparer avec la méthode basée sur la PLNE,

puisque FIFO est couramment utilisée par les entreprises lorsqu'aucune stratégie particulière de réordonnancement n'est adoptée (Rahmani and Ramezanian, 2016). Il est aussi intéressant de comparer la méthode basée sur la PLNE avec la règle de répartition *wSPT*, car cette règle considère les poids des jobs. Nous avons alors adapté les deux règles de répartition, FIFO et *wSPT* pour les faire correspondre à notre problème. Les résultats fournis par ces méthodes sont ensuite comparés avec la méthode basée sur la PLNE. 10 instances différentes ont été testées pour 3 valeurs de α différentes (0,5, 0,75, 1) et 3 valeurs de p_θ différentes (0,2, 0,6, 0,7), soit 10x3x3 = 90 problèmes différents. Le Tableau 19 présente les moyennes des valeurs obtenues.

Tableau 19. Comparaison des résultats pour des méthodes de résolution différentes

		PLNE				*wSPT*				FIFO			
p_θ(*NTJA*)	α	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
0,2 (10 jobs)	0,5	60,9	9,4	**35,15**	0,59	67,4	22,7	48,1	0,08	104,8	0	52,4	0,06
	0,75	59,2	12,4	**47,5**	0,62	67,4	22,7	56,225	0,08	104,8	0	78,6	0,06
	1	58,4	16,8	**58,4**	0,76	67,4	22,7	61,9078	0,08	104,8	0	104,8	0,06
0,6 (28 jobs)	0,5	305,3	79,4	**192,35**	247,8	286,7	162,9	224,8	0,08	596	0	298	0,07
	0,75	277,3	121,7	**238,5**	242,1	286,7	162,9	255,75	0,09	596	0	447	0,08
	1	261,3	154,6	**261,2**	235,6	286,7	162,9	286,7	0,1	596	0	596	0,08
0,7 (33 jobs)	0,5	770,42	212	**491,21**	1746,98	758	479,2	618,6	0,28	1523,8	0	761,9	0,11
	0,75	665,28	379,71	**593,89**	9530,24	758	479,2	688,3	0,28	1523,8	0	1110,6	0,12
	1	646,42	445,85	**646,42**	1606,10	758	479,2	758	0,29	1523,8	0	1523,8	0,10

(1) : *TAMP* (2) : *DMDFP* (3) : f_1 (4) : Temps de calcul (en secondes) *NTJA* : Nombre Total de Jobs Arrivés

La méthode proposée, basée sur la PLNE, fournit toujours des meilleurs résultats, en gras dans le Tableau 19, comparés à *wSPT* et FIFO. *wSPT*, qui consiste à séquencer les jobs par ordre de p_j/w_j croissant, rate quelques solutions optimales, car la présence des dates de disponibilité des jobs rend cette heuristique non optimale pour ce problème. D'autre part, elle ne tient pas compte de la stabilité de l'ordonnancement. De plus, comme attendu, FIFO qui consiste à placer, à chaque étape de réordonnancement, le nouveau job dans la dernière position, fournit des mauvaises solutions. Bien que l'heuristique FIFO fournisse évidement des meilleures solutions au niveau de la stabilité, celle-ci reste très faible au niveau de l'efficacité car elle ne considère pas les poids des jobs pour construire les ordonnancements. En effet, lorsque α augmente, l'efficacité de l'ordonnancement devient plus importante. Dans ce cas, la solution fournie par FIFO s'éloigne trop de la solution fournie par la PLNE. Cet écart devient encore plus large lorsque le nombre de jobs est important. La Figure 11 décrit ce comportement.

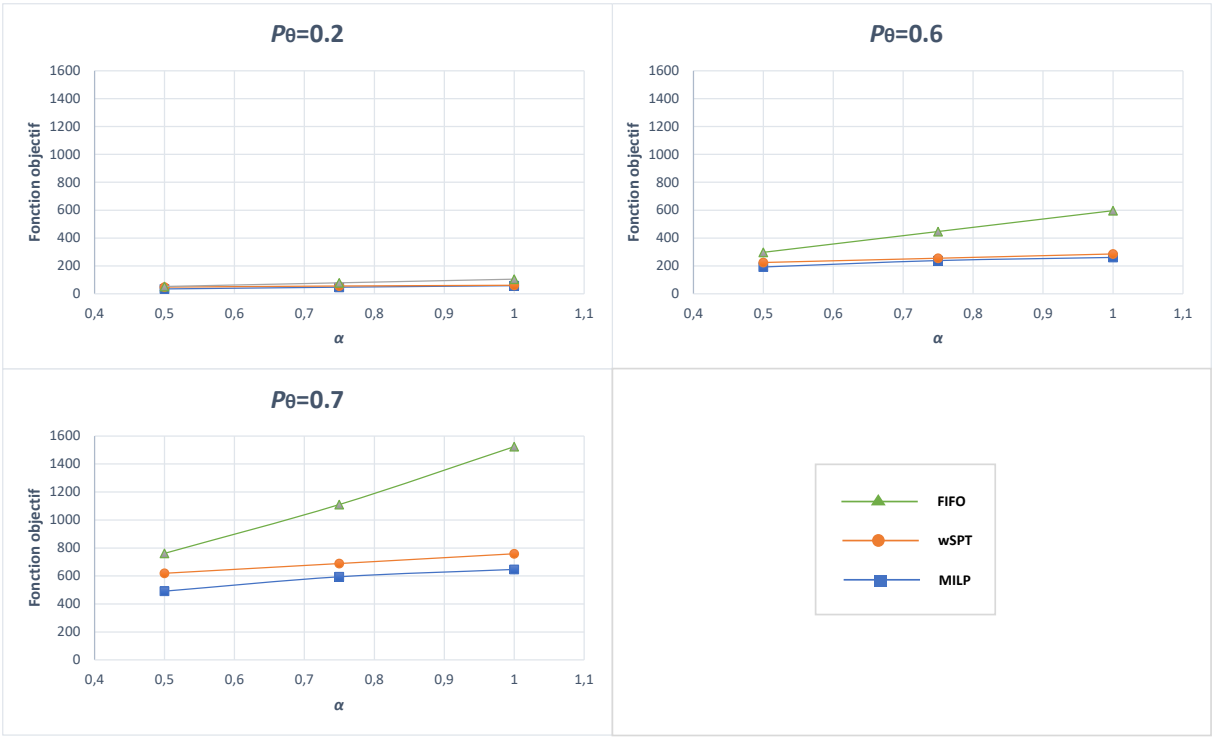

Figure 11. Variation de la fonction objectif pour des différentes méthodes de résolution

En termes de temps de résolution, les deux heuristiques *wSPT* et FIFO sont très rapides comparées à la PLNE. Le temps d'exécution de celle-ci devient excessif lorsque $p_\theta = 0{,}7$. Dans ce qui suit, une étude approfondie du temps de résolution a été menée.

5.5 Etude du temps de résolution

Dans cette section, nous avons analysé la Durée Maximale d'une Itération (*DMI*) pour explorer la limitation du modèle PLNE en termes de nombre de jobs. La *DMI* calcule le temps nécessaire entre l'apparition d'un job et l'établissement de l'ordonnancement. Les calculs ont été établis sur FICO Xpress IVE, sur un PC Core i7, 2.90 GHz, RAM 8 Go. 10 instances différentes par taille de problème ont été étudiées, 2 valeurs du nombre de jobs initiaux (5 et 7), et 4 valeurs de p_θ (0,2 ; 0,5 ; 0,6 ; 0,7), soit 10x2x4 = 80 problèmes différents. Les instances sont générées selon les paramètres de la section 5.1. Ensuite, la valeur maximale (Max), la valeur minimale (Min), la valeur moyenne (Moy), et l'écart type (ET) de la *DMI* sont calculés en secondes. Un horizon de simulation $T = 48$ *ut* et un coefficient d'efficacité-stabilité $\alpha = 0.8$ ont été considérés. Les résultats obtenus sont présentés dans le Tableau 20.

Tableau 20. Durée Maximale d'une Itération (*DMI*) avec la PLNE

Jobs initiaux	p_θ (*NTJA*)	Min *DMI* (*s*)	Max *DMI* (*s*)	Moy *DMI* (*s*)	ET *DMI* (*s*)
5	0,2 (10 jobs)	0,06	0,08	0,07	0,009
	0,5 (24 jobs)	0,38	0,42	0,4	0,01
	0,6 (28 jobs)	12,87	562,72	232,78	234,02
	0,7 (33 jobs)	56,34	3562,34	1107,67	1666,78
7	0,2 (10 jobs)	0,09	0,11	0,10	0,006
	0,5 (24 jobs)	0,56	0,82	0,63	0,10
	0,6 (28 jobs)	35,49	593,09	306,20	261,02
	0,7 (33 jobs)	-	-	-	-

NTJA : Nombre Total de Jobs Arrivés

Lorsque p_θ augmente, l'ensemble des jobs à réordonnancer à chaque itération augmente aussi, ainsi que la moyenne de la *DMI*. D'autre part, la moyenne de la *DMI* augmente aussi avec le nombre de jobs initiaux. Ainsi, la *DMI* dépend en même temps du nombre de jobs initiaux et de la fréquence d'apparition des jobs p_θ. Si ceux-ci augmentent, la moyenne et l'écart type de la *DMI* augmentent, ce qui rend difficile d'estimer le temps de calcul nécessaire pour l'exécution du modèle basé sur la PLNE.

Dans un système de réordonnancement, un planning doit être établi après chaque perturbation. Celui-ci doit se construire le plus rapidement possible, et de préférence avant l'apparition d'une autre perturbation. Dans notre étude, selon la discrétisation que nous avons établie, les perturbations peuvent arriver à chaque Δt. Ainsi, si la *DMI* dépasse Δt, elle sera considérée comme inacceptable. En se basant sur l'hypothèse que Δt est équivalente à 10 minutes (600 s), et selon le Tableau 20, la PLNE est capable de résoudre des problèmes qui contiennent 7 jobs initiaux, perturbés par l'arrivée de nouveaux jobs avec une fréquence d'apparition p_θ allant jusqu'à 0,6. Au total, c'est équivalent en moyenne à 35 jobs sur la période T. Nous avons aussi testé des problèmes avec 7 jobs initiaux et $p_\theta = 0{,}7$ (soit 40 jobs au total), mais la PLNE n'arrivait pas à résoudre le problème en un temps raisonnable. L'exécution a été interrompue au bout de 12 heures de calcul.

6. Conclusion

Dans ce chapitre, une nouvelle mesure de performance a été proposée et étudiée pour un problème de réordonnancement avec une machine unique. La mesure proposée combine simultanément l'efficacité de l'ordonnancement représentée par le *TAMP*, et sa stabilité représentée par la *DMDFP*. Ces deux critères sont associés par α, le coefficient d'efficacité-stabilité. Cette association des deux critères n'a pas été étudiée dans la littérature, et peut être très utile et significative dans les milieux industriels et hospitaliers. Un modèle basé sur la PLNE a été implémenté ainsi qu'une stratégie prédictive-réactive pour gérer les perturbations causées par l'arrivée de nouveaux jobs. Les résultats numériques ont démontré que les jobs qui ont un poids faible sont à chaque fois reportés. Par conséquent, une nouvelle conception a été proposée. Celle-ci consiste à augmenter les poids des jobs en fonction du temps. La résolution du modèle a permis d'extraire les conclusions suivantes :

- L'augmentation du temps d'attente moyen pondéré *TAMP* est due essentiellement aux jobs qui ont un poids faible. Ces jobs sont reportés à chaque étape de réordonnancement, car les nouveaux jobs qui arrivent s'insèrent au milieu des séquences.
- L'augmentation du poids des jobs en fonction du temps aide les jobs qui ont un poids faible à être exécutés. Cependant, le décideur doit choisir des valeurs de ρ plus élevées lorsque α augmente.
- En analysant la durée maximale d'une itération *DMI*, il a été constaté que le temps d'exécution dépend à la fois du nombre de jobs initiaux et de la fréquence d'apparition des jobs p_θ. Lorsque ceux-ci augmentent, la PLNE met plus de temps pour fournir les solutions.

Ce premier travail a fait l'objet d'une publication dans une revue internationale [AP3]. Cette contribution est d'un grand intérêt pour les preneurs de décision qui font face aux problèmes de

réordonnancement. Elle permet de fournir, à chaque étape de réordonnancement, une séquence efficace et stable en réponse aux perturbations dues à l'arrivée de nouveaux jobs dans un environnement de machine unique. Cependant, le modèle proposé, basé sur la PLNE, est limité en termes de nombre de jobs, car il permet de résoudre le problème jusqu'à 35 jobs. Dans le chapitre qui suit, nous étudions le comportement de ce nouveau critère sur des machines multiples, notamment dans un environnement de machines parallèles identiques.

Chapitre 3 : Optimisation de l'efficacité et de la stabilité dans un problème de réordonnancement : cas de machines parallèles

Dans ce chapitre, nous étudions un problème de minimisation du temps d'attente moyen pondéré et de la déviation moyenne des dates de fin pondérée dans un problème de réordonnancement de machines parallèles. Comme pour le deuxième chapitre, nous commençons par une description du problème, en motivant l'inspiration de notre idée. Nous fournissons ensuite une nouvelle formulation mathématique adaptée à ce problème précis. La fonction objectif considérée dans ce cas est basée sur celle décrite auparavant, qui combine le critère d'efficacité et de stabilité, à laquelle nous ajoutons une troisième mesure pour minimiser les changements de machines des jobs après le réordonnancement. Ce troisième critère a été optimisé à l'aide de la méthode d'optimisation lexicographique. Des tests numériques ont été menés pour étudier l'impact du coefficient d'efficacité-stabilité sur le système, et explorer les limites du modèle en termes de nombre de jobs.

1. Introduction

Le problème de réordonnancement sur des machines parallèles est un problème souvent rencontré dans des cas réels. Par exemple, dans les blocs opératoires identiques, où les opérations électives et urgentes doivent être traitées. Dans Wullink *et al*. (2007), les auteurs ont prouvé que l'approche qui consiste à partager les salles opératoires entre les opérations électives et urgentes est bien meilleure que celle qui consiste à dédier une salle ou plusieurs aux opérations urgentes. Par conséquent, quand il s'agit de gérer les opérations électives et urgentes, le mieux est de modéliser le système comme un système de réordonnancement sur des machines parallèles identiques. Le problème de réordonnancement sur des machines parallèles dans les ateliers de production est aussi une illustration de l'utilité du travail présenté ici.

Ce chapitre s'intéresse à un problème de réordonnancement sur des machines parallèles identiques, qui subissent des perturbations dues à l'arrivée de nouveaux jobs. Comme dans le chapitre précédent, la stratégie prédictive-réactive a été adoptée. Dans la phase prédictive, elle résout le problème d'ordonnancement statique, ayant comme objectif de minimiser le critère d'efficacité seul représenté par le temps d'attente moyen pondéré. À chaque arrivée d'une perturbation, la phase réactive réordonnance les jobs, avec cette fois comme objectif de minimiser les deux critères simultanément, le critère d'efficacité de l'ordonnancement, et le critère de stabilité représenté par la déviation moyenne des dates de fin pondérées par les poids des jobs. Ces deux critères sont associés dans la phase réactive par α, le coefficient d'efficacité-stabilité.

Selon Alagöz and Azizoğlu (2003), le changement d'affectation des jobs aux machines génère aussi des coûts supplémentaires, notamment des coûts de réinstallation d'outils dans le cas des applications industrielles ou des coûts de préparation des salles d'opération dans les milieux hospitaliers. Ainsi, dans la phase réactive, un troisième critère de stabilité est aussi optimisé, qui consiste à minimiser le nombre de jobs qui changent de machine après le réordonnancement. Ce troisième critère est optimisé en utilisant l'optimisation lexicographique. Cette nouvelle conception de problème de réordonnancement n'a pas été traitée avant dans la littérature. Elle contribue alors au domaine de la recherche opérationnelle par :

- L'étude du *TAMP* combiné avec la *DMDFP* dans un problème de réordonnancement sur des machines parallèles identiques.
- L'implantation d'un modèle basé sur la PLNE après l'avoir amélioré pour qu'il soit adapté à ce type de problème, et le développement d'une stratégie prédictive-réactive qui gère les perturbations dues à l'arrivée de nouveaux jobs.
- La mise en œuvre de la méthode d'optimisation lexicographique pour minimiser le nombre de jobs qui changent de machine après le réordonnancement, et l'analyse de cette méthode sur la performance des solutions obtenues.

2. Description du problème

Nous étudions un problème de réordonnancement sur des machines parallèles identiques qui subissent des perturbations causées par l'arrivée de nouveaux jobs. Ce problème est résolu en deux phases. La phase prédictive, qui commence avant l'apparition d'une perturbation, consiste à minimiser le *TAMP*. La phase réactive, qui commence après l'apparition d'une perturbation,

consiste à minimiser simultanément le *TAMP* ainsi que la *DMDFP*. A chaque job *j*, est associé un poids w_j, une durée p_j et une date de disponibilité r_j.

Phase prédictive

Comme décrit dans le chapitre 2, dans cette première phase du problème, toutes les informations sont connues à l'avance et l'objectif est de déterminer la séquence qui va minimiser le critère d'efficacité *TAMP*, i.e. $min\ \sum_{j\in N} w_j W_j$. Mais, comme nous sommes dans le cas des machines parallèles identiques, ce problème peut être représenté dans la notation standard des problèmes d'ordonnancement par $P|r_j|\sum w_j W_j$.

Phase réactive

La phase réactive débute lorsqu'un nouveau job *j'* arrive à l'instant *t* et que l'ordonnancement établi doit être mis à jour pour répondre à cette nouvelle perturbation. Il s'agit donc de réordonnancer les jobs de *No*, c'est-à-dire les jobs qui n'ont pas commencé leur exécution sur les machines à l'instant *t*, et le nouveau job *j'*. L'ensemble des jobs *N* sera alors mis à jour tel que $N \leftarrow No \cup \{j'\}$. Sur la Figure 12, le job E arrive à l'instant *t*. Les jobs A et B, qui ont commencé leur exécution avant l'instant *t*, seront supprimés de la liste. Les jobs C et D constituent alors l'ensemble *No*, et vont être combinés avec le job E pour être réordonnancés.

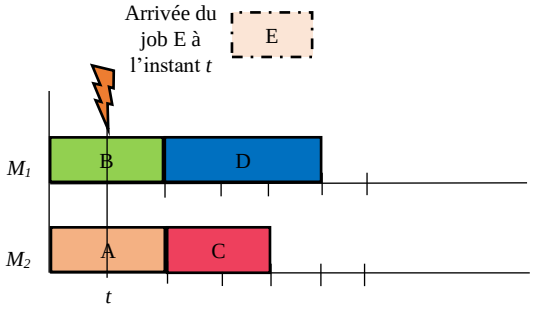

Figure 12. Arrivée d'une perturbation à l'instant t

Le premier critère de stabilité : Le premier critère de stabilité pris en considération est toujours le *DMDFP*, i.e $\sum_{j\in No} w_j D_j$, où $D_j = \max\{0, C_j - Co_j\}$ représente la déviation du job *j*, mesurée entre sa date de fin lorsqu'il est planifié pour la première fois et sa date de fin après le réordonnancement.

Dans la phase réactive, après l'arrivée d'un nouveau job, l'objectif est donc de minimiser la fonction suivante :

$$f_1 = \alpha \sum_{j\in N} w_j W_j + (1-\alpha) \sum_{j\in No} w_j D_j$$

Le deuxième critère de stabilité : Contrairement au cas à une machine, un second critère de stabilité a été considéré. Il consiste à minimiser le nombre de jobs qui changent de machine après le réordonnancement. Pour cela, on a défini Dif_j une variable binaire, telle que :

$$Dif_j = \begin{cases} 1, & Si\ le\ job\ j\ change\ de\ machine\ après\ le\ réordonnancement \\ 0, & Si\ non \end{cases}$$

Il s'agit donc de minimiser le nombre de jobs qui changent de machine après le réordonnancement, i.e. $min \sum_{j \in No} Dif_j$.

Cependant, ce critère n'est minimisé qu'après l'optimisation de la première fonction objectif f_1 en utilisant l'optimisation lexicographique. Un paragraphe est dédié à l'explication de cette approche dans la formulation mathématique.

Ce problème peut être représenté dans la notation standard des problèmes d'ordonnancement par $P|r_j|\alpha \sum w_j W_j + (1-\alpha) \sum w_j D_j , \sum Dif_j$.

3. Stratégie prédictive-réactive pour le problème P|r_j| α $\Sigma w_j W_j$+(1-α) $\Sigma w_j D_j$, ΣDif_j

Comme décrit dans les chapitres 1 et 2, la stratégie prédictive-réactive proposée consiste, dans la phase prédictive, à résoudre un problème classique d'ordonnancement ayant comme objectif de minimiser le *TAMP*, $min \sum_{j \in N} w_j W_j$. Puis, dans sa phase réactive, elle parcourt un horizon de simulation étape par étape, itérativement. À chaque étape, le processus vérifie si un job arrive. Si c'est le cas, l'ensemble *N* des jobs est mis à jour et un nouvel ordonnancement est établi. Dans cette deuxième phase, l'objectif est de minimiser la fonction f_1 qui combine l'efficacité et la stabilité de l'ordonnancement. Après avoir minimisé f_1, nous utilisons l'optimisation lexicographique pour minimiser f_2, la fonction qui exprime le nombre de jobs qui changent de machine après le réordonnancement.

Le logigramme de la Figure 13 décrit la stratégie prédictive-réactive proposée pour résoudre ce problème.

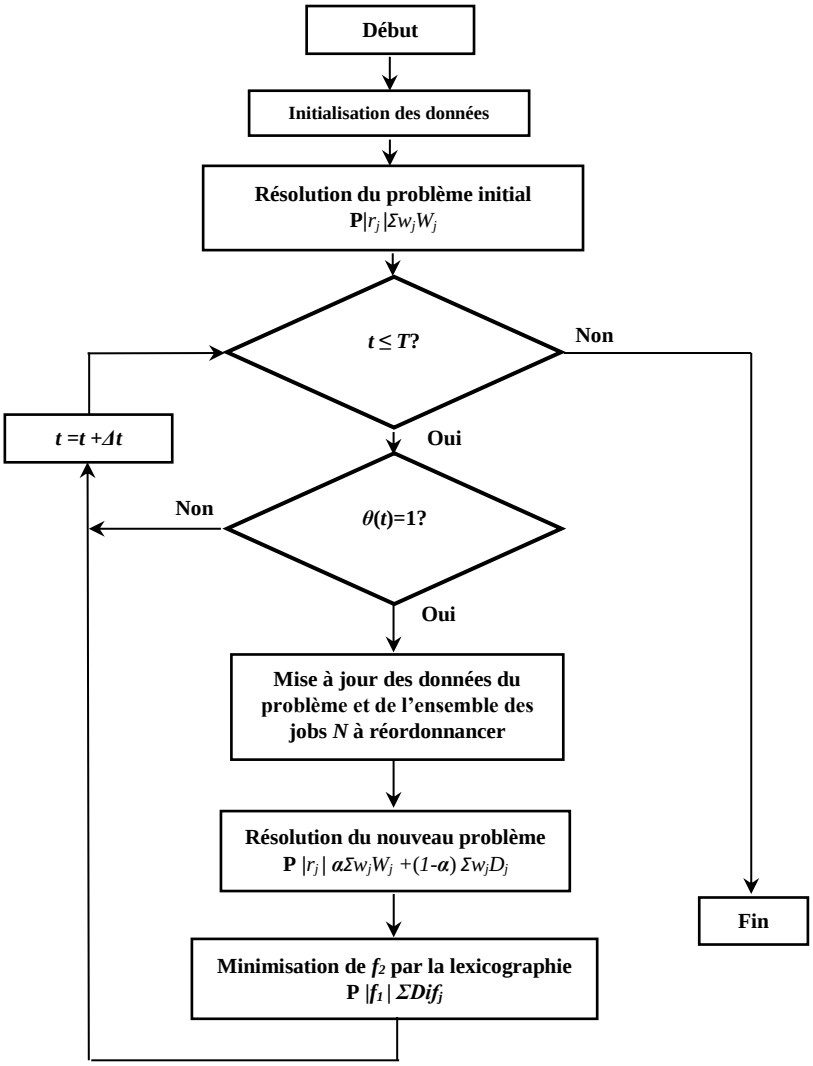

Figure 13. Stratégie prédictive-réactive pour le problème P|r_j| α $\Sigma w_j W_j$+(1-α) $\Sigma w_j D_j$, ΣDif_j

4. Modèles mathématiques

Dans cette section, deux modèles mathématiques basés sur la PLNE sont proposés. Le premier est dédié au modèle mathématique avant la perturbation, et le second au modèle mathématique après la perturbation.

4.1 Modèle mathématique avant la perturbation

Le modèle mathématique proposé est une extension du modèle présenté dans le chapitre précédent au cas à plusieurs machines parallèles identiques, qui utilise une formulation basée sur les positions, i.e. affection d'un job à une position. Cette formulation a déjà été utilisé pour résoudre les problèmes d'ordonnancement pour des machines parallèles (Abdel-Jabbar *et al*. (2014), et Beezão *et al*. (2017)), mais en général ces travaux considèrent le Makespan comme critère et ne prennent pas en considération les poids des jobs, les dates de disponibilité et le temps d'attente. Nous avons indiqué en gras, les données et variables modifiées ou nouvelles par rapport au modèle à une machine. La formulation mathématique est décrite au-dessous.

Ensembles et indices

N : ensemble de jobs {1,2, …, n}

K : ensemble de positions {1,2, …, n}

M : ensemble de machines {1,2, …, m}

j : indice des jobs, j = 1,2, …, n

k : indice des positions, k = 1,2, …, n

i : indice des machines, i = 1,2, …, m

Paramètres

w_j : poids du job j

r_j : date de disponibilité du job j

p_j : durée du job j

$bigM$: grande valeur, $bigM = \sum_{j\epsilon N} p_j + \max r_j$

Variables de décision

$$x_{jki} = \begin{cases} \mathbf{1}, & \textbf{Si le job } j \textbf{ est affecté à la machine } i \textbf{ dans la } k^{ème} \textbf{ position} \\ \mathbf{0}, & \textbf{Sinon} \end{cases}$$

W_j : temps d'attente du job j

C_j : date de fin du job j

S_{ki} : date de début du job placé en la $k^{ème}$ position sur la machine i

Cp_{ki} : date de fin du job placé en $k^{ème}$ position sur la machine i

Fonction objectif

$$min \sum_{j\in N} w_j W_j$$

Contraintes

$$\sum_{i\epsilon m} \sum_{k\epsilon N} x_{jki} = 1, \ \forall j\epsilon N \quad (2.1)$$

$$\sum_{j\epsilon N} x_{jki} \leq 1, \ \forall k\epsilon K, \ \forall i\epsilon M \quad (2.2)$$

$$\sum_{j\epsilon N} x_{j(k+1)i} \leq \sum_{j\epsilon N} x_{jki}, \ \forall k\epsilon 1..n-1, \ \forall i\epsilon M \quad (2.3)$$

$$S_{ki} \geq \sum_{j\epsilon N} r_j x_{jki}, \ \forall k\epsilon K, \ \forall i\epsilon M \quad (2.4)$$

$$S_{(k+1)i} \geq S_{ki} + \sum_{j\epsilon N} p_j x_{jki}, \ \forall k\epsilon 1..n-1, \ \forall i\epsilon M \quad (2.5)$$

$$Cp_{ki} = S_{ki} + \sum_{j\epsilon N} p_j x_{jki}, \ \forall k\epsilon K, \ \forall i\epsilon M \quad (2.6)$$

$$C_j \geq Cp_{ki} - bigM(1 - x_{jki}), \ \forall j\epsilon N, \ \forall k\epsilon K, \ \forall i\epsilon M \quad (2.7)$$

$$C_j \leq Cp_{ki} + bigM(1 - x_{jki}), \ \forall j\epsilon N, \ \forall k\epsilon K, \ \forall i\epsilon M \quad (2.8)$$

$$W_j = C_j - p_j - r_j, \ \forall j \epsilon N \quad (2.9)$$

$$S_{ki}, \ Cp_{ki} \geq 0, \ \forall k \epsilon K, \ \forall i \epsilon M \quad (2.10)$$

$$W_j, \ C_j \geq 0, \ \forall j \epsilon N \quad (2.11)$$

$$x_{jki} \in \{0,1\}, \ \forall j \epsilon N, \ \forall k \epsilon K, \ \forall i \epsilon M \quad (2.12)$$

Signification des contraintes

- Contrainte (2.1) : Elle permet de n'avoir, sur une machine, qu'une seule position par job.
- Contrainte (2.2) : Elle permet de n'avoir, sur une machine, au plus un job par position.
- Contrainte (2.3) : Elle oblige la position *k* sur une machine à être occupée si la position *k*+1 est occupée.
- Contrainte (2.4) : Elle oblige la date de début du job en $k^{ème}$ position à être supérieure ou égale à sa date de disponibilité.
- Contrainte (2.5) : Elle impose que la date de début du job en $(k+1)^{ème}$ position soit supérieure ou égale à la date de fin du job dans la position précédente ($k^{ème}$ position).
- Contrainte (2.6) : Elle impose que la date de fin du job en $k^{ème}$ position soit égale à sa date de début plus sa durée.
- Contraintes (2.7) et (2.8) : Si le job *j* est en $k^{ème}$ position, ces contraintes permettent d'imposer que la date de fin du job *j* soit égale à la date de fin du job en position *k*. On a pris $bigM = \sum_{j \in N} p_j + \max r_j$.
- Contrainte (2.9) : Elle définit le temps d'attente du job *j* en fonction sa date de fin, sa date de disponibilité et sa durée.
- Contraintes (2.10) et (2.11) : Elles sont des contraintes de non-négativité qui consistent à mettre toutes les variables supérieures ou égales à zéro.
- Contrainte (2.12) : Elle définit que les variables x_{jki} sont binaires.

4.2 Modèle mathématique après la perturbation

Ce second modèle est généré après l'apparition de chaque perturbation. Comme décrit dans le chapitre 2, les nouveaux jobs ont les mêmes paramètres que les anciens. La formulation mathématique de ce second modèle est présentée dans les sous-paragraphes suivants.

4.2.1 Optimisation de l'efficacité et la stabilité de l'ordonnancement

Comme dans le chapitre 2, après avoir mis à jour l'ensemble des jobs et défini les nouveaux paramètres, *Coj* et α, l'objectif est de minimiser f_1, tel que $f_1 = \alpha \sum_{j \in N} w_j W_j + (1-\alpha) \sum_{j \in No} w_j D_j$, soumise aux contraintes (2.1) jusqu'à (2.12), ainsi qu'aux contraintes (1.11) et (1.12) qui permettent de calculer la valeur de la variable $D_j = \max\{0, C_j - Co_j\}$.

4.2.2 Minimisation du nombre de jobs qui changent de machine

Le critère qui minimise le nombre de jobs qui changent de machine après le réordonnancement est optimisé en utilisant l'optimisation lexicographique. Celle-ci est très utilisée dans les problèmes multi-objectif. Elle fournit toujours un ensemble de solutions Pareto-optimales (Chang, 2014). Cette méthode consiste à obtenir une amélioration arbitrairement faible pour le critère le plus important à travers des petites pertes sur la performance des critères moins importants (Zykina, 2004).

L'optimisation lexicographique consiste, dans une première étape, à optimiser la première fonction objectif, *i.e.* celle qui a la priorité supérieure. Ensuite, dans une deuxième étape, à optimiser le deuxième objectif en mettant la valeur optimale de la première fonction comme contrainte, et ainsi de suite (Mavrotas, 2009). Dans notre cas, nous procédons de la même manière. Nous considérons la fonction $f_1 = \alpha \sum_{j \in N} w_j W_j + (1-\alpha) \sum_{j \in No} w_j D_j$ comme la fonction objectif principale. Après avoir obtenu la valeur optimale $f_1^* = \min f_1$, nous minimisons ensuite $f_2 = \sum_{j \in No} Dif_j$, en ajoutant la contrainte $f_1 \leq f_1^*$ afin de conserver la solution optimale de la première optimisation. $f_2 = \sum_{j \in No} Dif_j$ représente la somme des jobs qui changent de machine après la perturbation. Pour cela, nous avons défini le paramètre binaire A_{ji} qui est égal à 1, si le job *j* est affecté à la machine *i* dans la période qui précède l'apparition de la perturbation, et 0 sinon. Comme la variable binaire Dif_j est égale à 1 si l'affectation du job *j* à la machine est modifiée, et 0 sinon, nous avons ajouté les contraintes suivantes au modèle :

$$Dif_j \geq \sum_{k \in No} x_{jki} - A_{ji}, \ \forall j \in No, \ \forall i \in M \quad (2.13)$$

$$Dif_j \in \{0, 1\}, \ \forall j \in No \quad (2.14)$$

5. Expérimentations numériques

L'algorithme basé sur les PLNE a été programmé sur FICO Xpress IVE sur un PC Core i5, 2.40 GHz, RAM 4 Go. Il est composé de deux parties. La première partie consiste à résoudre le problème d'ordonnancement ayant comme objectif de minimiser le *TAMP*. La deuxième partie consiste à réordonnancer les jobs en réponse aux perturbations avec comme objectif de minimiser le *TAMP*, *DMDFP*, et le nombre de jobs qui changent de machine après le réordonnancement en adoptant la stratégie prédictive-réactive décrite dans le logigramme de la Figure 13. Dans la sous-section qui suit, nous décrivons la génération des instances.

5.1 Génération des instances

Le Tableau 21 présente les valeurs des paramètres utilisés pour le problème $1|r_j| \ \alpha \Sigma w_j W_j + (1-\alpha) \Sigma w_j D_j, \Sigma Dif_j$

Tableau 21. Valeurs des paramètres utilisés pour $1|r_j| \ \alpha \Sigma w_j W_j + (1-\alpha) \Sigma w_j D_j, \Sigma Dif_j$

Paramètres	Valeurs
w_j	~ $U(1, 5)$
p_j	~ $N(2.5, (0.5)^2)$ (*ut*) ou ~ $U(1, 10)$ (*ut*)
$\theta(t)$	~ $B(p_\theta)$
T	24 (*ut*)
m	2

Nous avons considéré un horizon de temps de 24 *ut*. Si on considère que 1 *ut* est équivalente à 15 minutes, alors la simulation correspond à un horizon de $T = 24$ *ut* = 360 *min* = 6*h*, qui peut représenter le temps d'ouverture d'une entreprise ou d'un hôpital. Comme dans le chapitre 2,

les valeurs des poids varient uniformément entre 1 et 5 et les valeurs de la variable $\theta(t)$ sont générées aléatoirement suivant une loi de Bernoulli qui fournit, à chaque période *t*, une valeur 1 avec une probabilité p_θ et 0 avec une probabilité 1-p_θ. Les tests ont été établis sur deux machines parallèles identiques.

Nous avons d'abord considéré le cas où les temps des jobs suivent une loi uniforme discrète avec un paramètre qui varie de 1 à 10. Par ailleurs, dans la littérature, d'autres travaux ont considéré une loi normale pour générer les durées des opérations chirurgicales, vue sa traçabilité et son applicabilité (Choi et Wilhelm, 2012). Plusieurs auteurs ont utilisé cette loi, comme par exemple Lei et Xie (2008) qui ont utilisé une loi normale pour générer des durées d'opérations stochastiques pour un problème d'ordonnancement dans un problème de type jobshop. Yue et Zhou (2021) ont aussi généré des durées stochastiques de jobs par une loi de distribution normale pour un problème spécifique d'ordonnancement et d'affectation à échéance des jobs dans une machine unique. Dans notre cas, nous avons généré des durées d'opérations qui suivent une loi normale avec une moyenne de 2,5 et un écart type de 0,5. Ensuite, nous avons discrétisé les valeurs obtenues pour avoir des opérations qui varient entre 15 *minutes* (1 *ut*) et 60 *minutes* (4 *ut*). Ceci permet de générer des durées entre 30 *minutes* (2 *ut*) et 45 *minutes* (3 *ut*) selon une probabilité de 68% et des durées entre 15 *minutes* (1 *ut*) et 60 *minutes* (4 *ut*) selon une probabilité de 99%.

Trois études ont été menées pour ce deuxième travail :

- L'analyse de l'impact du coefficient d'efficacité-stabilité α sur la performance de l'ordonnancement ;
- L'étude de l'optimisation lexicographique pour minimiser le nombre de jobs qui changent de machine après le réordonnancement ;
- L'analyse du temps de résolution de la PLNE sur différentes tailles d'instance pour explorer ses limites en termes de nombre de jobs.

5.2 L'impact du coefficient d'efficacité-stabilité α sur la performance de l'ordonnancement

Dans ce paragraphe, nous étudions l'impact de α, le coefficient d'efficacité-stabilité sur la performance de l'ordonnancement. D'abord, nous présentons une étude sur un exemple particulier d'instance, puis sur des instances différentes.

5.2.1 Exemple

Dans cet exemple, les durées p_j, les dates de disponibilité r_j, et les poids w_j de 5 jobs sont donnés dans le Tableau 22.

Tableau 22. Données du problème initial sur deux machines parallèles identiques

Job	*1*	*2*	*3*	*4*	*5*
r_j	0	1	0	1	2
p_j	1	2	2	3	4
w_j	5	1	4	2	1

L'objectif est de minimiser le *TAMP*. La séquence optimale est présentée sur la Figure 14. La valeur obtenue est de $\sum_{j=1}^{5} w_j W_j = 3$.

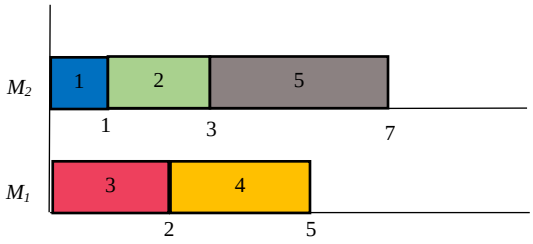

Figure 14. Séquence optimale pour le problème initial

Dans la phase réactive, à chaque période, un job arrive avec une probabilité $p_\theta = 0{,}8$. 20 jobs arrivent sur un horizon $T = 24$ *ut*. A chaque étape de réordonnancement, les jobs sont réordonnancés et quatre informations sont prélevées : la valeur du *TAMP*, la valeur de la *DMDFP*, le Nombre de Jobs qui Changent de Machine après le réordonnancement, *NJCM*, ainsi que la valeur de f_1. Nous avons également numéroté chaque ordonnancement, tel que le « $2^{ème}$ ordo » correspond à l'ordonnancement établi après la première perturbation, et ainsi de suite. Le Tableau 23 présente les résultats obtenus.

Tableau 23. Variation de α pour l'instance exemple

α	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
	$2^{ème}$ Ordo				$3^{ème}$ Ordo				$4^{ème}$ Ordo				$5^{ème}$ Ordo			
1	5	2	2	5	8	5	1	8	13	6	0	13	16	9	2	16
0,9	5	2	2	4,7	8	2	2	7,4	13	3	0	12	16	3	1	14,7
0,8	5	2	2	4,4	8	2	2	6,8	13	3	0	11	16	3	1	13,4
0,7	5	2	2	4,1	8	2	2	6,2	13	3	0	10	16	3	1	12,1
0,6	5	2	2	3,8	8	2	2	5,6	13	3	0	9	16	3	1	10,8
0,5	5	2	2	3,5	8	2	2	5	13	3	0	8	16	3	1	9,5
	$6^{ème}$ Ordo				$7^{ème}$ Ordo				$8^{ème}$ Ordo				$9^{ème}$ Ordo			
1	18	10	1	18	23	11	0	23	27	13	0	27	29	13	0	29
0,9	19	3	0	17,4	25	4	0	22,9	28	4	1	25,6	29	5	2	26,6
0,8	19	3	0	15,8	25	4	0	20,8	28	4	0	23,2	29	5	0	24,2
0,7	19	3	0	14,2	25	4	0	18,7	28	4	0	20,8	29	5	0	21,8
0,6	19	3	0	12,6	25	4	0	16,6	28	4	0	18,4	29	5	0	19,4
0,5	19	3	0	11	25	4	0	14,5	28	4	1	16	29	5	2	17
	$10^{ème}$ Ordo				$11^{ème}$ Ordo				$12^{ème}$ Ordo				$13^{ème}$ Ordo			
1	30	14	1	30	32	16	0	32	36	16	0	36	38	16	0	38
0,9	30	5	0	27,5	32	7	0	29,5	36	8	1	33,2	38	10	1	35,2
0,8	30	5	0	25	32	7	0	27	36	8	1	30,4	38	10	1	32,4
0,7	30	5	0	22,5	32	7	0	24,5	36	8	1	27,6	38	10	1	29,6
0,6	30	5	0	20	33	5	0	21,8	39	5	0	25,4	43	5	0	27,8
0,5	30	5	0	17,5	33	5	0	19	39	5	0	22	43	5	0	24
	$14^{ème}$ Ordo				$15^{ème}$ Ordo				$16^{ème}$ Ordo				$17^{ème}$ Ordo			
1	41	16	0	41	42	18	0	42	44	18	2	44	46	20	0	46
0,9	41	10	0	37,9	42	11	0	38,9	44	11	2	40,7	46	13	0	42,7
0,8	41	10	0	34,8	42	11	0	35,8	44	11	2	37,4	46	13	2	39,4
0,7	41	10	0	31,7	42	11	0	2	44	11	2	34,1	46	13	2	36,1
0,6	45	5	0	29	47	7	0	31	49	7	2	32,2	54	9	0	36
0,5	45	5	0	25	47	7	2	27	49	7	2	28	54	9	0	31,5
	$18^{ème}$ Ordo				$19^{ème}$ Ordo				$20^{ème}$ Ordo				$21^{ème}$ Ordo			
1	50	20	0	50	52	21	0	52	55	24	1	55	57	26	0	57
0,9	**49**	16	0	45,7	**50**	17	2	46,7	**52**	19	2	48,7	**53**	20	2	49,7
0,8	**49**	16	2	42,4	**50**	17	0	43,4	**52**	19	0	45,4	**53**	20	2	46,4
0,7	**49**	16	2	39,1	**50**	17	0	40,1	**52**	19	0	42,1	**53**	20	2	43,1
0,6	58	9	0	38,4	60	10	2	40	63	13	2	43	65	15	2	45
0,5	58	9	1	33,5	60	10	2	35	63	13	1	38	66	14	0	40

(1) : *TAMP* (2) : *DMDFP* (3) : *NJCM* (4) : f_1

Lorsque α = 1, la fonction objectif considère seulement le *TAMP* comme critère, ce qui explique l'augmentation de la valeur *DMDFP*.

D'autre part, lorsque α varie entre 0,9 et 0,7, la stabilité de l'ordonnancement est opportunément considérée. Dans ce cas, nous avons relevé deux observations.

- Jusqu'au 17ème Ordo, les valeurs du *TAMP* sont similaires lorsque α est entre 1 et 0,7, tandis que la valeur de la *DMDFP* est plus petite lorsque α est entre 0,7 et 0,9 comparée à α = 1, car dans ce cas, la PLNE priorise les solutions qui minimisent la *DMDFP*.
- A partir du 18ème Ordo, la valeur du *TAMP* et de la *DMDFP* est plus petite lorsque α est entre 0,7 et 0,9 comparée à α = 1. Il a été constaté que le fait de considérer la stabilité pour construire les ordonnancements précédents, a un impact positif sur l'efficacité dans les étapes suivantes. Cet effet proactif induit par le critère de stabilité est illustré dans l'exemple ci-dessous.

Et finalement, lorsque α descend au-dessous de 0,6, la valeur du *TAMP* augmente puisque la stabilité prend beaucoup plus d'importance. La Figure 15 décrit la variation du *TAMP* en fonction de α pour chaque étape.

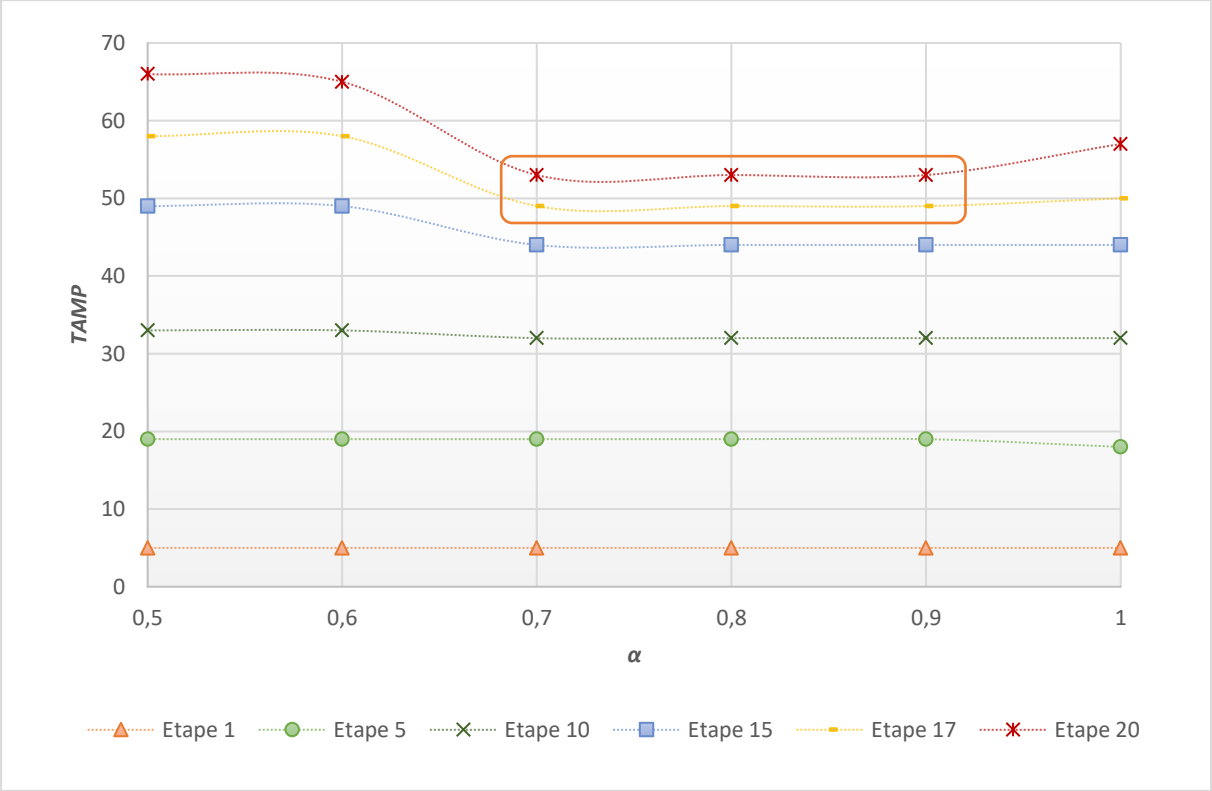

Figure 15. *TAMP* en fonction de α pour p_θ = 0,8

La Figure 15 montre que jusqu'à l'étape 10, la valeur du *TAMP* est stable quel que soit α. Entre les étapes 17 et 20, la valeur du *TAMP* est plus faible lorsque α est entre 0,9 et 0,7 comparée aux autres valeurs de α. Cela est dû à l'effet proactif induit par le critère de stabilité.

Pour illustrer clairement ce phénomène, un diagramme de Gantt est présenté Figure 16. Il présente la disposition des séquences optimales lorsque α = 0,8, comparée à celle lorsque α = 1 pour les onze premières perturbations. Les jobs fixes (ou les jobs qui ont déjà débuté sur les machines) sont coloriés en rouge et le nouveau job est colorié en vert. Les valeurs des données utilisées sont présentées dans le Tableau 24.

Tableau 24. Données du problème

Job j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
r_j	0	1	0	1	2	2	3	4	6	7	9	10	11	12	13	14
p_j	1	2	2	3	4	3	4	1	1	4	1	2	1	3	2	4
w_j	5	1	3	2	1	1	1	5	3	1	5	3	4	1	1	1

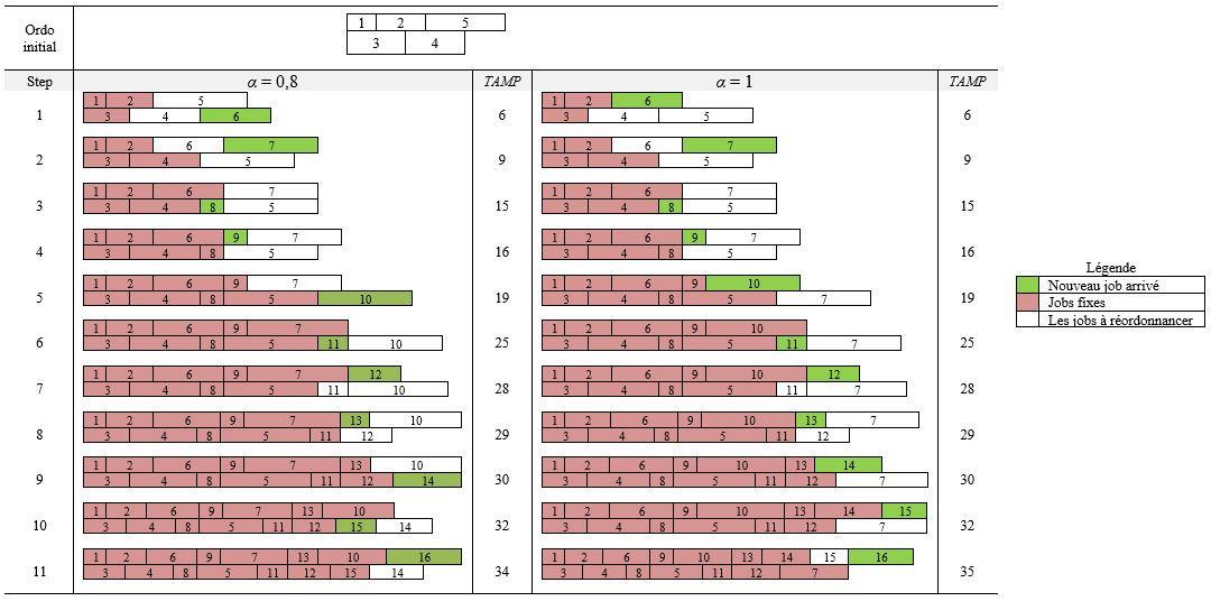

Figure 16. Diagramme de Gantt des solutions obtenues

Jusqu'à l'étape 8, la valeur du *TAMP* est la même dans les deux cas. En termes de stabilité, les solutions fournies sont différentes. Lorsque $\alpha = 1$, le job 10 est placé avant le job 7. Tandis que lorsque $\alpha = 0{,}8$, le job 7 est placé avant le job 10 pour ne pas trop perturber l'ordonnancement précédent, car la stabilité est alors prise en compte. Dans ce dernier cas, l'ordonnancement est plus stable, comparé au cas $\alpha = 1$. A l'étape 9, lorsque $\alpha = 1$, le modèle place le job 14 avant le job 7. Lorsque $\alpha = 0{,}8$, le modèle choisi l'inverse. Ce choix impacte la valeur du *TAMP* dans l'étape 11, pour laquelle nous avons observé une différence d'une unité de temps entre les deux cas.

Cet exemple montre que lorsque $\alpha = 1$, le modèle fournit une solution qui optimise l'efficacité, mais au niveau de la stabilité, il choisit arbitrairement une solution parmi plusieurs solutions, car la stabilité n'est pas prise en compte. Ce choix est aléatoire et peut produire de mauvaises solutions en termes d'efficacité dans les étapes suivantes. Cependant, lorsque $\alpha = 0{,}8$, puisque la stabilité est considérée dans la fonction objectif, le modèle ne fournit qu'une seule solution, celle qui optimise à la fois l'efficacité et la stabilité.

Pour confirmer ce comportement, une étude a été menée sur 25 instances différentes, générées aléatoirement. Dans la sous-section 5.2.2, des expérimentations ont été effectuées avec des durées de jobs normalement distribuées et dans la sous-section 5.2.3, avec des durées de jobs uniformément distribuées, selon les paramètres présentés dans la section 5.1.

5.2.2 Expérimentations avec des durées de jobs normalement distribuées

Les expériences ont été effectuées sur deux machines parallèles identiques. Le Tableau 25 présente les valeurs finales du *TAMP*, de la *DMDFP*, et f_1 en fonction de α pour 25 instances aléatoires, avec 5 jobs initiaux et $p_\theta = 0{,}8$. En moyenne, 20 jobs arrivent par instance sur un horizon $T = 24$ *ut*.

Tableau 25. Valeurs finales du *TAMP*, *DMDFP*, et f_1 en fonction de α avec p_θ = 0,8

	α = 1			α = 0,9			α = 0,8			α = 0,7			α = 0,6			α = 0,5		
	(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
Instance 1	71	40	71	71	35	67,4	71	35	63,8	71	35	60,2	74	32	57,2	77	27	52
Instance 2	66	37	66	69	35	65,6	69	35	62,2	72	28	58,8	72	28	54,4	83	16	49,5
Instance 3	99	49	99	93	40	87,7	93	40	82,4	94	37	76,9	95	27	67,8	96	23	59,5
Instance 4	102	62	102	102	49	96,7	101	51	91	101	51	86	103	41	78,2	103	41	72
Instance 5	65	38	65	69	30	65,1	69	30	61,2	70	27	57,1	66	24	49,2	69	19	44
Instance 6	110	67	110	106	63	101,7	106	63	97,4	106	63	93,1	110	45	84	111	37	74
Instance 7	78	39	78	78	36	73,8	78	36	69,6	78	36	65,4	89	26	63,8	89	22	55,5
Instance 8	119	75	119	112	54	106,2	112	54	100,4	112	54	94,6	112	39	82,8	112	33	72,5
Instance 9	82	62	82	92	45	87,3	92	45	82,6	92	41	76,7	96	36	72	1101	29	65
Instance 10	45	20	45	47	20	44,3	46	22	41,2	43	18	35,5	43	18	33	56	8	32
Instance 11	97	55	97	99	41	93,2	99	41	87,4	100	38	81,4	100	32	72,8	102	28	65
Instance 12	81	47	81	81	44	77,3	81	44	73,6	81	44	69,9	86	30	63,6	104	19	61,5
Instance 13	64	36	64	64	29	60,5	64	29	57	69	29	57	76	24	55,2	78	14	46
Instance 14	78	53	78	78	48	75	78	48	72	78	48	69	87	34	65,8	88	20	54
Instance 15	100	64	100	100	54	95,4	100	54	90,8	102	45	84,9	106	41	80	107	38	72,5
Instance 16	98	55	98	98	50	93,2	91	43	81,4	91	43	76,6	99	37	74,2	103	34	68,5
Instance 17	60	33	60	60	29	56,9	60	29	53,8	61	26	50,5	73	19	51,4	73	19	46
Instance 18	111	66	111	103	61	98,8	100	63	92,6	105	55	90	110	47	84,8	112	31	71,5
Instance 19	119	63	119	115	50	108,5	115	50	102	115	50	95,5	115	47	87,8	120	36	78
Instance 20	99	73	99	99	64	95,5	99	64	92	102	58	88,8	104	46	80,8	118	43	80,5
Instance 21	100	61	100	104	51	98,7	104	51	93,4	106	45	87,7	106	37	78,4	106	33	69,5
Instance 22	87	65	87	87	53	83,6	87	53	80,2	96	45	80,7	96	39	73,2	97	34	65,5
Instance 23	98	51	98	98	38	92	98	38	86	99	37	80,4	97	38	73,4	104	30	67
Instance 24	91	69	91	91	65	88,4	91	65	85,8	93	61	83,4	106	44	81,2	108	26	67
Instance 25	110	73	110	110	60	105	110	60	100	110	55	93,5	119	49	91	126	40	83
Moyennes	89,2	54,12	89,2	89,04	45,76	84,712	88,56	45,72	79,992	89,88	42,76	75,744	93,6	35,2	70,24	137,72	28	62,9

(1) *TAMP*, (2) *DMDFP*, (3) f_1, Les expériences ont été conduite avec 5 jobs initiaux, p_θ =0,8 (en moyenne 20 jobs arrivent par instance), T=24 *ut*, et m=2.

L'effet proactif a été observé sur 8 instances parmi les 25 testées (en gras). Le *TAMP* diminue lorsque α passe de 1 à 0,8. Puis, il augmente lorsque α passe au-dessous de 0,7.

Lorsque α = 1, le système ne considère pas la stabilité. Si plusieurs solutions fournissent la même valeur du *TAMP*, le modèle choisit arbitrairement l'une d'entre elles. Comme observé sur la Figure 16, ce choix peut impacter la valeur du *TAMP* dans les séquences qui suivent. Nous concluons alors que le fait de considérer la stabilité peut générer un effet proactif lorsque α est compris entre 0,7 et 0,9, ce qui fournit des meilleurs résultats que ceux obtenus lorsque l'on ne considère que l'efficacité seule. Cet effet proactif apparaît quand le modèle a plusieurs choix à faire lors d'une étape donnée. Autrement dit, quand le nombre de jobs concernés par le réordonnancement est grand.

Avec p_θ = 0,5 et p_θ = 0,2, d'autres tests ont été menés sur deux machines parallèles identiques. Les valeurs finales du *TAMP*, de la *DMDFP*, et f_1 en fonction de α pour 25 instances aléatoires sont respectivement présentés dans le Tableau 26 et le Tableau 27.

Tableau 26. Valeurs finales du *TAMP*, *DMDFP*, et f_1 en fonction de α avec p_θ = 0,5

	α = 1			α = 0,9			α = 0,8			α = 0,7			α = 0,6			α = 0,5		
	(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
Instance 1	19	6	19	19	4	17,5	19	4	16	19	4	14,5	19	4	13	19	4	11,5
Instance 2	7	3	7	7	2	6,5	7	2	6	7	2	5,5	7	2	5	7	2	4,5
Instance 3	13	5	13	13	2	11,9	13	2	10,8	13	2	9,7	18	0	10,8	18	0	9
Instance 4	26	2	26	26	0	23,4	26	0	20,8	26	0	18,2	26	0	15,6	26	0	13
Instance 5	8	2	8	8	0	7,2	8	0	6,4	8	0	5,6	8	0	4,8	8	0	4
Instance 6	19	0	19	19	0	17,1	19	0	15,2	19	0	13,3	19	0	11,4	19	0	9,5
Instance 7	22	3	22	22	3	20,1	22	3	18,2	22	3	16,3	22	3	14,4	22	3	12,5
Instance 8	19	1	19	19	1	17,2	19	1	15,4	19	1	13,6	19	1	11,8	19	1	10
Instance 9	17	4	17	17	3	15,6	17	3	14,2	17	3	12,8	17	3	11,4	17	3	10
Instance 10	7	2	7	7	1	6,4	7	1	5,8	7	1	5,2	7	1	4,6	7	1	4
Instance 11	14	2	14	14	2	12,8	14	2	11,6	14	2	10,4	14	2	9,2	14	2	8
Instance 12	10	4	10	10	4	9,4	10	4	8,8	10	4	8,2	11	2	7,4	11	2	6,5
Instance 13	12	2	12	12	2	11	12	2	10	12	2	9	13	0	7,8	13	0	6,5
Instance 14	14	3	14	14	1	12,7	14	1	11,4	14	1	10,1	14	1	8,8	14	1	7,5
Instance 15	13	4	13	15	3	13,8	15	3	12,6	15	3	11,4	15	3	10,2	15	3	9
Instance 16	14	5	14	14	2	12,8	14	2	11,6	14	2	10,4	14	2	9,2	14	2	8
Instance 17	15	2	15	15	2	13,7	15	2	12,4	15	2	11,1	15	2	9,8	15	2	8,5
Instance 18	13	3	13	13	3	12	13	3	11	13	3	10	13	3	9	13	9	8
Instance 19	17	6	17	17	6	15,9	17	6	14,8	17	6	13,7	17	6	12,6	17	6	11,5
Instance 20	15	5	15	15	5	14	15	5	13	15	5	12	15	4	10,6	16	3	9,5
Instance 21	13	3	13	13	3	12	13	3	11	13	3	10	13	3	9	13	3	8
Instance 22	16	10	16	16	10	15,4	16	10	14,8	16	10	14,2	21	5	14,6	21	5	13
Instance 23	10	3	10	10	2	9,2	10	2	8,4	10	2	7,6	10	2	6,8	10	2	6
Instance 24	18	5	18	18	5	16,7	18	5	15,4	18	5	14,1	18	5	12,8	18	5	11,5
Instance 25	19	0	19	19	0	17,1	19	0	15,2	19	0	13,3	19	0	11,4	19	0	9,5
Moyennes	14,8	3,4	14,8	14,88	2,64	13,656	14,88	2,64	12,432	14,88	2,64	11,208	15,36	2,16	10,08	15,4	2,36	8,76

(1) *TAMP*, (2) *DMDFP*, (3) f_1, Les expériences ont été conduite avec 5 jobs initiaux, p_θ =0,5 (en moyenne 12 jobs arrivent par instance), T=24 *ut*, et m=2.

Tableau 27. Valeurs finales du *TAMP*, *DMDFP*, et f_1 en fonction de α avec p_θ = 0,2

	α = 1			α = 0,9			α = 0,8			α = 0,7			α = 0,6			α = 0,5		
	(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
Instance 1	14	4	14	14	4	13	14	4	12	14	4	11	14	4	10	14	4	9
Instance 2	10	2	10	10	2	9,2	10	2	8,4	10	2	7,6	10	2	6,8	10	2	6
Instance 3	16	5	16	16	2	14,6	16	2	13,2	16	2	11,8	21	0	12,6	21	0	10,5
Instance 4	9	2	9	9	0	8,1	9	0	7,2	9	0	6,3	9	0	5,4	9	0	4,5
Instance 5	8	0	8	8	0	7,2	8	0	6,4	8	0	5,6	8	0	4,8	8	0	4
Instance 6	21	0	21	21	0	18,9	21	0	16,8	21	0	14,7	21	0	12,6	21	0	10,5
Instance 7	16	1	16	16	1	14,5	16	1	13	16	1	11,5	16	1	10	16	1	8,5
Instance 8	15	3	15	15	0	13,5	15	0	12	15	0	10,5	15	0	9	15	0	7,5
Instance 9	17	0	17	17	0	15,3	17	0	13,6	17	0	11,9	17	0	10,2	17	0	8,5
Instance 10	8	3	8	8	3	7,5	8	3	7	8	3	6,5	8	3	6	8	3	5,5
Instance 11	18	4	18	21	2	19,1	21	2	17,2	21	2	15,3	21	2	13,4	21	2	11,5
Instance 12	15	6	15	12	4	11,2	12	4	10,4	12	4	9,6	12	4	8,8	12	4	8
Instance 13	10	5	10	10	3	9,3	10	3	8,6	10	3	7,9	10	3	7,2	10	3	6,5
Instance 14	5	0	5	5	0	4,5	5	0	4	5	0	3,5	5	0	3	5	0	2,5
Instance 15	12	5	12	12	3	11,1	12	3	10,2	12	3	9,3	12	3	8,4	12	3	7,5
Instance 16	8	2	8	8	2	7,4	8	2	6,8	8	2	6,2	8	2	5,6	8	2	5
Instance 17	10	0	10	10	0	9	10	0	8	10	0	7	10	0	6	10	0	5
Instance 18	9	2	9	9	2	8,3	9	2	7,6	9	2	6,9	9	2	6,2	9	2	5,5
Instance 19	8	1	8	8	1	7,3	8	1	6,6	8	1	5,9	8	1	5,2	8	1	4,5
Instance 20	10	3	10	10	3	9,3	10	3	8,6	10	3	7,9	10	3	7,2	10	3	6,5
Instance 21	16	6	16	17	5	15,8	17	5	14,6	17	5	13,4	17	5	12,2	17	5	11
Instance 22	12	4	12	13	4	12,1	13	4	11,2	13	4	10,3	13	4	9,4	13	4	8,5
Instance 23	11	4	11	11	4	10,3	11	4	9,6	15	4	11,7	15	4	10,6	15	4	9,5
Instance 24	12	3	12	16	3	14,7	16	3	13,4	16	3	12,1	16	3	10,8	16	3	9,5
Instance 25	6	0	6	6	0	5,4	6	0	4,8	6	0	4,2	6	0	3,6	6	0	3
Moyennes	11,84	2,6	11,84	12,08	1,92	11,064	12,08	1,92	10,048	12,24	1,92	9,144	12,44	1,84	8,2	12,44	1,84	7,14

(1) *TAMP*, (2) *DMDFP*, (3) f_1, Les expériences ont été conduite avec 5 jobs initiaux, p_θ =0,2 (en moyenne 5 jobs arrivent par instance), T=24 *ut*, et m=2.

Dans le Tableau 26, la moyenne du *TAMP* augmente lorsque α est au-dessous de 0,7. L'effet proactif n'apparait plus car le nombre de jobs à réordonnancer diminue puisque p_θ = 0,5. Dans le Tableau 27, la moyenne du *TAMP* diminue lorsque α augmente. Lorsque p_θ = 0,2, la fréquence d'apparition des jobs est petite. Ainsi, l'effet proactif disparaît. Ce dernier cas a été testé pour 7 jobs initiaux. Les résultats sont présentés dans le Tableau 28.

Tableau 28. Valeurs finales du *TAMP*, *DMDFP*, et f_1 en fonction de α avec p_θ=0,2 et 7 jobs initiaux

	α = 1			α = 0,9			α = 0,8			α = 0,7			α = 0,6			α = 0,5		
	(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
Instance 1	61	15	61	61	13	56,2	61	13	51,4	61	13	46,6	61	13	41,8	63	11	37
Instance 2	63	13	63	63	13	58	63	13	53	63	13	48	63	13	43	65	11	38
Instance 3	58	13	58	58	13	53,5	58	13	49	58	13	44,5	58	13	40	60	11	35,5
Instance 4	56	11	56	58	11	53,3	58	11	48,6	58	11	43,9	58	11	39,2	58	11	34,5
Instance 5	64	13	64	63	9	57,6	63	9	52,2	63	9	46,8	63	9	41,4	65	7	36
Instance 6	69	16	69	69	16	63,7	69	16	58,4	69	16	53,1	69	16	47,8	71	14	42,5
Instance 7	44	9	44	44	7	40,3	44	7	36,6	44	7	32,9	44	7	29,2	44	7	25,5
Instance 8	63	22	63	63	22	58,9	63	22	54,8	63	22	50,7	63	22	46,6	63	22	42,5
Instance 9	56	11	56	56	11	51,5	56	11	47	56	11	42,5	56	11	38	56	11	33,5
Instance 10	54	4	54	54	4	49	54	4	44	54	4	39	54	4	34	54	4	29
Instance 11	75	15	75	75	15	69	75	15	63	75	15	57	78	11	51,2	79	10	44,5
Instance 12	63	14	63	60	11	55,1	60	11	50,2	60	11	45,3	60	11	40,4	60	11	35,5
Instance 13	61	16	61	62	11	56,9	62	11	51,8	62	11	46,7	62	11	41,6	64	9	36,5
Instance 14	68	14	68	68	14	62,6	68	14	57,2	68	14	51,8	68	14	46,4	68	14	41
Instance 15	61	17	61	61	15	56,4	61	15	51,8	61	15	47,2	61	15	42,6	61	15	38
Instance 16	61	13	61	61	13	56,2	61	13	51,4	61	13	46,6	61	13	41,8	63	11	37
Instance 17	64	16	64	64	14	59	64	14	54	64	14	49	64	14	44	64	12	38
Instance 18	54	13	54	54	11	49,7	54	11	45,4	54	11	41,1	54	11	36,8	56	9	32,5
Instance 19	61	13	61	61	13	56,2	61	13	51,4	61	13	46,6	61	13	41,8	61	13	37
Instance 20	56	6	56	56	6	51	56	6	46	56	6	41	59	6	37,8	59	6	32,5
Instance 21	59	19	59	59	19	55	59	19	51	59	19	47	59	19	43	61	17	39
Instance 22	54	12	54	54	12	49,8	56	12	47,2	54	12	41,4	56	12	38,4	54	12	33
Instance 23	71	16	71	71	16	65,5	71	16	60	71	16	54,5	72	13	48,4	74	10	42
Instance 24	81	21	81	81	15	74,4	81	15	67,8	81	10	59,7	81	10	52,6	83	8	45,5
Instance 25	56	13	56	56	11	51,5	56	11	47	56	11	42,5	56	11	38	58	9	33,5
Moyennes	61,32	13,8	61,32	61,28	12,6	56,412	61,36	12,6	51,608	61,28	12,4	46,616	61,64	12,12	41,832	62,56	11	36,78

(1) *TAMP*, (2) *DMDFP*, (3) f_1, Les expériences ont été conduite avec 7 jobs initiaux, p_θ =0,2 (en moyenne 5 jobs arrivent par instance), T=24 *ut*, et m=2.

Dans le Tableau 28, l'effet proactif apparaît dans 2 instances parmi les 25 testées (en gras). L'augmentation du nombre de jobs à réordonnancer a permis d'avoir cet effet. La Figure 17 présente les moyennes du *TAMP* en fonction de α pour tous les cas étudiés.

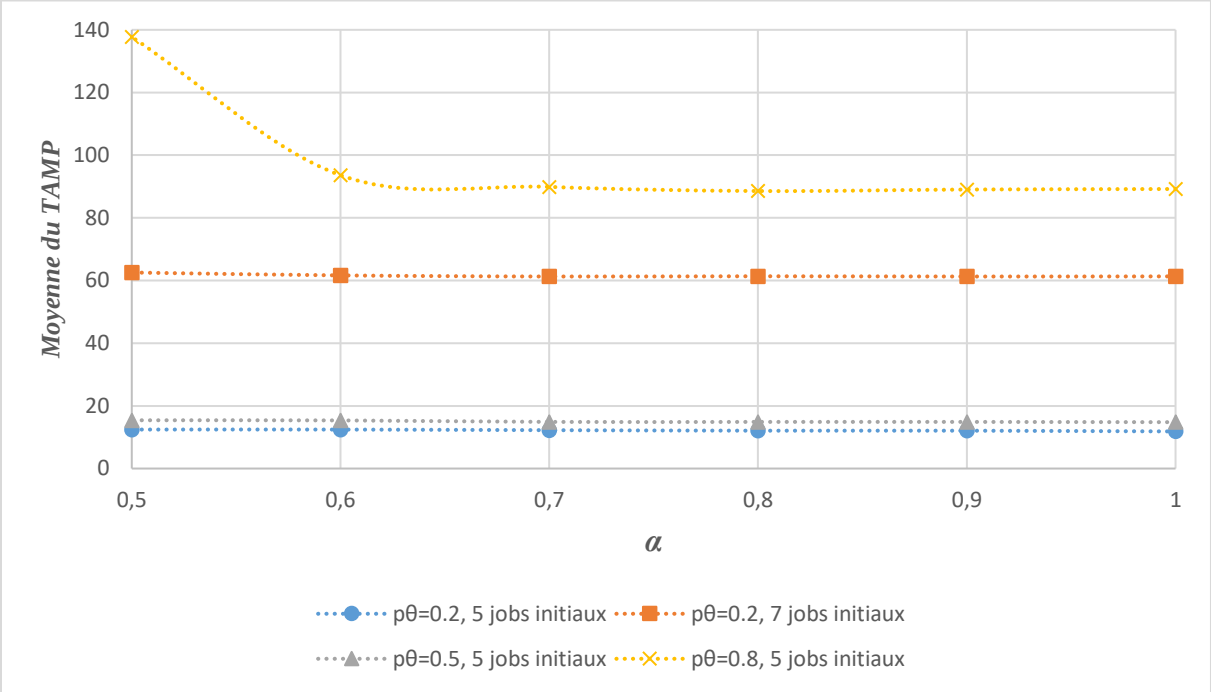

Figure 17. Moyennes du *TAMP* en fonction de α avec des durées de jobs normalement distribuées

La Figure 17 montre que l'augmentation de la valeur du *TAMP* est due à l'augmentation de p_θ, mais aussi du nombre des jobs initiaux. Dans le cas où p_θ = 0,8 avec 5 jobs initiaux, la moyenne du *TAMP* subit une large diminution lorsque α tend vers 1. Une diminution est produite aussi dans les autres cas mais elle est légère par rapport au dernier cas.

5.2.3 Expérimentations avec des durées de jobs uniformément distribuées

Dans cette section, la variation de α est étudiée pour des durées de jobs générées suivant une loi uniforme discrète, donnant des valeurs comprises entre 1 *ut* (15 minutes) et 10 *ut* (150 minutes), $U \sim (1,10)$. L'objectif est d'analyser le comportement des critères sur un ensemble plus large d'instances.

Dans le cas où $p_\theta = 0,8$, la PLNE n'arrive pas à résoudre le problème en un temps raisonnable lorsque les durées des jobs sont grandes. La machine consomme plus de temps à exécuter les jobs de grande durée. Dans ce cas, plusieurs jobs s'accumulent pendant le temps d'occupation des machines. Ainsi, le nombre de jobs à réordonnancer à chaque itération augmente, ce qui rend plus difficile la résolution du problème. C'est pour cela que nous avons effectué nos études avec p_θ = 0,5 et p_θ = 0,2. Les valeurs finales du *TAMP*, de la *DMDFP*, et f_1 en fonction de α pour 25 instances aléatoires sont respectivement présentés dans le Tableau 29 et le Tableau 30.

Tableau 29. Variation de α avec des durées uniformément distribuées et p_θ = 0,5

	α = 1			α = 0,9			α = 0,8			α = 0,7			α = 0,6			α = 0,5		
	(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
Instance 1	232	86	232	230	78	214,8	233	81	202,6	233	75	185,6	235	68	168,2	246	57	151,5
Instance 2	169	102	169	167	84	158,7	167	84	150,4	170	67	139,1	179	58	130,6	209	51	130
Instance 3	241	104	241	241	97	226,6	241	97	212,2	245	76	194,3	245	76	177,4	245	76	160,5
Instance 4	145	89	145	140	74	133,4	140	74	126,8	140	74	120,2	147	65	114,2	148	56	102
Instance 5	209	102	209	238	93	223,5	245	70	210	245	70	192,5	245	70	175	251	60	155,5
Instance 6	236	96	236	205	106	195,1	236	92	207,2	241	81	193	241	81	177	241	81	161
Instance 7	173	88	173	169	77	159,8	170	68	149,6	171	66	139,5	171	60	126,6	181	45	113
Instance 8	184	86	184	184	74	173	184	75	162,2	185	72	151,1	188	61	137,2	191	52	121,5
Instance 9	197	106	197	198	86	186,8	198	86	175,6	230	77	184,1	230	77	168,8	230	77	153,5
Instance 10	265	134	265	268	124	253,6	269	111	237,4	272	102	221	272	102	204	301	51	176
Instance 11	214	99	214	217	75	202,8	217	75	188,6	217	75	174,4	217	75	160,2	217	75	146
Instance 12	255	105	255	255	98	239,3	255	98	223,6	259	86	207,1	263	69	185,4	263	69	166
Instance 13	236	101	236	236	95	221,9	236	95	207,8	241	80	192,7	242	68	172,4	246	64	155
Instance 14	213	125	213	226	100	213,4	226	89	198,6	227	78	182,3	228	75	166,8	230	70	150
Instance 15	187	92	187	186	76	175	186	76	164	187	70	151,9	187	70	140,2	187	70	128,5
Instance 16	150	69	150	139	59	131	139	59	123	146	66	122	155	47	111,8	155	47	101
Instance 17	171	61	171	173	41	159,8	173	41	146,6	174	38	133,2	174	38	119,6	196	30	113
Instance 18	187	73	187	187	65	174,8	187	65	162,6	187	65	150,4	189	61	137,8	190	54	122
Instance 19	185	95	185	185	84	174,9	185	80	164	185	80	153,5	185	80	143	185	80	132,5
Instance 20	192	101	192	196	84	184,8	196	78	172,4	206	57	161,3	209	42	142,2	209	42	125,5
Instance 21	188	93	188	188	91	178,3	188	72	164,8	188	72	153,2	188	72	141,6	194	61	127,5
Instance 22	134	74	134	126	74	120,8	126	74	115,6	126	64	107,4	126	64	101,2	137	54	95,5
Instance 23	240	117	240	245	114	231,9	245	114	218,8	245	111	204,8	265	89	194,6	265	89	177
Instance 24	146	94	146	146	80	139,4	146	80	132,8	156	59	126,9	156	59	117,2	156	49	102,5
Instance 25	197	106	197	197	105	187,8	197	105	178,6	197	93	165,8	197	93	155,4	216	85	150,5
Moyennes	197,84	95,92	197,84	197,68	85,36	186,448	199,4	81,56	175,832	202,92	74,16	164,292	205,36	68,8	150,736	211,56	61,8	136,68

(1) *TAMP*, (2) *DMDFP*, (3) f_1, Les expériences ont été conduite avec 5 jobs initiaux, p_θ =0,5 (en moyenne 12 jobs arrivent par instance), T=24 *ut*, et m=2.

Comme montré dans le Tableau 29, la moyenne du *TAMP* diminue lorsque α = 0,9 comparée à α = 1. Puis, elle réaugmente lorsque α passe au-dessous de 0,9. Ce comportement est aussi le résultat de l'effet proactif. Ce dernier se produit pour 7 instances parmi les 25 testées (en gras). Plus des jobs de longues durées apparaissent, plus l'effet proactif est observé. En effet, lorsqu'un un job de longue durée occupe la machine, d'autres jobs peuvent s'accumuler car les machines sont occupées. Par conséquent, plusieurs solutions similaires en termes de stabilité sont possibles lorsque α = 1, car la *DMDFP* n'est pas considérée. Le choix de l'une d'entre elles est aléatoire et peut impacter l'efficacité dans les étapes suivantes. En conclusion, l'effet proactif peut non seulement être la conséquence d'un grand nombre de jobs, mais aussi celle de la prise en compte de jobs de longues durées.

Cependant, l'effet proactif disparaît lors de petites fréquences d'apparition, car le nombre de jobs diminue, comme présenté dans le Tableau 30 avec p_θ = 0,2.

Tableau 30. Variation de α avec des durées uniformément distribuées et p_θ = 0,2

	α =1			α =0,9			α =0,8			α =0,7			α =0,6			α =0,5		
	(1)	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**	**(2)**	**(3)**
Instance 1	79	11	79	79	11	72,2	79	11	65,4	79	11	58,6	79	11	51,8	79	8	43,5
Instance 2	53	9	53	53	7	48,4	53	7	43,8	53	7	39,2	53	7	34,6	53	7	30
Instance 3	52	12	52	52	9	47,7	54	9	45	54	9	40,5	54	9	36	54	7	30,5
Instance 4	32	9	32	32	9	29,7	32	9	27,4	32	9	25,1	32	9	22,8	32	9	20,5
Instance 5	36	9	36	47	1	42,4	47	1	37,8	47	1	33,2	47	1	28,6	47	1	24
Instance 6	48	10	48	48	10	44,2	49	11	41,4	49	11	37,6	49	11	33,8	49	11	30
Instance 7	51	0	51	51	0	45,9	51	0	40,8	51	0	35,7	51	0	30,6	51	0	25,5
Instance 8	65	8	65	65	8	59,3	65	8	53,6	65	8	47,9	65	8	42,2	65	8	36,5
Instance 9	97	29	97	97	19	89,2	97	19	81,4	97	19	73,6	100	13	65,2	100	13	56,5
Instance 10	93	28	93	93	19	85,6	93	19	78,2	93	19	70,8	93	19	63,4	94	16	55
Instance 11	79	2	79	79	2	71,3	79	2	63,6	79	2	55,9	79	2	48,2	79	2	40,5
Instance 12	87	13	87	89	12	81,3	89	12	73,6	89	12	65,9	89	12	58,2	89	12	50,5
Instance 13	60	22	60	60	16	55,6	60	16	51,2	60	16	46,8	60	16	42,4	60	16	38
Instance 14	71	12	71	71	5	64,4	71	5	57,8	53	7	39,2	71	5	44,6	71	5	38
Instance 15	38	10	32,4	38	10	35,2	38	10	32,4	38	10	29,6	49	4	31	49	4	26,5
Instance 16	35	13	35	38	10	35,2	38	10	32,4	43	8	32,5	43	8	29	43	8	25,5
Instance 17	19	3	19	19	3	17,4	19	3	15,8	19	3	14,2	19	3	12,6	19	3	11
Instance 18	31	11	31	31	11	29	31	11	27	31	11	25	31	11	23	31	11	21
Instance 19	52	20	52	52	20	48,8	52	20	45,6	52	20	42,4	52	20	39,2	52	20	36
Instance 20	32	8	32	33	6	30,3	33	6	27,6	33	6	24,9	33	6	22,2	33	6	19,5
Instance 21	51	10	51	51	10	46,9	57	5	46,6	57	5	41,4	57	5	36,2	57	5	31
Instance 22	38	16	38	38	11	35,3	38	11	32,6	38	11	29,9	38	11	27,2	38	11	24,5
Instance 23	68	19	68	70	19	64,9	70	19	59,8	70	19	54,7	70	19	49,6	79	16	47,5
Instance 24	44	12	44	44	12	40,8	44	12	37,6	44	12	34,4	44	12	31,2	47	3	25
Instance 25	76	12	76	76	12	69,6	76	12	63,2	76	12	56,8	76	12	50,4	76	12	44
Moyennes	55,48	12,32	55,256	56,24	10,08	51,624	56,6	9,92	47,264	56,08	9,92	42,232	57,36	9,36	38,16	57,88	8,56	33,22

(1) *TAMP*, (2) *DMDFP*, (3) f_1, Les expériences ont été conduite avec 5 jobs initiaux, p_θ =0,2 (en moyenne 5 jobs arrivent par instance), T=24 *ut*, et m=2.

La Figure 18 présente les moyennes du *TAMP* en fonction de α pour tous les cas étudiés avec des durées uniformément distribuées.

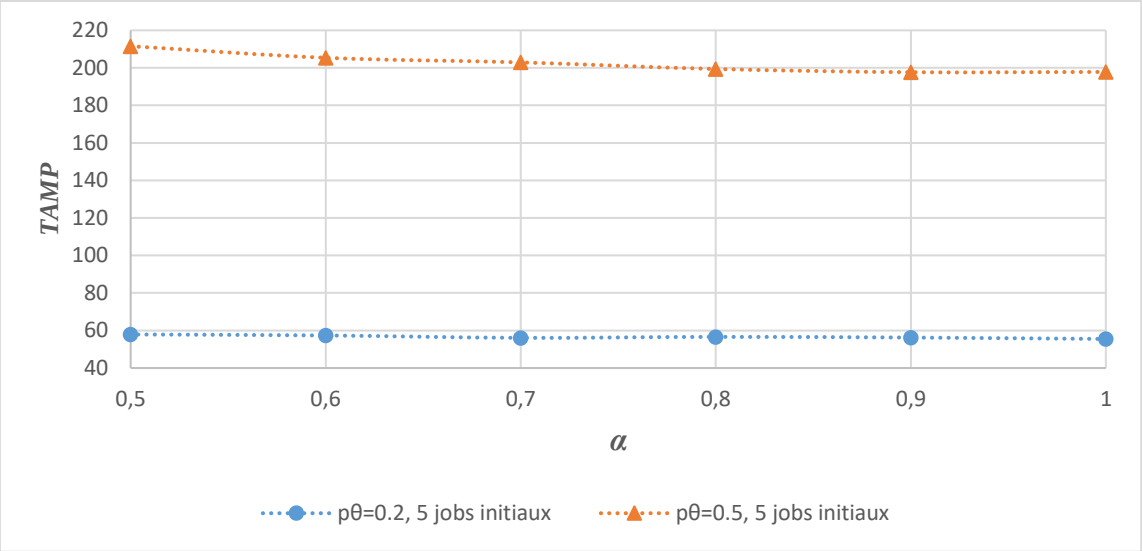

Figure 18. Moyennes du *TAMP* en fonction de α avec des durées uniformément distribuées

Une large augmentation de la valeur du *TAMP* est observée dans la Figure 18 lorsque p_θ augmente. Cela est dû à l'arrivée d'un grand nombre de jobs avec des longues durées. Une diminution de la moyenne du *TAMP* se produit aussi lorsque α tend vers 1, mais elle est légère.

5.3 Optimisation lexicographique pour minimiser le nombre de jobs qui changent de machine

L'objectif de l'optimisation lexicographique est de minimiser f_2, la fonction objectif qui représente le nombre de jobs qui changent de machine après le réordonnancement, avec $f_1 \leq f_1^*$ comme contrainte. Cependant, si le modèle n'arrive pas à minimiser f_2 à zéro, la contrainte est relaxée. Elle devient $f_1 \leq f_1^*(1+\varepsilon)$, avec ε un coefficient de relaxation. Dans ce qui suit, l'optimisation lexicographique est appliquée sur le même exemple que précédemment (section 5.2.1).

5.3.1 Exemple

Dans la solution présentée dans le Tableau 23, le nombre de jobs qui changent de machine n'est pas optimisé. A chaque étape de réordonnancement, le modèle optimise f_1 sans tenir compte de f_2. On voit dans le Tableau 31 le résultat de l'application de l'optimisation lexicographique sur ce même exemple. La valeur de ε est initialisée, à chaque étape, à zéro, puis elle augmente si $f_2 \neq 0$ avec un pas de valeur 0,01, jusqu'à ce que f_2 soit égale à 0. Dans cet exemple, la valeur de α est fixée à 0,8, valeur pour laquelle l'effet proactif a été observé.

Tableau 31. Minimisation de f_2 avec $f_1 \leq f_1^*(1+\varepsilon)$ comme contrainte

ε	(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
	$2^{ème}$ Ordo					$3^{ème}$ Ordo					$4^{ème}$ Ordo					$5^{ème}$ Ordo				
0	5	2	4,4	2	0	8	2	6,8	2	0	13	3	11	0	0	16	3	13,4	1	0
	6^{th} Ordo					$7^{ème}$ Ordo					$8^{ème}$ Ordo					$9^{ème}$ Ordo				
0	19	3	15,8	0	0	25	4	20,8	0	0	28	4	23,2	0	0	29	5	24,2	0	0
	$10^{ème}$ Ordo					$11^{ème}$ Ordo					$12^{ème}$ Ordo					13^{th} Ordo				
0	30	5	25	0	0	32	7	27	0	0	36	8	30,4	1	1	42	7	35	1	0
0,01											36	8	30,7	1	1					
0,02											36	8	31,0	1	1					
0,03											36	8	31,3	1	1					
0,04											36	8	31,6	1	1					
0,05											38	7	31,9	1	0					
	$14^{ème}$ Ordo					$15^{ème}$ Ordo					$16^{ème}$ Ordo					$17^{ème}$ Ordo				
0	44	7	0	0	0	46	9	38,6	0	0	48	9	40,2	2	0	53	11	44,6	2	0
	$18^{ème}$ Ordo					$19^{ème}$ Ordo					$20^{ème}$ Ordo					$21^{ème}$ Ordo				
0	57	11	47,8	2	0	59	12	49,4	0	0	62	15	52,6	1	1	64	17	54,6	2	0
0,01											62	15	53,1	1	1					
0,02											63	16	53,6	1	0					

(1) : *TAMP* (2) : *DMDFP* (3) : $f_1^*(1+\varepsilon)$ (4) : f_2 avant la lexicographie (5) : f_2 après la lexicographie

Comme présenté dans le Tableau 31, les valeurs de f_2 diminuent jusqu'à zéro après l'application de l'optimisation lexicographique. Du 2ème jusqu'au 11ème Ordo, le modèle réduit la valeur de f_2 à zéro sans dégrader f_1. Dans le 12ème Ordo, puisque le modèle n'arrivait pas à réduire f_2 à zéro, nous avons augmenté la valeur de ε jusqu'à 0,05, en utilisant $f_1^*(1+\varepsilon)$ comme contrainte pour obtenir $f_2 = 0$. Dans le 13ème Ordo, nous avons procédé de la même façon, en considérant la dernière solution obtenue lors de l'étape précédente.

Parmi les 20 ordonnancements, la valeur de ε a été augmentée seulement deux fois. Cela montre que dans la majorité des cas, nous pouvons minimiser f_2 sans dégrader f_1^*. Cependant, quand la valeur de ε est augmentée, la valeur de $f_1^*(1+\varepsilon)$ augmente aussi, ainsi que la valeur du *TAMP* et de la *DMDFP*. La valeur de f_1^* est alors plus grande dans les étapes suivantes, comme pour le 13ème Ordo par exemple, où la valeur de $f_1^* = 35$ comparée à sa valeur dans le Tableau 23, où elle est égale à 32,4.

En utilisant l'optimisation lexicographique, il est possible de réduire f_2 à zéro. Parfois, il est nécessaire de relaxer f_1^*, la fonction principale. Cette relaxation dégrade sa valeur. Dans ce cas, c'est aux décideurs de choisir ou non de relaxer f_1^*, et de choisir une valeur de ε en fonction de leur priorité. Dans ce qui suit, nous étudions l'optimisation lexicographique pour des instances différentes.

5.3.2 Expérimentation avec des durées de jobs normalement distribuées

L'optimisation lexicographique est appliquée pour 25 instances différentes, avec des durées de jobs qui suivent une loi normale, selon les paramètres du Tableau 21. Pour chaque instance, les valeurs du *TAMP*, de la *DMDFP*, de f_1 et de f_2 sont calculées à chaque étape. Si $f_2 > 0$, l'optimisation lexicographique est utilisée pour réduire la valeur de f_2 en considérant $\varepsilon = 0$. L'Erreur Relative (*ER*) est alors calculée pour le *TAMP*, la *DMDFP*, et f_1 pour évaluer la dégradation de leurs valeurs par rapport à leurs valeurs avant la lexicographie.

$$ER = \frac{La\ valeur\ après\ la\ lexicographie - La\ valeur\ avant\ la\ lexicographie}{La\ valeur\ avant\ la\ lexicographie} \qquad (2.15)$$

Dans le Tableau 32, pour chaque instance, la valeur moyenne de *ER* (*MER*) est calculée pour le *TAMP*, la *DMDFP*, et f_1, ainsi que le Nombre Moyen des Jobs Optimisés par itération (*NMJOL*), le nombre de fois où $f_2 = 0$, et le nombre de fois où $f_2 > 0$. Nous considérons que $\alpha = 0,8$ et $p_\theta = 0,8$. En moyenne, 20 jobs arrivent sur un horizon $T = 24$ *ut*.

Tableau 32. Optimisation lexicographique en utilisant des durées de jobs normalement distribuées avec ε=0

	***TAMP* final**	***DMDFP* final**	**f_1 final**	***MER TAMP***	***MER DMDFP***	***MER* f_1**	***NMJOL***	**Nombre de fois où f_2=0**	**Nombre de fois où f_2>0**
Instance 1	73,00	29,00	64,20	0,00	0,02	0,00	0,56	2,00	7,00
Instance 2	69,00	34,00	62,00	0,00	0,01	0,00	1,17	6,00	6,00
Instance 3	86,00	34,00	75,60	0,00	0,00	0,00	1,73	10,00	1,00
Instance 4	105,00	40,00	92,00	0,00	0,00	0,00	1,23	7,00	6,00
Instance 5	54,00	28,00	48,80	0,00	0,00	0,00	1,17	7,00	5,00
Instance 6	109,00	49,00	97,00	0,00	0,00	0,00	1,69	8,00	5,00
Instance 7	71,00	30,00	62,80	0,00	0,00	0,00	1,45	9,00	2,00
Instance 8	103,00	49,00	92,20	0,00	0,01	0,00	1,92	8,00	5,00
Instance 9	73,00	43,00	67,00	0,00	0,00	0,00	1,15	7,00	6,00
Instance 10	41,00	14,00	35,60	0,00	0,00	0,00	0,56	4,00	5,00
Instance 11	83,00	38,00	75,20	0,00	0,00	0,00	1,87	12,00	3,00
Instance 12	87,00	42,00	78,00	0,00	0,00	0,00	1,33	11,00	4,00
Instance 13	82,00	32,00	72,00	0,00	0,00	0,00	1,50	8,00	4,00
Instance 14	71,00	44,00	65,60	0,00	0,00	0,00	1,85	10,00	3,00
Instance 15	103,00	40,00	90,40	0,00	0,00	0,00	1,13	7,00	9,00
Instance 16	81,00	44,00	73,60	0,00	0,00	0,00	1,57	10,00	4,00
Instance 17	75,00	16,00	63,20	0,00	0,00	0,00	1,50	11,00	1,00
Instance 18	103,00	50,00	92,40	0,00	0,00	0,00	1,47	9,00	6,00
Instance 19	79,00	36,00	70,40	0,00	0,00	0,00	1,83	5,00	1,00
Instance 20	90,00	54,00	82,80	0,00	0,00	0,00	1,18	5,00	6,00
Instance 21	94,00	50,00	85,20	0,00	0,00	0,00	1,50	7,00	5,00
Instance 22	72,00	33,00	64,20	0,00	0,01	0,00	1,00	6,00	8,00
Instance 23	92,00	55,00	84,60	0,00	0,00	0,00	1,33	4,00	8,00
Instance 24	91,00	46,00	82,00	0,00	0,00	0,00	1,55	7,00	4,00
Instance 25	85,00	45,00	77,00	0,00	0,00	0,00	1,75	7,00	5,00
Moyennes	82,88	39	74,152	0,00	0,00	0,00	1,40	7,48	4,76

L'optimisation lexicographique a permis d'optimiser en moyenne 1,4 jobs par itération. Parmi les 20 étapes de réordonnancement, la lexicographie peut, en moyenne, réduire 7,48 fois f_2 à zéro sans dégrader f_1 la fonction principale (*MER* $f_1 = 0$). Cependant, la valeur de f_2 reste supérieure à zéro 4,76 fois en moyenne. Cela montre la nécessité d'utiliser une relaxation en augmentant la valeur de ε.

5.3.3 Expérimentation avec des durées de jobs uniformément distribuées

25 instances différentes sont générées avec des durées de jobs qui suivent une loi uniforme discrète. Les valeurs des durées de jobs varient entre 1 et 10 *ut*. Nous avons considéré que $\alpha = 0,8$ et $p_\theta = 0,5$, puisque le modèle n'arrive pas à résoudre le problème lorsque p_θ est au-dessus

de 0,5. En moyenne, 12 jobs arrivent sur un horizon $T = 24$ *ut*. Les résultats sont présentés dans le Tableau 33.

Tableau 33. Optimisation lexicographique en utilisant des durées de jobs uniformément distribuées avec ε=0

	***TAMP* final**	***DMDFP* final**	**f_1 final**	***MER TAMP***	***MER DMDFP***	***MER* f_1**	***NMJOL***	**Nombre de fois où f_2=0**	**Nombre de fois où f_2>0**
Instance 1	235,00	76,00	203,20	0,00	0,00	0,00	1,20	2,00	8,00
Instance 2	167,00	84,00	150,40	-0,01	0,04	0,00	1,11	2,00	7,00
Instance 3	215,00	79,00	187,80	0,00	0,00	0,00	0,71	1,00	6,00
Instance 4	140,00	74,00	126,80	0,00	0,00	0,00	1,75	3,00	5,00
Instance 5	255,00	65,00	217,00	0,00	0,00	0,00	1,60	2,00	3,00
Instance 6	213,00	83,00	187,00	0,00	0,00	0,00	0,80	2,00	7,00
Instance 7	172,00	63,00	150,20	0,00	0,00	0,00	0,56	2,00	7,00
Instance 8	203,00	94,00	181,20	0,00	0,00	0,00	0,86	2,00	5,00
Instance 9	135,00	71,00	122,20	0,00	0,00	0,00	2,22	6,00	3,00
Instance 10	201,00	75,00	175,80	0,00	0,00	0,00	0,89	3,00	6,00
Instance 11	175,00	64,00	152,80	0,00	0,00	0,00	0,75	2,00	6,00
Instance 12	192,00	87,00	171,00	0,00	0,00	0,00	1,00	1,00	7,00
Instance 13	201,00	97,00	180,20	0,00	0,00	0,00	1,14	2,00	5,00
Instance 14	226,00	89,00	198,60	0,00	0,00	0,00	1,70	4,00	6,00
Instance 15	187,00	77,00	165,00	0,00	-0,06	0,00	1,63	3,00	5,00
Instance 16	139,00	59,00	123,00	0,00	0,00	0,00	1,40	6,00	4,00
Instance 17	173,00	41,00	146,60	0,00	0,00	0,00	0,44	1,00	8,00
Instance 18	187,00	65,00	162,60	0,00	0,00	0,00	1,22	4,00	5,00
Instance 19	229,00	68,00	196,80	0,00	0,02	0,00	1,00	3,00	6,00
Instance 20	229,00	68,00	196,80	0,00	0,02	0,00	0,56	2,00	7,00
Instance 21	188,00	72,00	164,80	0,00	0,00	0,00	1,57	1,00	6,00
Instance 22	126,00	74,00	115,60	0,00	0,00	0,00	1,67	5,00	4,00
Instance 23	184,00	88,00	164,80	0,00	0,00	0,00	1,09	5,00	6,00
Instance 24	158,00	64,00	139,20	0,00	0,00	0,00	1,00	3,00	4,00
Instance 25	135,00	64,00	120,80	0,00	0,00	0,00	1,56	5,00	4,00
Moyennes	186,60	73,64	164,01	0,00	0,00	0,00	1,18	2,88	5,60

La variation des durées de jobs de 1 à 10 *ut* rend plus difficile la recherche des solutions équivalentes qui permettront d'avoir $f_2 = 0$ sans impacter f_1. Ceci explique la diminution du *NMJOL* et l'augmentation du nombre de fois où $f_2 > 0$. Dans ce cas, un petit changement dans la séquence a un grand impact sur la valeur de f_1.

5.3.4 Expérimentation en utilisant un coefficient de relaxation

Deux études ont été effectuées dans cette section, la première utilise une stratégie adaptative dans laquelle le coefficient de relaxation est itérativement modifié lorsque $f_2 > 0$. La seconde utilise un coefficient de relaxation défini a priori. Nous avons testé les mêmes instances que celles étudiées dans la section 5.3.2 pour assurer la comparaison.

5.3.4.1 Coefficient de relaxation itérativement modifié

La valeur de ε est initialisée à zéro à chaque itération, elle est augmentée lorsque $f_2 \neq 0$ avec un pas de 0,01, jusqu'à ce que $f_2 = 0$. Nous avons considéré que $\alpha = 0{,}8$ et $p_\theta = 0{,}8$. A chaque fois que ε augmente, nous présentons sa valeur finale, ainsi que la valeur de l'*ER* de chaque critère. Nous avons également calculé le nombre de fois où f_2 est optimisée avec $\varepsilon = 0$, et avec $\varepsilon > 0$. Le taux d'optimisation avec $\varepsilon = 0$ est calculé comme suit :

$$Taux\ d'optimisation\ avec\ (\varepsilon = 0) = \frac{Nombre\ d'optimisations\ avec\ \varepsilon = 0}{Nombre\ total\ d'optimisations} \qquad (2.18)$$

Selon le Tableau 34, l'optimisation lexicographique permet à 76% en moyenne de réduire f_2 à zéro sans utiliser la relaxation. La relaxation est utilisée en moyenne 2,64 fois parmi les 20 itérations de réordonnancement, avec un coefficient de relaxation égal à 0,03 en moyenne.

L'erreur relative (*RE*) du *DMDFP* peut être négative dans certains cas, car la minimisation de f_2 aide le critère de stabilité. Cependant, le *TAMP* se dégrade comparé à la *DMDFP*. Dans la $4^{ème}$ relaxation, les valeurs du *TAMP* sont grandes, l'*ER* devient plus petite comparée à la $1^{ère}$ relaxation. En conclusion, plus ε augmente, plus f_2 est minimisée, alors que f_1 se dégrade.

Tableau 34. Optimisation lexicographique avec ε itérativement modifié

	1ère relaxation				2ème relaxation				3ème relaxation				4ème relaxation				Nombre d'optimisation avec $\varepsilon>0$	Nombre d'optimisation avec $\varepsilon=0$	Taux d'optimisation avec $\varepsilon=0$
	ε	ER TAMP	ER DMDFP	ER f_1	ε	ER TAMP	ER DMDFP	ER f_1	ε	ER TAMP	ER DMDFP	ER f_1	ε	ER TAMP	ER DMDFP	ER f_1			
Instance 1	0,04	0,08	-0,40	0,04	0,03	0,02	0,07	0,03	0,02	0,02	0,04	0,02	0,02	0,02	0,04	0,02	4,00	7,00	0,64
Instance 2	0,07	0,09	-1,00	0,07	0,06	0,05	0,11	0,06	-	-	-	-	-	-	-	-	2,00	8,00	0,80
Instance 3	0,01	0,01	-0,06	0,01	-	-	-	-	-	-	-	-	-	-	-	-	1,00	8,00	0,89
Instance 4	0,08	0,06	0,25	0,08	0,05	0,04	0,17	0,05	0,04	0,03	0,11	0,04	0,07	0,06	0,17	0,07	4,00	7,00	0,64
Instance 5	0,06	0,07	-0,09	0,06	0,02	0,00	0,00	0,02	0,01	0,00	0,04	0,01	0,02	0,01	0,00	0,02	4,00	7,00	0,64
Instance 6	0,04	0,03	0,08	0,04	0,01	0,00	0,07	0,01	-	-	-	-	-	-	-	-	2,00	5,00	0,71
Instance 7	0,03	0,05	0,00	0,03	0,04	0,05	0,50	0,04	0,02	0,03	0,20	0,02	-	-	-	-	3,00	10,00	0,77
Instance 8	0,02	0,02	0,00	0,02	0,01	0,01	-0,05	0,01	-	-	-	-	-	-	-	-	2,00	10,00	0,83
Instance 9	0,02	0,02	0,00	0,02	-	-	-	-	-	-	-	-	-	-	-	-	1,00	6,00	0,86
Instance 10	0,06	0,07	-0,10	0,06	0,01	0,02	-0,18	0,01	0,02	0,03	0,07	0,02	0,02	0,01	0,04	0,02	4,00	5,00	0,56
Instance 11	0,05	0,06	-0,13	0,05	0,02	0,02	0,07	0,02	-	-	-	-	-	-	-	-	2,00	7,00	0,78
Instance 12	0,06	0,05	0,08	0,06	0,02	0,02	-0,04	0,02	0,05	0,04	0,08	0,05	-	-	-	-	3,00	10,00	0,77
Instance 13	0,06	0,06	0,17	0,06	0,11	0,10	0,25	0,11	0,04	0,04	0,09	0,04	-	-	-	-	3,00	6,00	0,67
Instance 14	0,05	0,07	-0,11	0,05	0,02	0,02	0,03	0,02	0,01	0,02	-0,07	0,01	0,04	0,03	0,05	0,04	4,00	7,00	0,64
Instance 15	0,08	0,06	0,50	0,08	0,03	0,02	0,08	0,03	0,02	0,01	0,02	0,02	-	-	-	-	3,00	13,00	0,81
Instance 16	0,04	0,05	-0,14	0,04	0,01	0,01	-0,03	0,01	-	-	-	-	-	-	-	-	2,00	9,00	0,82
Instance 17	0,02	0,02	0,11	0,03	0,01	0,00	0,11	0,01	-	-	-	-	-	-	-	-	2,00	9,00	0,82
Instance 18	0,06	0,05	0,11	0,06	0,01	0,02	-0,13	0,01	0,02	0,01	0,03	0,02	0,01	0,01	-0,02	0,01	4,00	7,00	0,64
Instance 19	0,06	0,13	0,00	0,06	-	-	-	-	-	-	-	-	-	-	-	-	1,00	10,00	0,91
Instance 20	0,03	0,04	-0,13	0,03	0,02	0,03	-0,17	0,02	0,02	0,01	0,03	0,02	0,01	0,00	0,03	0,01	4,00	8,00	0,67
Instance 21	0,04	0,08	-0,40	0,04	0,02	0,04	-0,20	0,02	0,01	0,01	0,03	0,01	-	-	-	-	3,00	14,00	0,82
Instance 22	0,04	0,07	-0,50	0,04	0,02	0,02	0,07	0,02	0,02	0,01	0,05	0,02	0,02	0,01	0,04	0,02	4,00	9,00	0,69
Instance 23	0,07	0,09	-1,00	0,07	0,04	0,06	0,17	0,04	-	-	-	-	-	-	-	-	2,00	15,00	0,88
Instance 24	0,05	0,09	0,00	0,09	-	-	-	-	-	-	-	-	-	-	-	-	1,00	13,00	0,93
Instance 25	0,05	0,06	-0,17	0,05	-	-	-	-	-	-	-	-	-	-	-	-	1,00	10,00	0,91

Moyennes	0,05	0,06	-0,12	0,05	0,03	0,03	0,05	0,03	0,02	0,02	0,06	0,02	0,03	0,02	0,04	0,03	2,64	8,80	0,76

5.3.4.2 Coefficient de relaxation défini a priori

L'optimisation lexicographique est effectuée en utilisant un coefficient de relaxation défini a priori. Deux valeurs de ε sont étudiées, 0,03 et 0,05. Les résultats sont respectivement présentés dans les Tableaux 35 et 36.

Tableau 35. Optimisation lexicographique avec des durées de jobs normalement distribuées et ε = 0,03

	***TAMP* final**	***DMDFP* final**	**f_1 final**	***MER TAMP***	***MER DMDFP***	***MER* f_1**	***NMJOL***	**Nombre de fois où $f_2 = 0$**	**Nombre de fois où $f_2 > 0$**
Instance 1	86,00	28,00	74,57	0,01	0,03	0,03	1,09	8,00	3,00
Instance 2	92,00	32,00	80,13	0,01	0,02	0,03	1,64	12,00	2,00
Instance 3	93,00	36,00	81,78	0,01	0,01	0,03	1,60	14,00	1,00
Instance 4	72,00	34,00	64,40	0,01	0,00	0,03	1,43	5,00	2,00
Instance 5	56,00	28,00	51,29	0,00	-0,01	0,03	1,50	10,00	4,00
Instance 6	111,00	48,00	98,67	0,02	0,03	0,03	2,00	11,00	0,00
Instance 7	75,00	30,00	66,33	0,02	0,05	0,03	1,80	10,00	0,00
Instance 8	75,00	30,00	66,33	0,01	0,04	0,03	1,80	12,00	2,00
Instance 9	92,00	50,00	83,64	0,01	0,04	0,03	1,33	12,00	3,00
Instance 10	41,00	14,00	35,60	0,01	0,00	0,03	0,56	4,00	5,00
Instance 11	84,00	39,00	75,60	0,01	0,03	0,03	1,93	12,00	1,00
Instance 12	84,00	44,07	76,01	0,01	0,01	0,03	1,36	12,00	2,00
Instance 13	86,00	35,04	75,81	0,01	0,05	0,03	1,55	8,00	3,00
Instance 14	78,00	41,00	70,60	0,01	0,00	0,03	2,07	13,00	1,00
Instance 15	111,00	51,00	99,09	0,01	0,01	0,03	1,33	12,00	3,00
Instance 16	86,00	45,34	77,87	0,01	0,02	0,03	1,82	9,00	2,00
Instance 17	81,00	23,00	71,48	0,00	0,00	0,03	1,30	9,00	1,00
Instance 18	103,00	57,00	94,35	0,01	0,03	0,03	1,67	11,00	4,00
Instance 19	84,00	37,89	74,78	0,01	0,01	0,03	1,57	6,00	1,00
Instance 20	95,00	57,00	87,96	0,01	0,01	0,03	1,64	11,00	3,00
Instance 21	109,00	44,00	98,06	0,02	0,01	0,03	1,93	12,00	2,00
Instance 22	72,00	33,00	65,10	0,01	0,03	0,03	1,61	13,00	5,00
Instance 23	104,00	49,00	94,76	0,02	-0,01	0,03	1,67	13,00	2,00
Instance 24	99,00	45,00	88,58	0,01	0,02	0,03	1,65	15,00	2,00
Instance 25	100,00	41,00	88,58	0,01	0,05	0,03	1,93	14,00	1,00
Moyennes	86,76	38,89	77,65	0,01	0,02	0,03	1,59	10,72	2,20

Tableau 36. Optimisation lexicographique avec des durées de jobs normalement distribuées et ε = 0,05

	***TAMP* final**	***DMDFP* final**	**f_1 final**	***MER TAMP***	***MER DMDFP***	***MER* f_1**	***NMJOL***	**Nombre de fois où $f_2 = 0$**	**Nombre de fois où $f_2 > 0$**
Instance 1	89,00	31,00	76,02	0,02	0,04	0,05	1,33	11,00	1,00
Instance 2	93,00	35,00	82,74	0,02	0,06	0,05	1,36	9,00	2,00
Instance 3	105,00	21,00	88,83	0,05	0,03	0,05	1,50	15,00	1,00
Instance 4	117,00	40,00	106,68	0,03	-0,03	0,05	1,64	8,00	3,00
Instance 5	117,00	40,00	106,68	0,01	-0,02	0,05	2,00	7,00	0,00
Instance 6	111,00	0,00	102,06	0,04	-0,09	0,05	1,50	11,00	1,00
Instance 7	85,00	29,00	77,07	0,03	-0,09	0,05	1,67	12,00	0,00
Instance 8	125,00	45,00	112,77	0,03	0,00	0,05	2,58	11,00	1,00
Instance 9	92,00	42,00	82,00	0,02	0,10	0,05	1,83	10,00	2,00
Instance 10	41,00	14,00	35,60	0,01	0,02	0,05	0,56	4,00	5,00
Instance 11	107,00	40,00	93,87	0,03	0,19	0,05	1,75	15,00	1,00
Instance 12	86,00	43,00	77,40	0,02	0,01	0,05	1,50	13,00	1,00
Instance 13	91,00	39,00	83,58	0,02	0,04	0,05	1,91	11,00	0,00
Instance 14	73,00	39,80	66,36	0,02	-0,01	0,05	2,15	13,00	0,00
Instance 15	121,00	53,00	107,52	0,03	0,06	0,05	1,76	16,00	1,00
Instance 16	85,00	47,00	80,22	0,02	0,00	0,05	1,77	13,00	0,00
Instance 17	53,00	23,00	47,00	0,01	0,09	0,05	1,42	12,00	0,00
Instance 18	107,00	52,00	96,60	0,02	0,09	0,05	1,38	13,00	3,00
Instance 19	84,00	35,00	74,55	0,03	0,02	0,05	1,57	6,00	1,00
Instance 20	113,00	43,00	101,64	0,02	-0,02	0,05	1,56	15,00	1,00
Instance 21	104,00	44,00	92,40	0,03	-0,01	0,05	1,65	16,00	1,00
Instance 22	82,00	50,00	75,60	0,02	0,09	0,05	1,27	12,00	3,00
Instance 23	109,00	61,00	100,38	0,02	0,23	0,05	1,50	13,00	3,00
Instance 24	90,00	52,00	83,37	0,01	0,03	0,05	1,92	11,00	1,00
Instance 25	106,00	40,00	93,24	0,02	0,03	0,05	2,31	13,00	0,00
Moyennes	95,44	38,35	85,77	0,02	0,03	0,05	1,66	11,60	1,28

Selon les Tableaux 35 et 36, l'augmentation de ε aide à optimiser la fonction f_2 puisque le nombre de fois où $f_2 = 0$ augmente. En revanche, f_1 se dégrade encore plus. Cela se traduit par l'augmentation de la valeur finale de f_1 qui passe de 77,65 lorsque ε = 0,03, à 85,77 lorsque ε = 0,05. La *MER* f_1 augmente également.

5.4 Etude du temps de résolution

Dans cette étude, nous évaluons sur 10 instances différentes la Durée Maximale d'une Itération (*DMI*), le Temps de Résolution (*TR*) de toutes les itérations et le nombre moyen des jobs optimisés par itération. Les instances étudiées contiennent respectivement 5 et 7 jobs initiaux. Ces derniers sont perturbés par l'arrivée de nouveaux jobs sur un horizon de 6 heures, T = 24 *ut* et α = 0,8. La fréquence d'apparition p_θ varie avec différentes valeurs (0,2 ; 0,5 ; 0,8 ; 1). Pour chaque valeur de p_θ, est associée un Nombre Total des Jobs Arrivés (*NTJA*) indiquant la taille de l'instance. Le minimum (Min), le maximum (Max), la moyenne (Moy) et l'écart type (ET) de la *DMI* et du *TR* sont présentés dans les Tableaux 37 et 38.

Tableau 37. *DMI* avec des durées de jobs suivant une loi normale

Jobs initiaux	p_θ (*NTJA*)	Min *DMI* (*s*)	Max *DMI* (*s*)	Moy *DMI* (*s*)	ET *DMI* (*s*)
5	0,2 (5 jobs)	0,12	0,26	0,17	0,05
	0,5 (12 jobs)	0,13	0,38	0,21	0,07
	0,8 (20 jobs)	0,21	1,14	0,44	0,31
	1 (24 jobs)	0,24	112	19,25	38,4
7	0,2 (5 jobs)	1,37	2,88	1,97	0,51
	0,5 (12 jobs)	1,58	3,66	2,03	0,69
	0,8 (20 jobs)	4,37	542	64,1	167
	1 (24 jobs)	403	6324	3606	2069

Tableau 38. *DMI* avec des durées de jobs suivant une loi uniforme

Jobs initiaux	p_θ (*NTJA*)	Min *DMI* (*s*)	Max *DMI* (*s*)	Moy *DMI* (*s*)	ET *DMI* (*s*)
5	0,2 (5 jobs)	0,19	0,65	0,467	0,16
	0,5 (12 jobs)	2,47	191	41	55
	0,8 (20 jobs)	-	-	-	-
7	0,2 (5 jobs)	0,68	1,37	1,04	0,22
	0,5 (12 jobs)	4,96	510	103	158
	0,8 (20 jobs)	-	-	-	-

Lorsque p_θ augmente, l'ensemble des jobs à réordonnancer augmente aussi. Comme pour le cas à machine unique, nous avons observé, dans ce cas, une augmentation de la moyenne de la *DMI*. Celle-ci augmente aussi lorsque l'ensemble des jobs initiaux augmente.

Dans un système de réordonnancement, un planning doit être établi après chaque perturbation et de préférence avant l'apparition de la perturbation suivante. Par conséquent, nous devons avoir un *DMI* inférieur à Δt. Si on considère un Δt égal à 15 minutes (900 s), et selon le Tableau 37, dans le cas où les durées de jobs suivent une loi normale, la PLNE est capable de résoudre des problèmes qui contiennent 7 jobs initiaux, perturbés par l'arrivée de nouveaux jobs avec une fréquence d'apparition p_θ allant jusqu'à 0,8. Au total, c'est équivalent en moyenne à 27 jobs sur la période $T = 24$ *ut*.

Dans le cas des durées de jobs qui suivent une loi uniforme discrète, le PLNE n'arrive pas à résoudre le problème lorsque p_θ excède 0,5. Comme les durées de jobs varient de 1 à 10 *ut*, elles sont plus grandes comparativement aux durées de la loi normale qui sont centrées sur 2,5. Ainsi, les machines consomment plus de temps à exécuter les jobs, d'où l'augmentation de la *DMI*.

Tableau 39. *TR* avec des durées de jobs suivant une loi normale

Jobs initiaux	p_θ (*NTJA*)	Min *TR* (*s*)	Max *TR* (*s*)	Moy *TR* (*s*)	ET *TR* (*s*)	Nombre moyen des jobs optimisés
5	0,2 (5 jobs)	0,27	0,51	0,35	0,07	0,6 jobs/itération
	0,5 (12 jobs)	0,42	0,69	0,51	0,07	0,9 jobs/itération
	0,8 (20 jobs)	1,01	2,31	1,57	0,4	1,9 jobs/itération
	1 (24 jobs)	2,39	460	65	142	6,6 jobs/itération
7	0,2 (5 jobs)	3,98	6,12	4,96	0,68	0,8 jobs/itération
	0,5 (12 jobs)	5,71	9,86	6,52	1,2	1,2 jobs/itération
	0,8 (20 jobs)	16,48	8012	829	2523	2,2 jobs/itération
	1 (24 jobs)	1231	20945	10622	6933	8,6 jobs/itération

Tableau 40. *TR* avec des durées de jobs suivant une loi uniforme

Jobs initiaux	p_θ (*NTJA*)	Min *TR* (*s*)	Max *TR* (*s*)	Moy *TR* (*s*)	ET *TR* (*s*)
5	0,2 (5 jobs)	0,42	2,66	1,461	0,66
	0,5 (12 jobs)	20,314	253	107,2704	73,25
	0,8 (20 jobs)	-	-	-	-
7	0,2 (5 jobs)	1,2	7,49	3,18	2,05
	0,5 (12 jobs)	25	1010	244	306
	0,8 (20 jobs)	-	-	-	-

Comme présenté dans le Tableau 39, le temps de résolution dépend de l'ensemble des jobs initiaux et de la fréquence d'apparitions des jobs. Si ceux-ci augmentent, la moyenne et l'écart type du *TR* augmentent, ce qui rend difficile l'estimation du temps nécessaire à la PLNE pour résoudre le problème.

Comme mentionné précédemment, l'ensemble de jobs concernés par le réordonnancement contient les jobs non-initiés par les machines et le nouveau job. Lorsqu'un job de longue durée occupe une machine et que des nouveaux jobs continuent d'arriver, plusieurs jobs non exécutés s'accumulent devant les machines, car celles-ci sont occupées. Cela explique l'incapacité de la PLNE à fournir des solutions en un temps raisonnable avec $p_\theta = 0{,}8$, lorsque les durées de jobs sont longues et uniformément distribuées, dans le Tableau 40. Ces tests montrent la limitation de la PLNE en termes de durées des jobs. En revanche, si ces durées tournent autour de 1 et 4 *ut*, les valeurs du *TR* se rapprochent des valeurs présentées dans le Tableau 39 avec une loi normale centrée sur 2,5.

Pour confirmer ces conclusions, le nombre moyen des jobs optimisés par itération est calculé pour les mêmes instances qui suivent la loi normale dans le Tableau 39, pour avoir une idée sur le nombre maximal de jobs qui peuvent être résolus par la PLNE dans chaque itération. Le nombre moyen de jobs optimisés par itération augmente lorsque l'ensemble des jobs initiaux et p_θ augmentent. Dans le dernier cas, 9 jobs, en moyenne, sont optimisés par itération. Cela rend difficile la résolution du problème en un temps raisonnable. La limitation du modèle basé sur la PLNE est clairement démontrée dans cette étude, ainsi que la nécessité d'opter pour des méthodes heuristiques pour améliorer la résolution en termes de nombre de jobs.

6. Conclusion

Dans ce chapitre, nous avons étudié une nouvelle mesure de performance combinant simultanément l'efficacité de l'ordonnancement représentée par le *TAMP* et sa stabilité évaluée par la *DMDFP* mais également le nombre de jobs qui changent de machine après le

réordonnancement, dans le cas de machines parallèles identiques. Cette combinaison de critères est très significative et utile dans les environnements hospitaliers, ainsi que dans les environnements industriels. Basé sur la stratégie prédictive-réactive, un modèle PLNE a été implémenté, ainsi qu'une méthodologie itérative qui permet de réordonnancer les jobs en réponse à des perturbations dues à l'arrivée de nouveaux jobs. Celle-ci est complétée par une optimisation lexicographique afin de combiner les deux critères de stabilité. Les résultats numériques ont permis d'en tirer les conclusions suivantes :

- Les critères de stabilité génèrent souvent un effet proactif lorsque α est compris entre 0,7 et 0,9, fournissant des meilleurs résultats, comparés à un monocritère qui ne considère que l'efficacité.
- En utilisant l'optimisation lexicographique, le nombre de jobs qui changent de machine se réduit, dans la plupart des cas à zéro, sans dégrader la fonction objectif principale.
- Le temps de résolution et la durée maximale d'itération dépendent en même temps de l'ensemble de jobs initiaux, de la fréquence d'apparition des nouveaux jobs, et des durées des jobs. Lorsque ceux-ci augmentent, le *TR* et la *DMI* deviennent excessifs.

Des tests ont été établi dans le cas des poids variables en utilisant la formule proposée dans le chapitre 2. Nous avons également constaté que l'augmentation du poids des jobs en fonction du temps aide les jobs qui ont un poids faible à être exécutés.

Ce travail est d'une grande utilité pour les preneurs de décision dans le domaine hospitalier ou industriel. Il permet de fournir, à chaque arrivée d'un nouveau job, un ordonnancement qui optimise à la fois l'efficacité et la stabilité dans un environnement de machines parallèles. Il a fait l'objet de plusieurs contributions :

- Un article publié dans une revue internationale [AP2] ;
- Un article publié dans une conférence internationale [CP1], en intégrant les poids dynamiques dans le problème ;
- Une application sur les cas des salles opératoires a été publiée dans une conférence internationale [CP4]. Ensuite, une version étendue, intégrant les poids dynamiques, a été soumise comme chapitre d'un livre [CL1].

Cependant, le modèle basé sur la PLNE ne permet de résoudre qu'un nombre limité de jobs. Dans le chapitre qui suit, nous nous intéressons à un problème de réordonnancement dans un système flowshop avec des contraintes de blocage mixtes et nous développons des heuristiques qui permettent de parcourir plus de jobs en un temps raisonnable.

Chapitre 4 : Optimisation de l'efficacité et de la stabilité dans un problème de réordonnancement : cas d'un système flowshop

Dans ce chapitre, nous traitons le problème de minimisation du temps d'attente moyen pondéré et de la déviation des dates de fin pondérées dans un problème de réordonnancement de type Flowshop avec des contraintes de blocage mixtes entre les machines. Après avoir fourni une description détaillée du problème, un modèle mathématique est proposé et ses résultats numériques sont discutés. Deux méthodes heuristiques basées sur la PLNE sont également proposées. Elles décomposent le problème de réordonnancement en sous problèmes, en divisant la partie concernée par le réordonnancement et en décalant cette partie. Ces heuristiques sont évaluées en variant la fréquence d'apparition des jobs, le nombre de jobs que contient la partie à réordonnancer, ainsi que le nombre de positions à décaler. Après l'analyse des résultats en termes de temps, et vu la complexité NP-difficile de ce type de problème, quatre autres heuristiques ont été conçues. La première réagit en insérant le nouveau job dans toutes les positions possibles. Les trois autres sont basées sur l'heuristique NEH. Enfin, une étude de sensibilité est menée pour comparer les heuristiques en termes de qualité des solutions et de temps de résolution.

1. Introduction

Le problème de réordonnancement dans les systèmes flowshop est un problème souvent rencontré dans les cas des usines de production où les machines sont disposées en série et les produits doivent passer dans le même ordre sur ces machines. Ce problème peut être aussi une illustration du cas des patients qui doivent passer dans le même ordre dans des unités d'un hôpital. Par exemple, l'unité d'attente préparatoire, le bloc opératoire et l'unité de soins post-anesthésique, comme décrit dans Wang *et al*. (2015), où les auteurs ont modélisé leur système comme étant un problème de réordonnancement dans un atelier de type flowshop sans autoriser la préemption. Ils ont considéré l'arrivée des patients urgents comme une perturbation.

Ce chapitre s'intéresse à un problème de réordonnancement dans un système flowshop qui subit des perturbations dues à l'arrivée et à l'annulation des jobs. La stratégie prédictive-réactive a été adoptée dans ce travail, consistant dans la phase prédictive à résoudre un problème d'ordonnancement statique, ayant comme objectif de minimiser le critère d'efficacité seul représenté par le temps d'attente moyen pondéré. À chaque arrivée d'une perturbation, intervient la phase réactive qui consiste à réordonnancer les jobs, avec cette fois comme objectif de minimiser les deux critères simultanément, le critère d'efficacité de l'ordonnancement, et le critère de stabilité représenté par la déviation moyenne des dates de fin pondérées par les poids des jobs. Ces deux critères sont associés dans la phase réactive par un coefficient d'efficacité-stabilité, noté α.

Les problèmes d'ordonnancement de type flowshop diffèrent aussi selon les technologies utilisées et les contraintes auxquelles le système est soumis. La plupart des travaux considèrent un espace de stockage illimité entre les machines. Or, dans les situations réelles, cet espace est limité, voire nul. Cette réalité crée une situation de blocage, c'est-à-dire que la ressource reste bloquée par un job tant que l'espace de stockage pour le déposer est insuffisant. Dans la littérature, peu de travaux ont considéré les contraintes de blocages dans les problèmes de réordonnancement (Kecman *et al.*, 2013 ; Tao et Liu, 2019). En revanche, dans les problèmes d'ordonnancement statique, plusieurs types de contraintes de blocages ont été introduits aux systèmes flowshop, à savoir les contraintes *RSb*, *RCb* et *RCb**. Dans ce chapitre, nous nous intéressons aux contraintes de blocage mixtes. Ce type de contrainte décrit la situation dans laquelle plusieurs types de contraintes peuvent être mixés dans un seul système flowshop. C'est-à-dire, un cas général dans lequel nous pouvons choisir le type de contrainte que nous souhaitons appliquer entre deux machines successives. Le cas sans blocage (*Wb*) peut, lui aussi, être combiné avec les autres contraintes de blocage. Cette nouvelle conception de problème de réordonnancement n'a pas été traitée auparavant dans la littérature. Les principales contributions de ce travail sont :

- L'intégration des contraintes de blocage mixtes dans un problème de réordonnancement d'un atelier de type flowshop.
- L'étude du *TAMP* combiné avec la *DMDFP* dans un problème de réordonnancement d'un système flowshop avec des contraintes de blocage mixtes.
- L'implantation d'un modèle basé sur la PLNE après l'avoir amélioré pour s'adapter à ce type de problème, et le développement d'une stratégie prédictive-réactive qui gère les perturbations dues à l'arrivée des nouveaux jobs et à l'annulation de jobs.

- Le développement des méthodes heuristiques adaptées pour ce problème, qui permettent de parcourir un nombre important de jobs.

2. Description du problème

Nous étudions un problème de réordonnancement dans un système flowshop qui subit des perturbations causées par l'arrivée et l'annulation des jobs. Ce problème est résolu en deux phases. La phase prédictive, qui commence avant l'apparition d'une perturbation, consiste à minimiser le *TAMP*. La phase réactive, qui commence après l'apparition d'une perturbation, consiste à minimiser simultanément le *TAMP* ainsi que la *DMDFP*. Le flowshop est composé d'un ensemble *M* de machines {1,2, …, *nm*} et à chaque job *j*, est associé un poids w_j, une date de disponibilité r_j, et une durée p_{jm} sur la machine *m*.

Phase prédictive

Dans cette première phase du problème, toutes les informations sont connues à l'avance. Quand le job *j* commence son exécution sur une machine *m* à la date ST_{jm}, il va procéder sur cette machine jusqu'à sa date de fin CT_{jm}, sans préemption. Le temps d'attente d'un job *j*, noté W_j, est la somme des périodes durant lesquelles le job *j* attend au sein du système avant son exécution sur les différentes machines, voir la Figure 19.

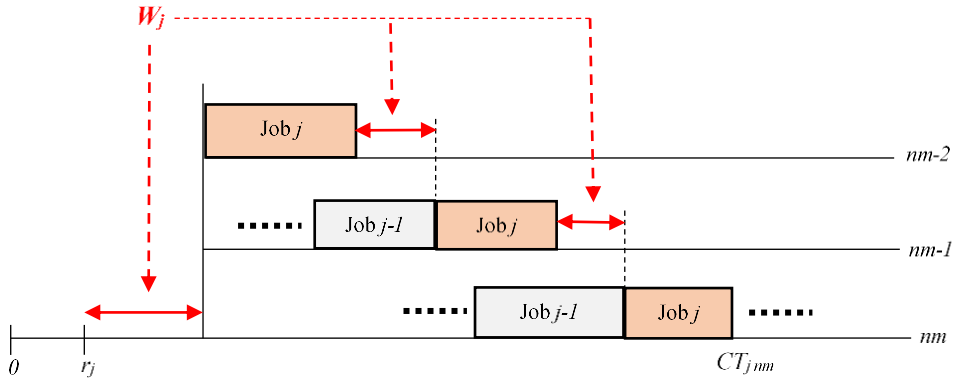

Figure 19. Temps d'attente du job *j* dans un système flowshop

Dans la Figure 19, les doubles flèches en rouge représentent la période totale durant laquelle le job *j* attend au sein du système avant son exécution qui correspond aussi à la somme des temps d'attente du job *j* devant chaque machine. Le temps d'attente W_j est alors calculé par $W_j = CT_{j\,nm} - r_j - \sum_{m=1}^{nm} p_{jm}$, avec $CT_{j\,nm}$ la date de fin du job *j* sur la dernière machine.

Il est supposé qu'un ensemble *N* de jobs {1,2, …, *n*} est prêt à être ordonnancé sur un ensemble *M* de machines {1,2, …, *nm*}. L'objectif est de déterminer la séquence qui va minimiser le critère d'efficacité *TAMP*, i.e. $min \sum_{j \in N} w_j W_j$. Ce problème peut être représenté dans la notation standard des problèmes d'ordonnancement par $F|r_j|\sum w_j W_j$.

Phase réactive

La phase réactive commence lorsqu'une perturbation arrive à l'instant *t*, l'ordonnancement établi doit être mis à jour pour répondre à cette nouvelle perturbation. Il s'agit de réordonnancer l'ensemble des jobs *N* qui combine les jobs de l'ensemble *No* avec le nouveau job *j'*. Comme décrit dans les chapitres 2 et 3, dans le cas de l'arrivée d'un nouveau job, l'ensemble des jobs

N est mis à jour tel que $N \leftarrow No \cup \{j'\}$. Sur la Figure 20, le job D arrive à l'instant t. Le job A, qui a commencé son exécution avant l'instant t est supprimé de la liste. Les jobs B et C constituent alors l'ensemble *No*, et vont être combinés avec le job D pour être réordonnancés.

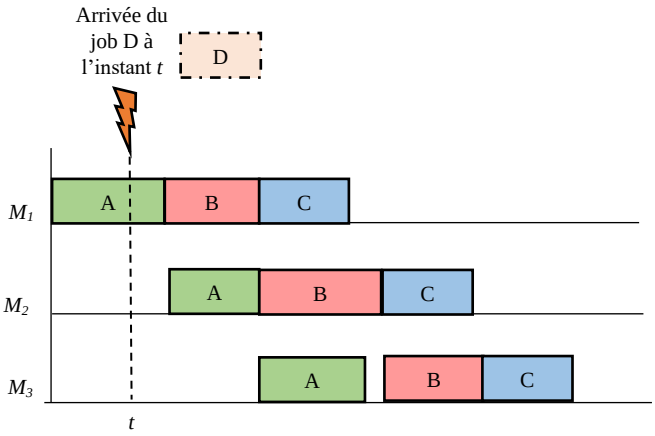

Figure 20. Arrivée du job D à l'instant *t*

Dans le cas d'annulation d'un job, l'ensemble N des jobs est mis à jour, tel que $N \leftarrow No$. Dans ce cas, *No* contient l'ensemble de jobs non encore débutés par la machine à l'instant de la perturbation, à l'exception du job annulé. Sur la Figure 21, le job B s'annule à l'instant t. Le job A, qui a commencé son exécution avant l'instant t est supprimé de la liste. Le job C constitue alors l'ensemble *No*, et va être le seul job à réordonnancer.

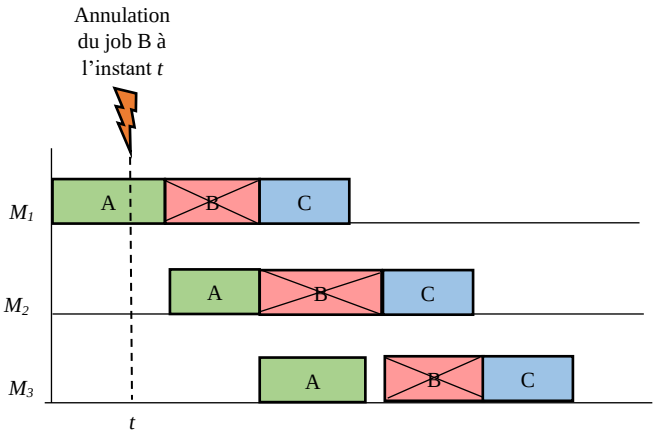

Figure 21. Annulation du job B à l'instant *t*

Le critère de stabilité : Le critère de stabilité pris en considération est toujours le *DMDFP*. Dans le cas d'un système flowshop, $D_{jm} = \max\{0, CT_{jm} - CTo_{jm}\}$ représente la déviation du job j sur la machine m, mesurée entre sa date de fin lorsqu'il est planifié pour la première fois CTo_{jm} et sa date de fin après le réordonnancement CT_{jm}. L'objectif sera de réduire l'instabilité du système à travers la minimisation de la *DMDFP*, $min\ \frac{1}{nm}\sum_{j\in No}\sum_{m\in M} w_j D_{jm}$. Le critère de stabilité n'évalue que les jobs qui appartiennent à l'ensemble *No*. D'autre part, nous avons divisé cette quantité par *nm*, le nombre de machines, pour la normaliser par rapport au critère d'efficacité.

Dans la phase réactive, après l'arrivée d'une perturbation, l'objectif est de minimiser la fonction suivante :

$$f_1 = \alpha \sum_{j \in N} w_j W_j + (1-\alpha) \frac{1}{nm} \sum_{j \in No} \sum_{m \in M} w_j D_{jm}$$

Les contraintes de blocage mixtes : La plupart des problèmes de réordonnancement des systèmes flowshop considèrent une capacité de stock illimitée entre les machines successives. C'est le cas d'un système flowshop classique sans blocage (*Wb*). Dans les cas réels, les capacités de stock entre les machines sont limitées, voire nulles. Cela entraine une situation de blocage, telle que : le blocage de type *RSb* qui exprime le cas où une machine est bloquée par un job jusqu'à ce que ce dernier commence sur la machine suivante, la contrainte de blocage spécifique *RCb** qui exprime le cas où une machine est bloquée par un job jusqu'à ce que ce dernier finisse sur la machine suivante, et la contrainte de blocage particulière *RCb* qui exprime le cas où une machine est bloquée par un job jusqu'à ce que ce dernier finisse et qu'il ait quitté la machine suivante, cette date correspond à la date de début du job sur la troisième machine. Dans ce travail, nous avons supposé que toutes les contraintes décrites puissent être mixées dans un seul système de production, avec n'importe quelle combinaison. Pour cela, un vecteur *V* est introduit pour décrire la séquence des contraintes entre les machines successives d'un problème. Par exemple, *V* = (*RCb*, *RSb*, *RCb**, *Wb*) est le vecteur de contraintes illustrées dans la Figure 22. Le vecteur *V* contient *nm*-1 éléments.

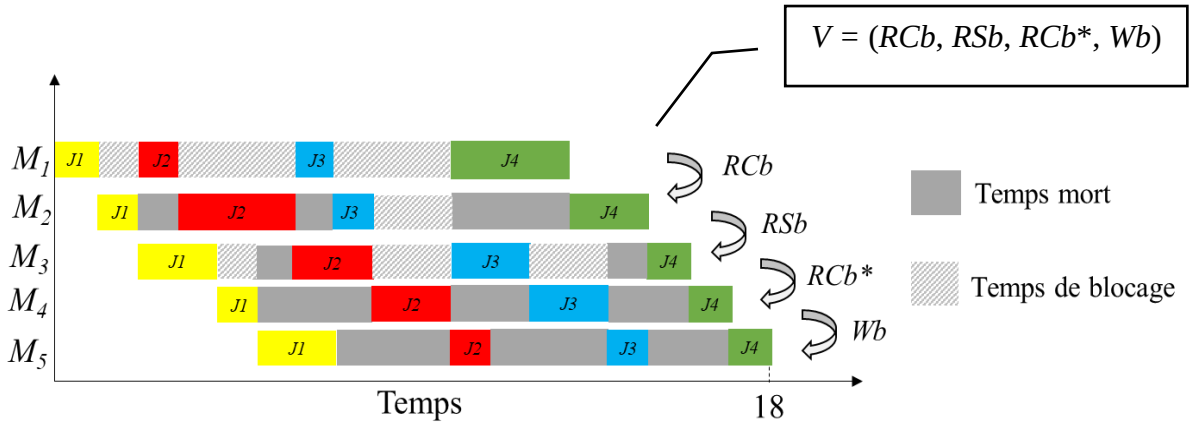

Figure 22. Contraintes de blocage mixtes avec le vecteur *V* = (*RCb*, *RSb*, *RCb, *Wb*)**

Ce nouveau problème est représenté dans la notation standard des problèmes d'ordonnancement par : $F|r_j, Mixte|\alpha \sum w_j W_j + (1-\alpha) \frac{1}{nm} \sum \sum w_j D_{jm}$.

3. Stratégie prédictive-réactive pour le problème F|r_j, *Mixte*|α $\Sigma w_j W_j$ + (1-α) 1/*nm* $\Sigma\Sigma w_j D_{jm}$

Comme décrit dans les chapitres précédents, la stratégie prédictive-réactive proposée consiste, dans la phase prédictive, à résoudre un problème classique d'ordonnancement ayant comme objectif la minimisation de *TAMP*, $min \sum_{j \in N} w_j W_j$. Puis, dans sa phase réactive, elle parcourt un horizon de simulation étape par étape, itérativement. À chaque étape, le processus

vérifie si une perturbation arrive. Si c'est le cas, l'ensemble N des jobs est mis à jour et un nouvel ordonnancement est établi. Dans cette deuxième phase, l'objectif est de minimiser la fonction f_1 qui combine l'efficacité et la stabilité de l'ordonnancement. Nous avons également défini en plus de $\theta(t)$, un paramètre binaire $\beta(t)$ tel que :

$$\beta(t) = \begin{cases} 1, & \text{Si un job s'annule à l'instant } t \\ 0, & \text{Sinon} \end{cases}$$

Comme déjà mentionné, nous supposons qu'un seul job arrive lorsque $\theta(t) = 1$ et qu'un seul job s'annule lorsque $\beta(t) = 1$. Ces deux évènements peuvent aussi arriver simultanément. Le logigramme de la Figure 23 décrit la stratégie prédictive-réactive proposée pour résoudre ce problème.

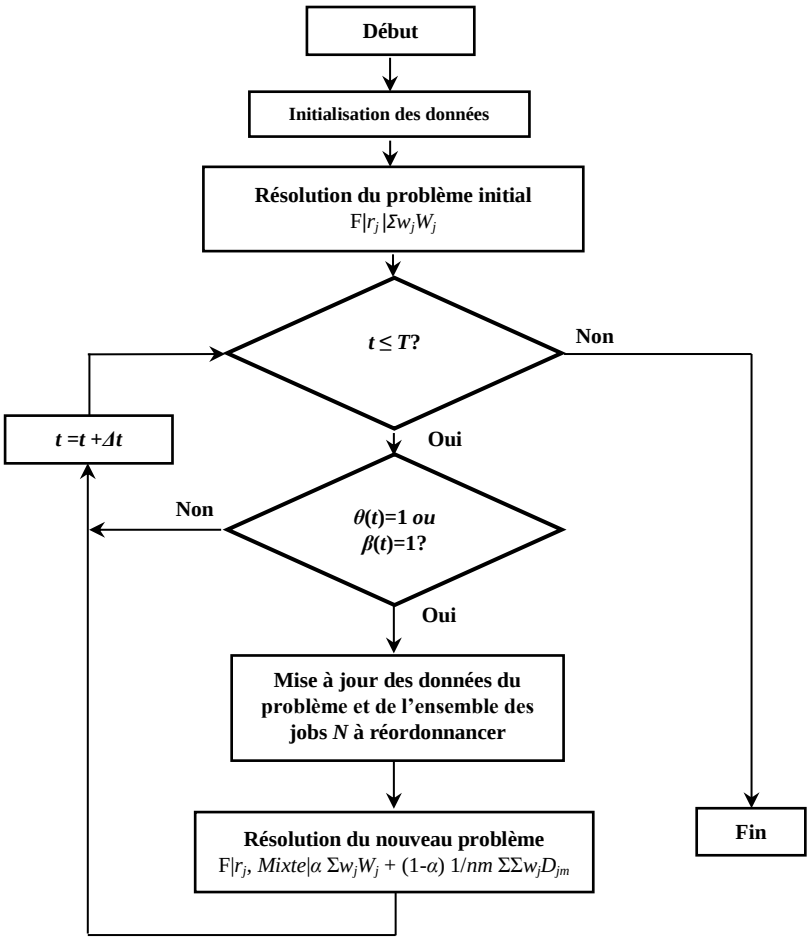

Figure 23. Stratégie prédictive-réactive pour le problème F|r_j, *Mixte*|α $\Sigma w_j W_j$ + (1-α) 1/nm $\Sigma\Sigma w_j D_{jm}$

4. Modèles mathématiques

Les modèles mathématiques avant et après la perturbation sont présentés respectivement dans les sections 4.1 et 4.2.

4.1 Modèle mathématique avant la perturbation

Le modèle mathématique proposé est une extension du modèle présenté dans le chapitre précédent au cas d'un système flowshop. Cette formulation a déjà été utilisée dans Trabelsi *et al*. (2012b), où les auteurs ont minimisé le Makespan dans un système flowshop avec des

contraintes de blocage mixtes. Pour correspondre au problème de la minimisation du *TAMP*, d'autres variables et paramètres ont été ajoutés, tel que les poids des jobs, les dates de disponibilité et le temps d'attente. Nous avons également indiqué en gras les données et variables modifiées ou nouvelles par rapport au modèle à plusieurs machines parallèles identiques. La formulation mathématique est décrite ci-dessous.

Ensembles et indices

N : ensemble de jobs $\{1,2, \ldots, n\}$

K : ensemble de positions $\{1,2, \ldots, n\}$

M : ensemble de machines $\{1,2, \ldots, nm\}$

j : indice des jobs, $j = 1,2, \ldots, n$

k : indice des positions, $k = 1,2, \ldots, n$

m : indice des machines, $i = 1,2, \ldots, nm$

Paramètres

w_j : poids du job j

r_j : date de disponibilité du job j

p_{jm} : durée du job j sur la machine m

$bigM$: grande valeur

$$\boldsymbol{B_{hm}} = \begin{cases} \mathbf{1}, & \textbf{Si il y a une contrainte de blocage de type } \boldsymbol{h} \textbf{ entre la machine } \boldsymbol{m} \textbf{ et } \boldsymbol{m+1} \\ \mathbf{0}, & \textbf{Sinon} \end{cases}$$

Avec : h = 1 s'il n'y a pas de contrainte de blocage entre la machine m et m+1.

h = 2 si il y a une contrainte de blocage de type *RSb* entre la machine m et m+1.

h = 3 si il y a une contrainte de blocage de type *RCb entre la machine m et m+1.**

h = 4 si il y a une contrainte de blocage de type *RCb* entre la machine m et m+1.

Variables de décision

$$\boldsymbol{x_{jk}} = \begin{cases} \mathbf{1}, & \textbf{Si le job } \boldsymbol{j} \textbf{ est affecté à la } \boldsymbol{k^{ème}} \textbf{ position} \\ \mathbf{0}, & \textbf{Sinon} \end{cases}$$

W_j : temps d'attente du job j

C_{km} : date de fin du job placé en la $k^{ème}$ position dans la machine m

S_{km} : date de début du job placé en la $k^{ème}$ position dans la machine m

CT_{jm} : date de fin du job j sur la machine m

Fonction objectif

$$min \sum_{j \in N} w_j W_j$$

Contraintes

$$\sum_{k\in N} x_{jk} = 1,\ \forall\, j\epsilon N \tag{3.1}$$

$$\sum_{j\in N} x_{jk} = 1,\ \forall\, k\epsilon K \tag{3.2}$$

$$C_{km} = S_{km} + \sum_{j\in N} p_{jm}x_{jk},\ \forall\, k\epsilon K,\ \forall\, m\epsilon M \tag{3.3}$$

$$S_{km} \geq C_{k\,m-1},\ \forall\, k\epsilon K,\ \forall\, m\epsilon\{2,\dots,nm\} \tag{3.4}$$

$$S_{km} \geq C_{k-1\,m}B_{1m} + S_{k-1\,m+1}B_{2m} + C_{k-1\,m+1}B_{3m} + S_{k-1\,m+2}B_{4m},\ \forall\, k\epsilon\{2,\dots,n\},\ \forall\, m\epsilon\{1,\dots,nm-2\} \tag{3.5}$$

$$S_{k\,nm-1} \geq C_{k-1\,nm-1}B_{1\,nm-1} + S_{k-1\,nm}B_{2\,nm-1} + C_{k-1\,nm}B_{3\,nm-1},\ \forall\, k\epsilon\{2,\dots,n\} \tag{3.6}$$

$$S_{k\,nm} \geq C_{k-1\,nm},\ \forall\, k\epsilon\{2,\dots,n\} \tag{3.7}$$

$$S_{km} \geq \sum_{j\in N} r_j x_{jk},\ \forall\, k\epsilon K, \forall\, m\epsilon M \tag{3.8}$$

$$CT_{jm} \geq C_{km} - bigM(1 - x_{jk}),\ \forall\, j\epsilon N,\ \forall\, k\epsilon K, \forall\, m\epsilon M \tag{3.9}$$

$$CT_{jm} \leq C_{km} + bigM(1 - x_{jk}),\ \forall\, j\epsilon N,\ \forall\, k\epsilon K, \forall\, m\epsilon M \tag{3.10}$$

$$W_j = CT_{j\,nm} - r_j - \sum_{m\in M} p_{jm},\ \forall\, j\epsilon N \tag{3.11}$$

$$x_{jk}\epsilon\{0,1\},\ \forall\, j\epsilon N, \forall\, k\epsilon K \tag{3.12}$$

$$S_{km}, C_{km}, CT_{jm}, W_j \geq 0,\ \forall\, j\epsilon N,\ \forall\, k\epsilon K, \forall\, m\epsilon M \tag{3.13}$$

Signification des contraintes

- Contrainte (3.1) : Elle permet de n'avoir qu'un seul job par position.
- Contrainte (3.2) : Elle permet de n'avoir qu'une seule position par job.
- Contrainte (3.3) : Elle impose que la date de fin du job en $k^{ème}$ position soit égale à sa date de début plus sa durée.
- Contrainte (3.4) : Elle impose que la date de début du job en $k^{ème}$ position soit supérieure ou égale à la date de fin de ce job sur la machine précédente.
- Contrainte (3.5) : Elle définit la date de début du job en $k^{ème}$ position, qui dépend du type de blocage déterminé par le paramètre B_{hm}.
- Contrainte (3.6) : Elle définit la date de début du job en $k^{ème}$ position dans l'avant dernière machine, où la contrainte *RCb* n'est pas possible car celle-ci dépend des deux machines suivantes.
- Contrainte (3.7) : Elle définit la date de début du job en $k^{ème}$ position dans la dernière machine, qui ne peut être que sans blocage.
- Contrainte (3.8) : Elle oblige la date de début du job en $k^{ème}$ position à être supérieure ou égale à sa date de disponibilité.

- Contraintes (3.9) et (3.10) : Si le job j est en $k^{ème}$ position, ces contraintes permettent d'imposer que la date de fin du job j soit égale à la date de fin du job en position k, avec *bigM* une valeur suffisamment grande.
- Contrainte (3.11) : Elle définit le temps d'attente du job j en fonction de sa date de fin sur la dernière machine, sa date de disponibilité et sa durée.
- Contrainte (3.12) : Elle impose que les variables x_{jk} soient binaires.
- Contraintes (3.13) : Elles sont des contraintes de non-négativité qui consistent à mettre toutes les variables supérieures ou égales à zéro.

4.2 Modèle mathématique après la perturbation

Ce second modèle est généré après l'apparition de chaque perturbation. L'ensemble des jobs est modifié de façon telle que :

No : L'ensemble des jobs existants qui n'ont pas encore commencé leur exécution, à l'exception du job annulé dans le cas d'annulation d'un job existant.

$N \leftarrow No \cup \{j'\}$: L'ensemble des jobs à réordonnancer dans le cas d'arrivée d'un nouveau job j'.

$N \leftarrow No$: L'ensemble des jobs à réordonnancer dans le cas d'annulation d'un job existant.

Nous avons également défini CTo_{jm}, la date de fin originale du job j sur la machine m, calculée lorsque ce job est ordonnancé pour la première fois. Les variables de décision x_{jk}, W_j, CT_{jm}, S_{km}, C_{km} sont aussi utilisées pour ce deuxième modèle, ainsi que D_{jm} la déviation de la date de fin du job j sur la machine m. Par conséquent, l'objectif est de minimiser la fonction f_1, telle que :

$$f_1 = \alpha \sum_{j \in N} w_j W_j + (1-\alpha) \frac{1}{nm} \sum_{j \in No} \sum_{m \in M} w_j D_{jm}$$

La minimisation de f_1 est soumise aux contraintes (3.1) jusqu'à (3.13), mais aussi deux contraintes additionnelles.

$$D_{jm} \geq CT_{jm} - CTo_{jm}, \ \forall j \in No, \forall m \in M \quad (3.14)$$

$$D_{jm} \geq 0, \ \forall j \in No, \ \forall m \in M \quad (3.15)$$

Les contraintes (3.14) et (3.15) permettent de calculer la valeur de la variable $D_{jm} = \max\{0, CT_{jm} - CTo_{jm}\}$.

5. Expérimentations numériques

L'algorithme basé sur la PLNE a été programmé sur FICO Xpress IVE sur un PC Core i7, 2.90 GHz, RAM 8 Go. Il est composé de deux parties. La première partie consiste à résoudre le problème d'ordonnancement avec l'objectif de minimiser le *TAMP*. La deuxième partie consiste à réordonnancer les jobs en réponse aux perturbations, avec l'objectif de minimiser le *TAMP* et la *DMDFP*, en adoptant la stratégie prédictive-réactive décrite dans le logigramme de la Figure 23. Dans la sous-section qui suit, nous décrivons la génération des instances.

5.1 Génération des instances

Le Tableau 41 présente les valeurs des paramètres utilisés pour le problème F|r_j, *Mixte*|α $\Sigma w_j W_j$ + (1-α) 1/nm $\Sigma\Sigma w_j D_{jm}$.

Tableau 41. Valeurs des paramètres utilisés pour F|r_j, *Mixte*|α $\Sigma w_j W_j$ + (1-α) 1/nm $\Sigma\Sigma w_j D_{jm}$

Paramètres	Valeurs
w_j	~ U(1, 5)
p_{jm}	~ U(1, 4) (*ut*)
$\theta(t)$	~ $B(p_\theta)$
$\beta(t)$	~ $B(p_\beta)$
T	48 (*ut*)
nm	5

Nous avons considéré un horizon de temps de 48 *ut*. Si on considère que 1 *ut* est équivalente à 10 minutes, alors la simulation correspond à un horizon de T = 48 *ut* = 480 *min* = 8*h*. Comme dans les deux chapitres précédents, les valeurs des poids varient uniformément entre 1 et 5. Les valeurs de la variable $\theta(t)$ (respectivement $\beta(t)$) sont générées aléatoirement suivant une loi de Bernoulli qui fournit, à chaque période *t*, une valeur 1 avec une probabilité p_θ (respectivement p_β) et 0 avec une probabilité 1-p_θ (respectivement 1-p_β). p_β représente la probabilité d'annulation d'un job à un instant *t*. Les tests ont été établis sur cinq machines disposées en série.

5.2 Etude du temps de résolution

Dans les tests numériques, nous avons étudié pour 10 instances différentes, le temps de résolution (*TR*) sous différentes fréquences d'apparition. Nous avons commencé par une petite fréquence d'apparition, valant p_θ = 0,2, puis une fréquence d'apparition moyenne p_θ = 0,5, et ensuite, une fréquence d'apparition élevée p_θ = 0,8. L'ensemble de jobs initiaux est fixé à 5 jobs et p_β = 0,1. Dans cette étude, nous avons considéré qu'il n'y a pas de contrainte de blocage entre les machines. Le minimum (Min), le maximum (Max), la moyenne (Moy), et l'écart type (ET) des temps de résolution en secondes sont présentés dans le Tableau 42.

Tableau 42. Temps de résolution avec différentes valeurs de p_θ et α

α	p_θ (*NTJA*)	Min *TR*	Max *TR*	Moy *TR*	ET *TR*
1	0,2 (10 *jobs*)	1,66	6,68	2,47	1,48
	0,5 (24 *jobs*)	45,82	1710	348,84	491,09
	0,8 (39 *jobs*)	9049	> 12h	14772	5303
0,75	0,2 (10 *jobs*)	1,81	5.47	2,31	1,11
	0,5 (24 *jobs*)	36,97	4198	850,8	1321
	0,8 (39 *jobs*)	10153	> 12h	16195	6720
0,5	0,2 (10 *jobs*)	1,81	7,01	2,49	1,59
	0,5 (24 *jobs*)	25,14	1188	374,1	387.43
	0,8 (39 *jobs*)	9014	> 12h	14062	5341

NTJA : Nombre Total de Jobs Arrivés

Le temps de résolution dépend de la fréquence d'apparition des jobs. Lorsque p_θ augmente, la moyenne et l'écart type du *TR* augmentent aussi, ce qui rend difficile d'estimer le temps nécessaire pour résoudre ce problème. En effet, lorsque p_θ excède 0,5, l'ensemble des jobs à réordonnancer augmente, ce qui rend la PLNE incapable de résoudre le problème en un temps raisonnable. Comme présenté dans le Tableau 43, lorsque p_θ = 0,8, la PLNE n'arrive pas à fournir constamment des solutions. Elle n'a fourni des solutions que pour six instances sur les dix testées. Pour le reste des instances, la simulation a été interrompue au bout de 12 heures de calcul.

La variation de p_β, la probabilité d'annulation de jobs, a aussi été étudiée pour évaluer son impact sur le *TR*. p_θ est dans ce cas fixée à 0,8, et différentes valeurs de p_β sont testées. Nous avons commencé par $p_\beta = 0{,}5$ ce qui représente une probabilité de 50% qu'un job s'annule à un instant *t*. $p_\beta = 0{,}3$ est ensuite testée, et finalement $p_\beta = 0{,}1$, ce dernier cas qui a déjà été testé lors de la variation de p_θ. Les résultats obtenus sont présentés dans le Tableau 43.

Tableau 43. Temps de résolution avec différentes valeurs de p_β et α

α	p_β (*NTJR*)	Min *TR*	Max *TR*	Moy *TR*	ET *TR*
1	0,5 (24 *jobs*)	32,94	42,42	37,99	3,01
	0,3 (15 *jobs*)	50,54	633,02	178,91	236,37
	0,1 (5 *jobs*)	9049	> 12h	14772	5303
0,75	0,5 (24 *jobs*)	33,07	44,81	40,17	3,52
	0,3 (15 *jobs*)	47,45	329,29	119,56	89,11
	0,1 (5 *jobs*)	10153	> 12h	16195	6720
0,5	0,5 (24 *jobs*)	25,03	45,89	36,74	6,43
	0,3 (15 *jobs*)	45,03	396,27	121,71	107,47
	0,1 (5 *jobs*)	9014	> 12h	14062	5341

NTJA : Nombre Total de Jobs Retirés

Lorsque p_β augmente, le nombre de jobs annulés augmente aussi. Même lorsqu'un nombre important de jobs arrive, lorsque p_β est grand, plusieurs jobs s'annulent en même temps. Cela diminue le nombre de jobs à réordonnancer à chaque itération, ce qui aide la PLNE à résoudre le problème en un temps inférieur. En revanche, lorsque p_β diminue, le nombre de jobs à réordonnancer augmente. Cela rend difficile la résolution du problème en un temps raisonnable. Cette observation confirme la conclusion déduite du Tableau 42.

Pour accélérer l'obtention des solutions, des méthodes heuristiques ont été développées, et sont présentées dans la section suivante.

6. Heuristiques

Dans cette section, des méthodes heuristiques sont proposées. La première sous-section présente deux heuristiques basées sur la PLNE. Ensuite, la deuxième sous-section présente quatre heuristiques basées sur la méthode NEH.

6.1 Heuristiques basées sur la PLNE

6.1.1 Description des heuristiques

Lorsqu'une perturbation arrive à l'instant *t*, tous les jobs qui ont commencé leur exécution avant cette date dans le système flowshop doivent garder les mêmes positions. L'ensemble de jobs est donc divisé en deux parties, une partie fixe qui contient les jobs déjà commencés par la machine, et une partie à réordonnancer qui contient le reste des jobs, combinés avec le nouveau job arrivant, voir la Figure 24.

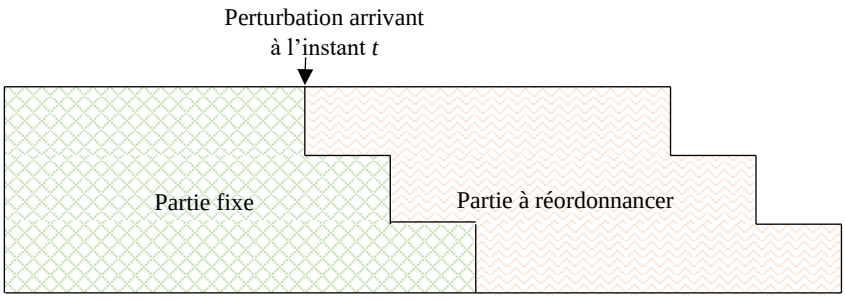

Figure 24. Système flowshop divisé en deux parties

Comme conclu dans les Tableaux 43 et 44, il est difficile de résoudre le problème en un temps raisonnable lorsque le nombre de jobs à réordonnancer est grand. Plus la partie du problème à réordonnancer est grande, plus le temps de résolution est important. Pour réduire ce temps de résolution, nous avons divisé la partie à réordonnancer en deux sous parties, de sorte à n'avoir dans la nouvelle partie à réordonnancer qu'un nombre limité de jobs, qui peut être résolue en un temps raisonnable.

Nous supposons que *Nr* est le nombre de jobs qui peuvent être résolus en un temps raisonnable par la PLNE. Ainsi, le système flowshop sera divisé en trois parties. L'ancienne partie fixe, la nouvelle partie fixe, et la partie à réordonnancer qui ne contient que *Nr* jobs incluant le nouveau job, voir la Figure 25.

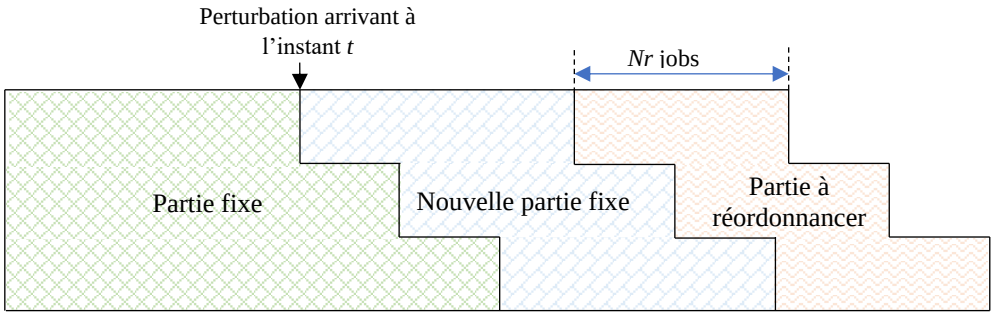

Figure 25. Système flowshop divisé en trois parties

Après avoir divisé le système flowshop en trois parties, nous résolvons le problème n'ayant que *Nr* jobs à réordonnancer. Ensuite, nous vérifions la position du nouveau job à l'intérieur de cette sous-partie. Si celui-ci est placé dans la première position à gauche, cela signifie qu'il est possible que ce job soit placé plus tôt dans un ordonnancement optimal. Dans ce cas, l'ensemble *Nr* est décalé vers la gauche d'une ou plusieurs positions et le problème est résolu une nouvelle fois, voir la Figure 26.

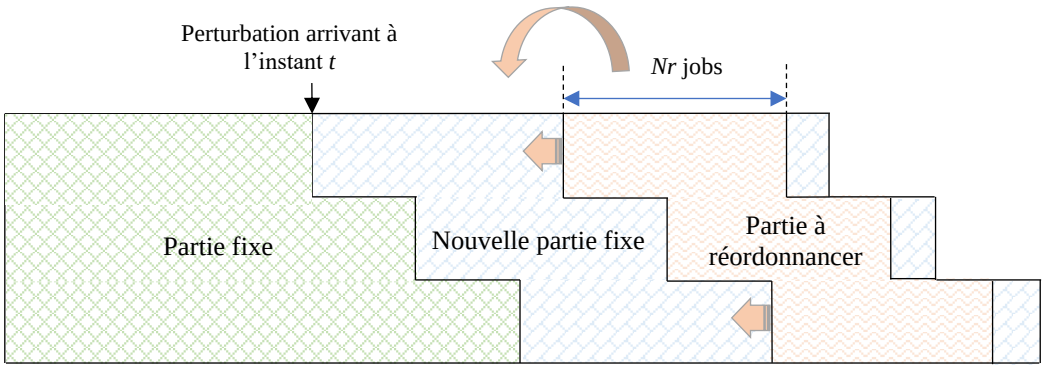

Figure 26. Système flowshop divisé avec *Nr* décalé

Après avoir décalé les *Nr* jobs, nous résolvons encore une fois le problème, puis nous revérifions la position du nouveau job. Si ce dernier reste toujours dans la première position à gauche, l'heuristique établit une amélioration en répétant le même processus jusqu'à ce que ce job soit placé dans une position autre que la première ou jusqu'à arriver à la date *t*. L'algorithme ci-joint décrit la méthodologie proposée.

Algorithme

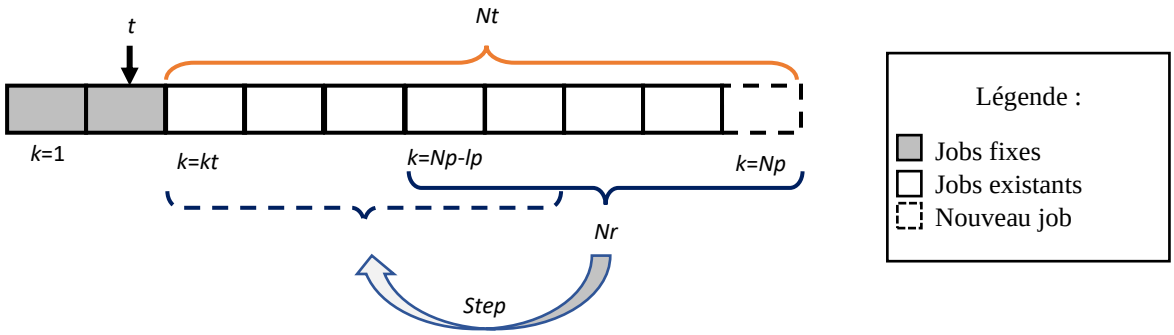

Figure 27. Illustration des paramètres utilisés

Données d'entrée :

- Ordonnancement optimal
- *new_job* : Nouveau job arrivé
- *Np* : Nombre total de positions, i.e. nombre de jobs déjà débutés et de jobs restant à ordonnancer
- *t* : Date de la perturbation
- *kt* : Position à partir de laquelle commence la partie à réordonnancer
- *Nt* : Nombre total de jobs à réordonnancer
- *Nr* : Nombre de jobs à réordonnancer à chaque étape d'amélioration
- *Step* : Nombre de positions à sauter
- *lp* : Parametre qui définit la position de départ des *Nr* jobs, *lp=Nr*-1
- *Xo(j,k)* : Variable qui affecte les jobs aux positions dans la séquence precèdente

Initialisation : $p = 0$, $Fin = 0$, *lp*, *step*, *Nr*

```
Si (Nt ⩽ Nr)
   Pour (j, k dans 1 jusqu'à Np-1) faire    ! le nouveau job n'est pas concerné
     Si (So(k, 1) ⩽ t)
       X(j,k) = Xo(j,k)                    ! garder la même position
     Fin-si
   Fin-pour
   Réordonnancer le nouveau problème.
   Afficher la solution.
Sinon
   Tant que (Np-lp-p ⩾ kt et Fin = 0)
     Pour (j, k dans 1 jusqu'à Np | k pas dans {Np-lp-p..Np-p}) faire
       X(j,k) = Xo(j,k)      ! garder la même position à l'exception des jobs dans Nr
     Fin-pour
     Réordonnancer le reste des jobs non fixés.
     Afficher la solution.
```

```
    Si (X(new_job, Np-lp-p) = 0 ou X(new_job, kt) = 1)
      Fin = 1
    Sinon
      Mémoriser les positions actuelles
      p = p + min(step, Np-lp-p-kt)
      Fin-si
  Fin-tant que
Fin-si
```

Deux positions de départ sont possibles pour choisir l'ensemble de *Nr* jobs, soit à partir de la gauche, soit à partir de la droite. Ainsi, nous avons nommé l'heuristique Hd1 la méthode qui parcourt les positions de la droite vers la gauche, comme décrit dans la Figure 26, et l'heuristique Hd2 celle qui traverse les positions de la gauche vers la droite.

6.1.2 Evaluation des heuristiques

Comme indiqué dans le Tableau 43, lorsque p_θ tend vers 0,8, la solution est difficilement obtenue en un temps raisonnable par la PLNE, en particulier lorsque p_β est petit. Pour ce cas particulier, $p_\beta = 0{,}1$, les heuristiques sont testées et évaluées, en termes de qualité de solution et de temps de résolution (*TR*).

Lorsqu'un nouveau job *j* est ordonnancé pour la première fois, sa date de fin originale CTo_{jm} est définie. Pour assurer une comparaison équitable, les heuristiques utilisent les dates originales fournies par la PLNE.

Les deux heuristiques sont, dans un premier temps, comparées avec la méthode basée sur la PLNE et *wSPT* lorsque $p_\theta = 0{,}2$ et $p_\theta = 0{,}5$. Par contre, comme pour les fréquences d'apparition plus grandes ($p_\theta = 0{,}8$ et $p_\theta = 1$), la PLNE n'arrive pas à résoudre le problème en un temps raisonnable. La méthode *wSPT*, qui est une méthode approchée pour ce problème, est utilisée pour générer un ordonnancement avec tous les jobs et obtenir ainsi les dates de fin originales pour comparer les deux heuristiques entre elles.

6.1.2.1 Variation de la fréquence d'apparition

La fréquence d'apparition p_θ est variée pour étudier son impact sur la qualité de solution et le temps de résolution. Quatre valeurs de p_θ (0,2; 0,5; 0,8; 1) et trois valeurs de α (0,5; 0,75; 1) sont testées. Nous avons aussi calculé le nombre d'améliorations (N° Aml) établi par l'heuristique avant de fournir la solution finale. Il est supposé que *Nr* = 5, *Step* = 4, et qu'il n'y a aucun blocage entre les machines. Dix instances différentes par problème sont testées, soit 10x4x3= 120 problèmes différents. Les moyennes sont présentées dans le Tableau 44.

Tableau 44. Comparaison des heuristiques pour différentes valeurs de p_θ

		p_θ = 0,2 (10 jobs)				p_θ = 0,5 (24 jobs)				p_θ = 0,8 (39 jobs)			p_θ = 1 (48 jobs)		
		Hd1	Hd2	*wSPT*	PLNE	Hd1	Hd2	*wSPT*	PLNE	Hd1	Hd2	*wSPT*	Hd1	Hd2	*wSPT*
	TR (*s*)	2,47	2,46	0.21	2,53	98,56	61,12	0.28	348,09	941	785	0,53	1954	2871	0,76
α=1	Solution	223	223	275.5	223	836,67	814,44	983.7	803,33	2657,11	2566,89	2590,89	3988,67	3648,67	3931,56
	N° Aml	0	0	-	-	15,00	12,29	-	-	45,67	51,00	-	58,56	69,78	-
	TR (*s*)	2,51	2,47	0.22	2,49	71,89	71.29	0.24	932,17	706	948	0,56	1545	2956	0,90
α=0,75	Solution	175,54	175,54	217.03	175,54	672,12	655,91	813.23	653,95	2374,03	2218,73	2221,46	3768,04	3187,79	3421,61
	N° Aml	0	0	-	-	13,89	13.67	-	-	41,33	52,56	-	56,44	72,00	-
	TR (*s*)	2,51	2,49	0.23	2,55	44,82	71,47	0.24	387,89	337	1082	0,52	750	4142	0,73
α=0,5	Solution	129,58	129.58	154.62	129.58	658,59	577,23	642.77	558,52	2335,80	1893,93	1852,02	3839,41	2745,88	2911,66
	N° Aml	0	0	-	-	13,44	16,44	-	-	34,78	55,78	-	47,44	72,25	-

Dans les faibles fréquences d'apparition (p_θ = 0,2), la méthode basée sur la PLNE et les heuristiques proposées fournissent les mêmes résultats en termes de temps de résolution et de qualité de solution. Cependant, les deux heuristiques ne font aucune amélioration pour fournir les solutions. D'autre part, *wSPT* fourni des solutions mauvaises comparée à la PLNE, Hd1 et Hd2.

Lors d'une fréquence d'apparition moyenne (p_θ = 0,5), les heuristiques fournissent des résultats en un temps plus court que la PLNE. Les solutions fournies par Hd2 sont meilleures que celles fournies par Hd1. Hd2 parcourt les positions de la gauche vers la droite. Les positions qui sont à gauche (au début) de l'ordonnancement, sont celles qui vont être exécutées plus tôt et celles qui ont un poids élevé. Le fait d'agir d'abord sur cette partie des jobs a un grand impact sur l'efficacité de la solution. D'où Hd2 est plus efficace que Hd1.

Pour des grandes fréquences d'apparition (p_θ = 0,8 et p_θ = 1), *wSPT* fournit rapidement des solutions. Cependant Hd2 fourni, dans la plupart des cas, des solutions meilleures que *wSPT*. Puisque cette dernière est utilisée, dans ces cas, comme une base pour obtenir les *CTo*, Hd1 n'arrive pas à fournir des solutions meilleures que *wSPT*. Hd2 reste toujours meilleure que Hd1.

Hd1 n'établit des améliorations que lorsque le nouveau job est placé dans la première position à gauche. Lorsque α diminue, la stabilité a plus d'importance. Alors, les positions qui sont à gauche seront occupées par les jobs existants pour ne pas trop changer la séquence précédente. Par conséquent, lorsque α diminue, Hd1 fait moins d'améliorations et consomme moins de temps pour fournir des solutions. En revanche, Hd2 fait plus d'améliorations et consomme plus de temps lorsque α diminue.

D'autre part, lorsque p_θ augmente, le nombre d'améliorations augmente aussi, car l'ordonnancement est face à plus de perturbations dues à l'arrivée de nouveaux jobs.

Nous avons observé aussi que les positions de certains jobs peuvent commuter après un réordonnancement. En effet, quand un nouveau job est inclus au sein d'une séquence, les jobs existants vont non seulement être décalés, mais leurs positions peuvent également être commutées. Pour illustrer ce phénomène, un exemple est présenté dans la section suivante.

6.1.2.2 Phénomène de commutation de jobs

Dans cet exemple, les durées p_{jm}, les dates de disponibilité r_j, et les poids w_j de 8 jobs sont présentés dans le Tableau 45. Les 5 premiers jobs sont des jobs initiaux pour lesquels toutes les

informations sont connues à l'avance. Ces jobs vont être résolus hors ligne. Ensuite, les jobs 6, 7, et 8 arrivent lors de l'exécution de l'ordonnancement initial, en ligne.

Tableau 45. Données du problème de réordonnancement dans un système flowshop

		Hors ligne					*En ligne*		
Job j		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
p_{jm}	M_1	3	5	2	5	2	2	1	3
	M_2	2	2	4	2	3	1	1	1
	M_3	3	2	3	4	1	1	2	5
r_j		0	1	0	1	2	2	3	4
w_j		2	1	3	2	1	2	1	4

La séquence optimale obtenue par la PLNE est présentée dans le Tableau 46. Nous avons supposé que $\alpha = 1$, afin de donner plus d'importance à l'efficacité et avoir moins de stabilité.

Tableau 46. Solution optimale fournie par le PLNE

	Séquence	Fonction objectif
Etape 0 : Séquence initiale optimale	3-1-4-5-2	36
Etape 1 : Séquence optimale après l'arrivée du job 6	3-6-1-4-5-2	54
Etape 2 : Séquence optimale après l'arrivée du job 7	3-6-1-7-4-5-2	66
Etape 3 : Séquence optimale après l'arrivée du job 8	3-6-8-1-**5**-4-**7**-2	104

Dans l'étape 3, le job 8 arrive avec un poids $w_8 = 4$. Ce poids élevé force la PLNE à placer le job 8 en troisième position, directement après les jobs 3 et 6. Cependant, les positions des jobs 5 et 7 ont commuté après le réordonnancement. D'où, la séquence optimale obtenue dans l'étape 3 est 3-6-8-1-5-4-7-2 avec une valeur de 104, au lieu de 3-6-8-1-7-4-5-2 qui a une valeur de 107. Pour illustrer cette commutation, les diagrammes de Gantt des étapes 2 et 3 sont présentés dans la Figure 28.

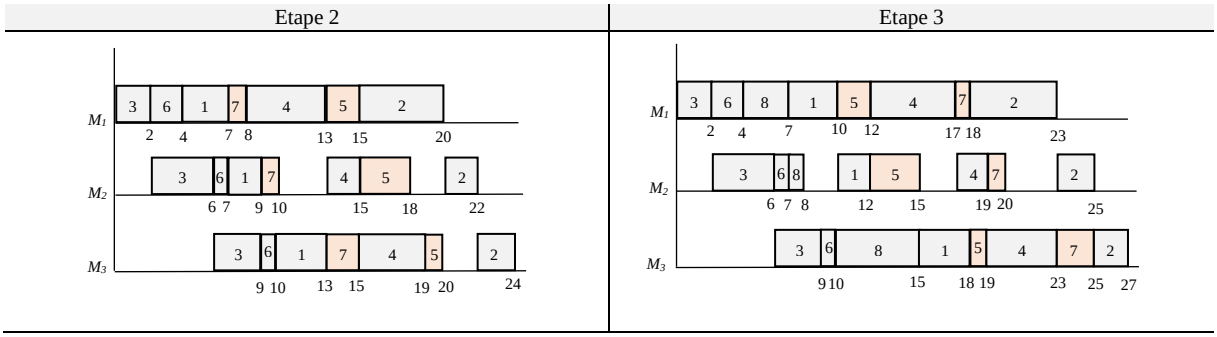

Figure 28. Diagrammes de Gantt des étapes 2 et 3

Comme observé dans la Figure 28, il est possible qu'un job se déplace vers l'avant après le réordonnancement, mais pas plus tôt que sa date de fin originale *CTo*. Par exemple, la date de fin du job 5 sur la machine 1, passe de $CT_{51} = 15$ à $CT_{51} = 12$. Ce phénomène est le résultat du changement des dates de disponibilité des machines après chaque perturbation due au réordonnancement. Cela est valable aussi dans le cas d'annulation de jobs.

En conclusion, le phénomène de commutation de jobs montre que les heuristiques proposées peuvent, dans certains cas, « rater » des solutions optimales en fixant une partie des jobs à réordonnancer.

6.1.2.3 Variation de *Nr* et *Step*

Dans ce paragraphe, les valeurs de *Nr* et *Step* sont variées pour évaluer leurs impacts sur la qualité de solution et le temps de résolution. Trois valeurs de α sont testées (0,5; 0,75; 1). Nous avons supposé que $p_\theta = 0{,}8$ pour avoir des fortes apparitions de jobs. *wSPT* est toujours utilisée pour comparer les heuristiques entre elles. 10 instances différentes pour chaque type de problème ont été testées. Les moyennes sont présentées dans le Tableau 47.

Tableau 47. Comparaison des heuristiques en fonction de *Nr* et *Step*

		***Nr* = 5**						***Nr* = 6**									***Nr* = 7**		
		***Step* = 3**			***Step* = 4**			***Step* = 3**			***Step* = 4**			***Step* = 5**			***Step* = 3**		
		Hd1	Hd2	*WSPT*	Hd1	Hd2	*WSPT*	Hd1	Hd2	*WSPT*	Hd1	Hd2	*WSPT*	Hd1	Hd2	*WSPT*	Hd1	Hd2	*WSPT*
	TR (s)	847	856	0,56	984	825	0,56	933	1011	0,56	865	649	0,56	720	593	0,56	1350	1070	0,56
α=1	Solution	2581	2469	2497	2573	2463	2497	2491	2490	2497	2479	2471	2497	2471	2468	2497	2468	**2440**	2497
	N° Aml	48,57	45,71	-	46,43	52,43	-	43,57	38,29	-	32,00	31,10	-	28,00	25,86	-	33,43	30,29	-
	TR (s)	814	1230	0,59	724	1019	0,59	987,23	1265	0,59	620	857	0,59	517	681	0,59	1246	1308	0,59
α=0,75	Solution	2324	2134	2133	2310	2132	2133	2266	2113	2133	2217	2107	2133	2214	2106	2133	2254	**2104**	2133
	N° Aml	45,86	57,29	-	42,71	54,29	-	31,57	38,71	-	28,00	31,14	-	23,86	28,00	-	22,29	31,29	-
	TR (s)	697	1580	0,54	345	1124	0,54	756	1883	0,54	443	1293	0,54	436	981	0,54	804	1898	0,54
α=0,5	Solution	2292	1835	1768	2275	1792	1768	2213	1789	1768	2099	1780	1768	2094	1778	1768	2098	**1771**	1768
	N° Aml	36,71	60,29	-	35,43	57,00	-	16,51	52,43	-	14,71	39,43	-	12,00	35,86	-	8,71	40,86	-

Nr représente le nombre de jobs à réordonnancer à chaque étape d'amélioration. Lorsque celui-ci augmente, la qualité de la solution augmente aussi dans la plupart des cas, car la PLNE prend en compte plus de jobs pour construire l'ordonnancement. En revanche, le temps de résolution augmente, même si les heuristiques font moins d'améliorations avant de fournir les solutions. D'autre part, Hd1 et Hd2 convergent vers les mêmes solutions.

Step représente le nombre de positions à sauter. Lorsque celui-ci augmente, les heuristiques arrivent à fournir des solutions en un temps plus court. La qualité de solution augmente aussi car le nouveau job sera comparé avec plus de jobs. Plus *Step* augmente, plus nous atteignons une solution meilleure en un temps plus court. Cependant, *Step* ne peut pas être supérieur ou égal à *Nr*, car si c'est le cas, le nouveau job ne sera pas inclus dans l'ensemble des jobs à réordonnancer. Le cas optimal est lorsque *Step* = *Nr*-1.

Les meilleures solutions sont obtenues par Hd2 lorsque *Nr* = 7. Cela confirme la supériorité de Hd2 sur Hd1. En revanche, *wSPT* qui est une méthode approchée pour ce problème manque aussi des solutions optimales, car elle ne tient pas compte de la stabilité pour construire les ordonnancements.

Nous avons également observé que le coefficient d'efficacité-stabilité α impacte le temps de résolution des heuristiques, comme nous l'avons déjà expliqué dans la section 6.1.2.1.

6.2 Heuristiques basées sur la méthode NEH

Les heuristiques proposées se basent sur la solution initiale pour insérer le nouveau job au sein de l'ordonnancement. Ces heuristiques parcourent l'horizon de simulation étape par étape. À chaque étape *t*, un job *j* arrive si $\theta(t) = 1$. La date *t* correspond à sa date de disponibilité r_j. Ce job est combiné avec l'ensemble des jobs non encore initiés par la machine *No* pour construire

l'ensemble *N* à réordonnancer. Ensuite intervient l'une des quatre heuristiques proposées. L'algorithme suivant décrit ce processus.

Pour t = 1 jusqu'à *T*
    Si ($\theta(t) = 1$)
        N = *No* + 1 *! Ajouter le nouveau job à l'ensemble No*
        r(*N*) = *t* *! Affecter la date t à la date de disponibilité du job*
        Mettre le nouveau job dans la dernière position
        Appliquer l'une des heuristiques proposées
    Fin Si
Fin pour

6.2.1 Description des heuristiques

Les heuristiques proposées sont basées sur l'heuristique NEH (Nawaz *et al*., 1983). Cette heuristique est parmi les heuristiques les plus utilisées pour les systèmes flowshop. Elle est non seulement efficace, mais aussi très simple à implémenter. Le pseudo code de la méthode NEH est rappelé ci-dessous.

- Classer les jobs par ordre décroissant des durées de jobs (Séquence initiale).
- Supprimer le premier élément de la liste classée et l'insérer comme premier élément de la séquence partielle.
- $k = 1$
- Faire
    - Prendre le premier élément de la liste classée.
    - Insérer ce job dans les k+1 positions possibles dans la séquence partielle actuelle.
    - Evaluer les k+1 séquences partielles.
    - Sélectionner la meilleure solution et la prendre comme nouvelle séquence partielle actuelle.
    - $k = k$+1
- Tant que ($k \leq N$)

6.2.1.1 Heuristique H1

Description de H1 : Nous considérons une liste de jobs séquencés dans l'ordre obtenu en résolvant la séquence précédente. Nous insérons le nouveau job dans toutes les positions possibles en commençant par la dernière position. La solution qui minimise la fonction objectif est retenue.

Algorithme H1 :

Best_objective = ∞
k = n'
Tant que (*k* > 0) *faire*
    Mettre le nouveau job dans la position *k* et décaler vers la droite le reste des jobs sans changer leurs positions.
    Objective = valeur de la fonction objectif pour cet ordonnancement
    Si (*Objective* < *Best_objective*)
        Sélectionner cette solution.

Best_objective = Objective
Fin Si
k = *k* - 1

Fin Tant que

Amélioration de H1 : Nous considérons la séquence obtenue par H1 et nous mettons le dernier job dans toutes les positions possibles. Nous sélectionnons la solution si elle est meilleure que la solution précédente. Nous répétons la même opération tant que la valeur de la fonction objectif s'améliore. Nous avons nommé cette heuristique H1*.

Algorithme de H1* :

Soit *S* la séquence obtenue par H1.
Best_objective = objectif obtenu avec *S.*
Objective = 0
Tant que (Objective ≤ *Best_objective*) *faire*
Appliquer H1 à partir de *S* en agissant sur le dernier job de la séquence (le dernier job sera inséré dans toutes les positions possibles), et conserver la meilleure séquence, appelée *S*1
Objective = objectif obtenu avec *S*1
Si (*Objective* < *Best_objective*)
S = *S*1
Best_objective = Objective
Fin Si

Fin Tant que

Exemple illustrant H1 et H1* : Dans le Tableau 48, les durées p_{jm}, les dates de disponibilité r_j et les poids w_j de 4 jobs sont présentés. Les 3 premiers jobs sont considérés comme des jobs initiaux pour lesquels toutes les informations sont connues. Ensuite, le job 4 arrive à l'instant *t* = 3, durant l'exécution de l'ordonnancement initial. Nous considérons qu'il n'y a pas blocage entre les machines, V = (*Wb*,*Wb*), et α = 0,5.

Tableau 48. Données du problème pour l'exemple illustratif

		Offline			*Online*
Job j		*1*	*2*	*3*	*4*
p_{jm}	M_1	3	1	5	1
	M_2	2	4	2	2
	M_3	3	2	4	3
r_j		0	1	0	3
w_j		2	1	3	5

La séquence initiale optimale est 1-2-3 avec f_1 = 15. Le Tableau 49 présente les solutions obtenues avec H1 et H1* après l'apparition du job 4.

Tableau 49. Solutions obtenues avec H1 et H1*

		H1		H1*	
		Séquence	*f1*	Séquence	*f1*
Première exécution	Itération 1	1-2-3-4	30	1-2-3-4	30
	Itération 2	1-2-4-3	27,5	1-2-4-3	27,5
	Itération 3	1-4-2-3	**19,83**	1-4-2-3	**19,83**
Amélioration 1	Solution initiale	-	-	1-4-2-3	19,83
	Itération 2	-	-	1-4-3-2	**18,5**
	Itération 3	-	-	1-3-4-2	32
Amélioration 2	Solution initiale	-	-	1-4-3-2	**18,5**
	Itération 2	-	-	1-4-2-3	19,83
	Itération 3	-	-	1-2-4-3	27,5
Meilleure solution		1-4-2-3	19,83	1-4-3-2	**18,5**
Temps de résolution (*s*)		0,046		0,069	

H1 place le job 4 dans toutes les positions possibles, sauf la première position puisque le job 1 a déjà commencé son exécution à l'instant $t = 3$. La meilleure solution obtenue par H1 est $f_1 = 19,83$. H1* améliore H1 en répétant la même opération tant que f_1 s'améliore et on obtient $f_1 = 18,5$. Cependant, le temps de résolution de H1* est plus grand que celui de H1.

6.2.1.2 Heuristique H2

Description de H2 : Nous considérons une liste de jobs séquencés par l'ordre obtenu en résolvant la séquence précédente (avant l'apparition du nouveau job). Nous plaçons temporairement le nouveau job dans la dernière position. Nous appliquons NEH à partir de cette liste. La solution qui minimise la fonction objectif est sélectionnée.

Amélioration de H2 : Nous considérons la séquence obtenue par H2 et nous appliquons récursivement l'algorithme NEH comme pour H1*, tant que la valeur de f_1 s'améliore. Nous avons nommé cette heuristique H2*.

Exemple illustrant H2 et H2* : Nous présentons une illustration de H2 et H2* du même exemple précèdent. Les résultats obtenus sont présentés dans le Tableau 50.

Tableau 50. Solutions obtenues avec H2 et H2*

		H2		H2*	
		Séquence	f_1	Séquence	f_1
Exécution de l'heuristique	Itération 1	1-2	1,5	1-2	1,5
	Itération 2	1-2-3	7,5	1-2-3	7,5
	Itération 3	1-3-2	9,5	1-3-2	9,5
	Itération 4	1-2-3-4	30	1-2-3-4	30
	Itération 5	1-2-4-3	27,5	1-2-4-3	27,5
	Itération 6	1-4-2-3	19,83	1-4-2-3	**19,83**
Amélioration 1	Itération 1	-	-	1-4	5
	Itération 2	-	-	1-4-2	8,33
	Itération 3	-	-	1-2-4	14
	Itération 4	-	-	1-4-2-3	19,83
	Itération 5	-	-	1-4-3-2	**18,5**
	Itération 6	-	-	1-3-4-2	32
Amélioration 2	Itération 1	-	-	1-4	5
	Itération 2	-	-	1-4-3	11
	Itération 3	-	-	1-3-4	23
	Itération 4	-	-	1-4-3-2	**18,5**
	Itération 5	-	-	1-4-2-3	19,83
	Itération 6	-	-	1-2-4-3	27,5
Meilleure solution		1-4-2-3	19,83	1-4-3-2	**18,5**
Temps de résolution (*s*)		0,047		0,078	

H2 considère la liste 1-2-3-4. A l'exception du job 1 qui a déjà commencé son exécution à l'instant $t = 3$, H2 place les jobs de la liste un par un dans la séquence partielle en appliquant NEH. La meilleure solution obtenue par H2 est $f_1 = 19,83$. H2* améliore la solution en répétant les mêmes opérations tant que f_1 s'améliore et on obtient $f_1 = 18,5$. Le temps de résolution de H2* est plus grand que celui de H2.

6.2.1.3 Heuristique H3

Description de H3 : Nous considérons une liste de jobs séquencés par l'ordre obtenu en résolvant la séquence précédente. Nous plaçons temporairement le nouveau job dans la première position. Nous appliquons NEH pour séquencer cette liste. La solution qui minimise la fonction objectif est sélectionnée.

Amélioration de H3 : Nous considérons la séquence obtenue par H3 et nous appliquons récursivement l'algorithme NEH comme pour H1*, tant que la valeur de f_1 s'améliore. Nous avons nommé cette heuristique H3*.

Exemple illustrant H3 et H3* : Nous présentons une illustration de H3 et H3* sur l'exemple précèdent. Les résultats obtenus sont présentés dans le Tableau 51.

Tableau 51. Solutions obtenues avec H3 et H3*

		H3		H3*	
		Séquence	f_1	Séquence	f_1
Exécution de l'heuristique	Itération 1	1-4	5	1-4	5
	Itération 2	1-4-2	8,33	1-4-2	8,33
	Itération 3	1-2-4	14	1-2-4	14
	Itération 4	1-4-2-3	19,83	1-4-2-3	19,83
	Itération 5	1-4-3-2	**18,5**	1-4-3-2	**18,5**
	Itération 6	1-3-4-2	32	1-3-4-2	32
Amélioration 1	Itération 1	-	-	1-4	5
	Itération 2	-	-	1-4-3	11
	Itération 3	-	-	1-3-4	23
	Itération 4	-	-	1-4-3-2	**18,5**
	Itération 5	-	-	1-4-2-3	19,83
	Itération 6	-	-	1-2-4-3	27,5
Meilleure solution		1-4-3-2	**18,5**	1-4-3-2	**18,5**
Temps de résolution (*s*)		0,047		0,063	

H3 considère la liste 1-4-2-3. A l'exception du job 1 qui a déjà commencé son exécution à l'instant t = 3, H3 place les jobs de la liste un par un dans la séquence partielle en appliquant NEH. La meilleure solution obtenue par H3 et H3* est f_1 = 18,5. Cependant, le temps de résolution de H3* est plus grand que celui de H3.

6.2.1.4 Heuristique H4

Description de H4 : Nous considérons une liste de jobs séquencés par ordre décroissant de w_j. Nous appliquons NEH pour séquencer cette liste. La solution qui minimise la fonction objectif est sélectionnée.

Amélioration de H4 : Nous considérons la séquence obtenue par H4 et nous appliquons récursivement l'algorithme NEH comme pour H1*, tant que la valeur de f_1 s'améliore. Nous avons nommé cette heuristique H4*.

Exemple illustrant H4 et H4* : Nous présentons une illustration de H4 et H4* sur l'exemple précèdent. Les résultats obtenus sont présentés dans le Tableau 52.

Tableau 52. Solutions obtenues avec H4 et H4*

		H4		H4*	
		Séquence	f_1	Séquence	f_1
Exécution de l'heuristique	Itération 1	1-4	5	1-4	5
	Itération 2	1-4-3	11	1-4-3	11
	Itération 3	1-3-4	23	1-3-4	23
	Itération 4	1-4-3-2	**18,5**	1-4-3-2	**18,5**
	Itération 5	1-4-2-3	19,83	1-4-2-3	19,83
	Itération 6	1-2-4-3	27,5	1-2-4-3	27,5
Amélioration 1	Itération 1	-	-	1-4	5
	Itération 2	-	-	1-4-3	11
	Itération 3	-	-	1-3-4	23
	Itération 4	-	-	1-4-3-2	**18,5**
	Itération 5	-	-	1-4-2-3	19,83
	Itération 6	-	-	1-2-4-3	27,5
Meilleure solution		1-4-3-2	**18,5**	1-4-3-2	**18,5**
Temps de résolution (*s*)		0,048		0,062	

H4 considère la liste 4-3-1-2 qui est séquencée par ordre décroissant des w_j. A l'exception du job 1 qui a déjà commencé son exécution à l'instant t = 3, H3 place les jobs de la liste un par

un dans la séquence partielle en appliquant NEH. La meilleure solution obtenue par H4 et H4* est f_1 = 18,5. Le temps de résolution de H4* est plus grand que celui de H4.

6.2.2 Evaluation des heuristiques

Dans cette étude, pour chaque instance, 5 jobs initiaux, pour lesquels toutes les informations sont connues à l'avance, vont être ordonnancés. Au cours de leur exécution, d'autres jobs arrivent, avec des fréquences d'apparition p_θ différentes 0,2, 0,5, 0,8 et 1. Trois valeurs de α sont testées : 0,5, 0,75, et 1. 10 instances par problème sont générées, soit 10x4x3= 120 problèmes différents. D'autre part, 3 vecteurs de contraintes sont étudiés pour analyser l'impact des contraintes de blocage sur la qualité de la solution et le temps de résolution.

6.2.2.1 Cas sans blocage *V*(*Wb, Wb, Wb, Wb*)

Dans ce paragraphe, le cas sans blocage *V* = (*Wb*, *Wb*, *Wb*, *Wb*) est considéré. Il représente un problème de réordonnancement classique. Deux études ont été menées. En premier lieu, une étude comparative des heuristiques avec leurs versions améliorées, pour quantifier l'écart dû à l'amélioration en termes d'efficacité et de temps de résolution. En deuxième lieu, les heuristiques sont comparées avec la méthode basée sur la PLNE.

a- Heuristiques versus heuristiques améliorées

Une comparaison des heuristiques (H) et de leurs versions améliorées (H*) a été effectuée. 10 instances différentes ont été générées suivant les paramètres présentés dans la section 5.1. Pour chaque heuristique, le nombre de fois où H* ≤ H (*NF*), le taux d'amélioration (*TA*), et le taux de différence de temps (*TDT*) ont été calculés. Les moyennes sont présentées dans le Tableau 53.

$$TA = \frac{\text{Solution fournie par H} - \text{Solution fournie par H}^*}{\text{Solution fournie par H}}$$

$$TDT = \frac{\text{Temps de résolution de H}^* - \text{Temps de résolution de H}}{\text{Temps de résolution de H}}$$

Tableau 53. Comparaison des heuristiques avec leurs versions améliorées

		p_θ = 0,2 (10 jobs)				p_θ = 0,5 (24 jobs)				p_θ = 0,8 (39 jobs)				p_θ = 1 (48 jobs)			
		H1*	H2*	H3*	H4*	H1*	H2*	H3*	H4*	H1*	H2*	H3*	H4*	H1*	H2*	H3*	H4*
	TA	0,009	0,001	0,027	0,000	-0,006	0,019	0,000	0,020	0,000	0,006	0,006	0,015	0,000	0,004	0,010	0,008
α=1	*TDT*	0,174	0,369	0,445	0,509	3,138	1,073	1,092	1,081	14,055	1,350	1,229	1,374	22,674	1,510	1,153	1,198
	NF	10	10	10	10	7	8	8	10	7	7	9	9	8	8	9	9
	TA	0,005	0,000	0,000	0,000	0,003	0,006	0,003	0,007	-0,007	0,018	0,000	0,007	-0,007	-0,004	0,008	0,005
α=0,75	*TDT*	0,251	0,310	0,333	0,328	3,295	1,013	1,136	1,141	12,772	1,154	1,221	1,498	22,332	1,321	1,393	1,137
	NF	10	10	10	10	8	9	8	10	6	9	7	6	4	2	8	5
	TA	0,005	0,009	0,004	-0,028	0,078	-0,003	0,004	0,011	0,030	0,008	0,009	0,006	0,026	0,001	0,013	0,012
α=0,5	*TDT*	0,244	0,235	0,370	0,178	3,123	1,035	1,047	1,007	12,613	1,087	1,267	1,410	22,312	1,419	1,277	1,443
	NF	7	9	9	5	9	5	8	6	8	8	7	6	8	6	6	7

Le taux d'amélioration (*TA*) est positif dans la majorité des cas. Cela prouve l'efficacité des versions améliorées des heuristiques. Cependant, pour des cas particuliers, le *TA* peut être négatif. Lorsqu'une amélioration est effectuée, les solutions fournies par H et H* dans une étape donnée sont différentes puisque H* est une version améliorée de H. Dans l'étape suivante, la stratégie consiste à retirer les jobs débutés sur une machine. Si l'ensemble des jobs retirés n'est pas le même, le problème à résoudre ne sera pas le même non plus. D'où la possibilité d'avoir un résultat final diffèrent, et parfois en faveur de H, la version non améliorée. La possibilité d'avoir ce cas particulier est relativement faible car le nombre de fois où H* ≤ H (*NF*) est en moyenne de 7,8.

En revanche, le *TDT* est toujours positif puisque les versions améliorées consistent à répéter plusieurs fois les mêmes opérations des heuristiques tant que la solution s'améliore. L'efficacité des versions améliorées nous a conduit à les utiliser pour le reste des résultats numériques.

b- Comparaison des méthodes de résolution

Dans les Tableaux 54 et 55, nous comparons, avec 50 instances différentes, les 4 versions améliorées des heuristiques avec la méthode basée sur la PLNE, en termes de qualité de solution et de temps de résolution.

Le pourcentage d'erreur est la différence entre la meilleure solution et la solution fournie, en pourcentage de la meilleure solution.

$$pourcentage\ d'erreur = \frac{\text{solution fournie} - \text{meilleure solution}}{\text{meilleure solution}} * 100$$

Tableau 54. Comparaison des méthodes de résolution avec *V* = (*Wb, Wb, Wb, Wb*)

		p_θ = 0,2 (10 jobs)					p_θ = 0,5 (24 jobs)					p_θ = 0,8 (39 jobs)					p_θ = 1 (48 jobs)				
		H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE
	f_1	**228,5**	**228,5**	228,6	**228,5**	**228,5**	882,7	**842,4**	857,55	**842,4**	878,5	2481	2397,2	2406	**2360**	-	4051,6	3900,7	3861,5	**3844**	-
α=1	*DMI* (*s*)	0,02	0,02	0,02	0,02	0,25	0,10	0,09	0,09	0,10	53,11	0,83	0,71	0,71	0,66	-	1,50	1,45	1,23	1,28	-
	TR (*s*)	0,27	0,27	0,29	0,30	1,60	1,15	1,27	1,25	1,29	144,76	9,50	8,59	8,64	8,88	-	22,44	21,97	20,25	20,46	-
	f_1	181,9	**181,8**	181,9	**181,8**	**181,8**	731,4	**704,9**	709,3	713,9	723,1	2164,7	**2002,4**	2037,3	2031,1	-	3532,3	**3217,4**	3303,8	3296,6	-
α=0,75	*DMI* (*s*)	0,02	0,02	0,02	0,02	0,27	0,09	0,09	0,09	0,10	497,57	0,68	0,64	0,66	0,68	-	1,38	1,41	1,44	1,52	-
	TR (*s*)	0,25	0,27	0,27	0,27	1,64	1,22	1,24	1,24	1,23	787,78	8,35	8,14	8,46	9,12	-	21,00	20,69	22,01	19,93	-
	f_1	134,1	**134,0**	134,1	**134,0**	**134,0**	539,6	**493,2**	513,8	509	497,0	1694	**1359,2**	1425	1430,2	-	2754,5	**2093,2**	2271,1	2254,7	-
α=0,5	*DMI* (*s*)	0,02	0,02	0,02	0,02	0,23	0,10	0,09	0,09	0,12	185,11	0,65	0,61	0,68	0,65	-	1,41	1,38	1,37	1,48	-
	TR (*s*)	0,26	0,27	0,28	0,27	1,59	1,22	1,27	1,21	1,24	370,73	8,67	8,14	9,01	8,74	-	22,17	20,50	20,68	22,62	-

Tableau 55. Pourcentage d'erreur et écart type entre la solution fournie et la meilleure solution dans le cas de *V* = (*Wb, Wb, Wb, Wb*)

		p_θ = 0,2 (10 jobs)					p_θ = 0,5 (24 jobs)					p_θ = 0,8 (39 jobs)					p_θ = 1 (48 jobs)				
		H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE
	Pourcentage d'erreur	**0,00**	**0,00**	0,01	**0,00**	**0,00**	5,28	**0,62**	2,34	**0,62**	5,01	5,44	1,89	2,27	**0,31**	-	5,82	1,91	0,86	**0,44**	-
α=1	Ecart type	0,00	0,00	0,07	0,00	0,00	4,20	1,70	2,88	1,70	3,27	2,39	1,26	4,10	0,74	-	2,00	1,73	0,90	1,01	-
	N° des meilleures solutions	50,00	50,00	47,00	50,00	50,00	5,00	40,00	10,00	40,00	3,00	5,00	4,00	20,00	35,00	-	0,00	4,00	15,00	35,00	-
	Pourcentage d'erreur	0,02	**0,00**	0,03	**0,00**	**0,00**	5,93	**2,39**	2,89	3,47	4,91	8,79	**0,67**	2,38	2,08	-	9,92	**0,21**	2,83	2,63	-
α=0,75	Ecart type	0,10	0,00	0,11	0,00	0,00	4,22	4,65	3,40	3,90	4,11	2,38	1,31	3,93	4,03	-	3,06	0,60	2,19	2,46	-
	N° des meilleures solutions	41,00	50,00	45,00	50,00	50,00	10,00	20,00	10,00	20,00	2,00	0,00	35,00	5,00	10,00	-	0,00	43,00	2,00	5,00	-
	Pourcentage d'erreur	0,11	**0,00**	0,02	**0,00**	**0,00**	11,39	**1,89**	6,12	5,14	2,70	24,86	**0,09**	5,07	5,35	-	31,61	**0,01**	8,50	7,71	-
α=0,5	Ecart type	0,33	0,00	0,15	0,00	0,00	7,68	2,11	5,28	4,23	2,59	3,51	0,28	3,69	5,97	-	4,15	0,07	3,36	3,79	-
	N° des meilleures solutions	41,00	50,00	48,00	50,00	50,00	3,00	20,00	10,00	15,00	10,00	0,00	45,00	0,00	5,00	-	0,00	48,00	2,00	0,00	-

Comme observé dans les deux tableaux, lorsque p_θ augmente, le nombre de jobs qui arrivent à chaque étape de réordonnancement augmente. Cela rend la PLNE incapable de fournir des solutions en un temps raisonnable. Cependant, les heuristiques arrivent à fournir des solutions lorsque p_θ est au-dessus de 0,5. La *DMI* est la durée maximale que consomme la méthode dans une étape donnée avant de fournir une solution. Elle mesure la période entre l'arrivée d'un job et l'établissement de l'ordonnancement. Dans le Tableau 54, la *DMI* de la PLNE devient trop grande lorsque p_θ dépasse 0,5.

Lorsque p_θ = 0,2, comme il y a moins de perturbations, les heuristiques et la PLNE fournissent des solutions proches. Dans ce cas, le pourcentage d'erreur et l'écart type sont faibles.

Lorsque p_θ = 0,5, la PLNE consomme plus de temps que les heuristiques pour fournir des solutions. Cependant, dans certains cas, les heuristiques fournissent des solutions meilleures que la méthode basée sur la PLNE. Comme déjà mentionné précédemment, dans une étape donnée, la solution fournie par les heuristiques et la PLNE peuvent être différentes. Dans l'étape suivante, la stratégie consiste à retirer les jobs initiés par la machine. Si, l'ensemble de jobs retirés n'est pas le même, le problème à résoudre ne sera pas le même non plus. D'où la possibilité d'avoir un résultat final diffèrent, et parfois en faveur des heuristiques.

Lorsque p_θ = 0,8 et 1, la méthode basée sur la PLNE n'arrive pas à fournir une solution en un temps raisonnable. La simulation a été interrompue au bout de 12 heures de calcul. En revanche, les heuristiques arrivent à fournir rapidement des solutions. En général, les quatre heuristiques sont proches l'une de l'autre en termes de temps de résolution. Ce temps de résolution est inférieur à la minute sur toutes les instances testées.

H4* considère une liste de jobs séquencés par l'ordre décroissant de w_j et utilise la méthode NEH pour ordonnancer cette liste de jobs. Cette heuristique fournit des meilleures solutions lorsque α =1. Puisque la stabilité de l'ordonnancement n'est pas prise en compte dans ce cas, les poids des jobs ont beaucoup d'influence.

H2* considère une liste de jobs séquencés par l'ordre obtenu en résolvant la séquence précédente et place temporairement le nouveau job dans la dernière position. Elle utilise ensuite la méthode NEH pour ordonnancer cette liste de jobs. Cette heuristique fournit des meilleures solutions lorsque α = 0,75. Comme la stabilité est prise en compte, dans ce cas, l'ordre de la séquence précédente est souvent maintenu et le nouveau job est souvent placé dans la dernière position, en fonction de son poids. Cette situation est le principe même de l'heuristique H2*, ce qui explique sa supériorité dans ce cas.

Lorsque α = 0,5, la stabilité de l'ordonnancement a plus d'importance. D'après le Tableau 56, le nombre de fois où H2* fournit la meilleure solution augmente. Cela confirme la supériorité de H2* lorsque la stabilité est prise en compte.

H1* place le nouveau job dans toutes les positions possibles, sans utiliser la méthode NEH. Dans la plupart des cas, elle fournit de mauvaises solutions comparées à H2*, H3*, et H4*, puisque celles-ci sont des versions améliorées de H1*.

En conclusion, l'un des meilleurs compromis qu'un décideur peut choisir, c'est d'utiliser, pour chaque cas, une heuristique parmi celles proposées. Idéalement, H4* lorsque α = 1, et H2* lorsque α = 0,5 ou α = 0,75.

6.2.2.2 Cas de V = (*Wb, RSb, RCb*, RCb*)

Le vecteur V = (*Wb, RSb, RCb*, RCb*) est étudié dans ce paragraphe. Les moyennes des résultats des 50 instances sont présentées dans les Tableaux 56 et 57.

Lorsque les contraintes de blocage sont mixées dans un système de production, l'espace des solutions réalisables se réduit, puisqu'il y a beaucoup de contraintes à satisfaire en même temps. Comme observé dans le Tableau 56, le *TR* et la *DMI* deviennent plus grands, comparés au cas sans blocage. La méthode basée sur la PLNE peut, dans ce cas, difficilement fournir des solutions lorsque p_θ = 0,5.

La méthode basée sur la PLNE et les heuristiques fournissent toujours des résultats proches les unes des autres lorsque p_θ = 0,2. Les résultats divergent lorsque p_θ > 0,2. D'après le Tableau 57, nous observons toujours une supériorité de H4* lorsque α = 1, et H2* lorsque α = 0,75 et α = 0,5. Ainsi, l'interprétation établie dans la section 6.2.2.1 à propos de l'impact de α sur la performance des solutions reste valable.

Cependant, une diminution du pourcentage d'erreur et de l'écart type est observée dans le Tableau 57 comparée au Tableau 55. Comme les contraintes de blocage mixtes sont utilisées dans cette étude, l'espace des solutions réalisables diminue. Dans ce cas, les méthodes ont moins de possibilité d'établir des solutions différentes. Les heuristiques H2*, H3* et H4* convergent vers les mêmes solutions, comparées au cas sans blocage. L'heuristique H1*, qui n'utilise pas la méthode NEH, a toujours un pourcentage d'erreur élevé.

La contrainte *RCb** décrit le cas où une machine reste bloquée par un job, jusqu'à ce que celui-ci finisse son exécution sur la machine suivante. Selon Sauvey *et al*. (2020), cette contrainte de blocage relie deux machines autour du même job, puisqu'elle considère l'opération suivante pour ordonnancer l'opération en cours. Alors, nous considérons dans la section suivante, la contrainte *RCb** introduite entre la première et la deuxième machine, suivie de deux *Wb* successives. Nous évaluons ensuite, l'impact de cette situation sur le temps de résolution et le pourcentage d'erreur des méthodes proposées.

Tableau 56. Comparaison des méthodes de résolution avec *V* = (*Wb, RSb, RCb*, RCb*)

		p_θ = 0,2 (10 jobs)					p_θ = 0,5 (24 jobs)					p_θ = 0,8 (39 jobs)					p_θ = 1 (48 jobs)				
		H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE
α=1	f_1	614,7	614,7	614,7	**612,0**	**612,0**	2341,2	**2311,4**	2337,4	**2311,4**	2325,3	5628,4	5610,1	5626,1	**5604,8**	-	8476,8	8353,6	8355,9	**8338,3**	-
	DMI (*s*)	0,02	0,02	0,02	0,02	0,24	0,12	0,11	0,11	0,13	1647,66	0,93	0,83	0,87	0,87	-	1,37	1,14	1,19	1,18	-
	TR (*s*)	0,24	0,24	0,24	0,23	1,68	1,48	1,38	1,43	1,45	4138,47	11,95	11,12	10,98	11,74	-	20,21	18,33	18,41	19,35	-
α=0,75	f_1	469,7	**467,1**	467,5	467,5	467,4	1842,7	**1836,9**	1839,5	1843,1	1843,8	4707,1	**4636,8**	4640,3	4642,8	-	7036,8	**6877,3**	6877,4	6886,5	-
	DMI (*s*)	0,02	0,02	0,02	0,02	0,23	0,12	0,11	0,12	0,14	593,00	0,86	0,79	0,83	0,86	-	1,22	1,16	1,21	1,20	-
	TR (*s*)	0,25	0,25	0,25	0,24	1,58	1,45	1,46	1,52	1,58	1697,34	11,01	10,64	11,06	11,54	-	18,93	18,24	18,70	18,83	-
α=0,5	f_1	320,6	**319,0**	319,3	319,3	319,8	1379,1	**1301,8**	1312,3	1310,3	1308,8	3553,7	**3246,1**	3283,9	3312,6	-	5551,8	**4803,3**	4849,5	4824,3	-
	DMI (*s*)	0,02	0,02	0,02	0,02	0,23	0,10	0,11	0,11	0,10	1071,68	0,85	0,89	0,87	0,87	-	1,27	1,19	1,17	1,19	-
	TR (*s*)	0,26	0,24	0,24	0,24	32,78	1,35	1,35	1,37	1,44	1764,43	11,67	11,26	11,31	11,23	-	20,47	18,71	19,12	18,55	-

Tableau 57. Pourcentage d'erreur et écart type entre la solution fournie et la meilleure solution dans le cas de *V* = (*Wb, RSb, RCb*, RCb*)

		p_θ = 0,2 (10 jobs)					p_θ = 0,5 (24 jobs)					p_θ = 0,8 (39 jobs)					p_θ = 1 (48 jobs)				
		H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE
α=1	Pourcentage d'erreur	0,47	0,47	0,47	**0,00**	**0,00**	1,36	**0,03**	1,18	**0,03**	0,65	0,53	0,23	0,49	**0,14**	-	1,61	0,19	0,23	**0,01**	-
	Ecart type	1,48	1,48	1,48	0,00	0,00	1,12	0,05	2,19	0,05	0,86	1,19	0,33	0,63	0,23	-	1,89	0,23	0,49	0,03	-
	N° des meilleures solutions	45	45	45	50	50	10	35	20	35	20	15	25	25	30	-	0	25	35	40	-
α=0,75	Pourcentage d'erreur	0,57	**0,00**	0,09	0,09	0,08	0,58	**0,22**	0,38	0,59	0,61	1,71	**0,23**	0,30	0,35	-	2,37	**0,07**	0,07	0,20	-
	Ecart type	0,84	0,00	0,30	0,30	0,24	1,14	0,53	0,63	0,81	0,89	1,39	0,34	0,40	0,37	-	1,04	0,15	0,08	0,21	-
	N° des meilleures solutions	30	50	45	45	45	20	30	25	20	25	15	25	5	5	-	0	40	20	15	-
α=0,5	Pourcentage d'erreur	0,59	**0,07**	0,19	0,19	0,34	6,84	**0,80**	1,61	1,44	1,29	11,08	**1,43**	2,62	3,48	-	16,14	**0,28**	1,29	0,73	-
	Ecart type	0,87	0,21	0,42	0,42	0,60	3,73	1,78	1,34	1,17	0,91	5,53	3,74	3,52	4,52	-	5,10	0,40	0,81	0,91	-
	N° des meilleures solutions	20	45	40	40	35	0	40	5	5	0	5	30	0	15	-	0	25	5	20	-

6.2.2.3 Cas de *V* = (*RCb*, Wb, Wb, RSb*)

Le vecteur *V* = (*RCb*, Wb, Wb, RSb*) est étudié dans ce paragraphe. Les moyennes des résultats des 50 instances sont présentées dans les Tableaux 58 et 59.

Lorsque la contrainte *RCb** est introduite entre la première et la deuxième machine, le problème devient difficile à résoudre. En effet, cette contrainte considère l'opération suivante pour ordonnancer celle en cours et la méthode basée sur la PLNE a besoin de beaucoup de temps pour trouver les solutions. La PLNE n'arrive à trouver des solutions que lorsque le système subit des faibles perturbations. Lorsque p_θ dépasse 0,2, celle-ci n'arrive plus à fournir des solutions en un temps raisonnable. La simulation a été interrompue au bout de 12 heures de calcul. En revanche, les heuristiques arrivent toujours à fournir des solutions, mais leurs *TR* et *DMI* augmentent comparativement au cas du blocage étudié précédemment.

D'autre part, une diminution du pourcentage d'erreur et de l'écart type a été observée dans le Tableau 59 comparé au Tableau 57. Comme la contrainte *RCb** est introduite entre la première et la deuxième machine, l'espace des solutions réalisables devient encore plus petit. H2*, H3* et H4* qui se basent sur la méthode NEH pour réordonnancer les jobs, convergent encore plus souvent vers les mêmes solutions. H1* a toujours un pourcentage d'erreur élevé par rapport aux autres heuristiques et dans la plupart des cas, elle n'arrive pas à fournir de bonnes solutions.

Dans ce cas particulier, H4* reste toujours efficace lorsque la stabilité n'est pas prise en compte, $\alpha = 1$. H2* reste aussi efficace lorsque la stabilité est prise en compte. H3* fournit de meilleures solutions comparées au cas de blocage étudié précédemment.

Tableau 58. Comparaison des méthodes de résolution avec *V* = (*RCb*, Wb, Wb, RSb*)

		p_β = 0,2 (10 jobs)					p_β = 0,5 (24 jobs)					p_β = 0,8 (39 jobs)					p_β = 1 (48 jobs)				
		H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE
	f_1	**443,6**	**443,6**	**443,6**	**443,6**	**443,6**	1876,4	1847,0	1870,0	**1844,6**	-	5183,8	5166,0	5178,7	**5165,1**	-	8218,3	8198,6	8198,1	**8064,0**	-
α=1	DMI (*s*)	0,04	0,05	0,04	0,04	3,04	0,28	0,26	0,32	0,29	-	1,32	1,10	1,13	1,13	-	2,15	2,02	1,92	2,12	-
	TR (*s*)	0,46	0,42	0,43	0,41	11,92	2,73	2,90	3,05	2,93	-	15,75	14,64	14,37	14,49	-	32,61	31,35	29,98	30,52	-
	f_1	385,5	**383,5**	383,6	383,6	**383,5**	1671,6	**1627,3**	1631,6	1632,1	-	4574,7	**4398,8**	4444,4	4421,8	-	7130,1	**6841,5**	6930,4	6898,4	-
α=0,75	DMI (*s*)	0,04	0,04	0,04	0,04	3,73	0,25	0,30	0,29	0,28	-	1,14	1,18	1,12	1,17	-	2,32	1,84	2,03	2,16	-
	TR (*s*)	0,40	0,42	0,43	0,44	12,92	2,85	2,98	3,03	2,95	-	14,82	14,98	14,85	14,85	-	33,48	29,83	30,68	31,13	-
	f_1	310,1	**286,5**	287,7	286,6	286,9	1393,1	**1166,7**	1188,2	1186,1	-	3689,0	**2948,1**	3052,4	2997,8	-	5646,8	**4462,3**	4558,9	4533,4	-
α=0,5	DMI (*s*)	0,04	0,04	0,04	0,04	3,82	0,33	0,27	0,30	0,30	-	1,16	1,12	1,15	1,04	-	2,48	2,11	1,89	2,14	-
	TR (*s*)	0,43	0,41	0,43	0,43	13,64	3,32	3,03	3,23	3,08	-	15,41	14,54	15,66	14,46	-	35,74	31,71	29,78	31,67	-

Tableau 59. Pourcentage d'erreur et écart type entre la solution fournie et la meilleure solution dans le cas de *V* = (*RCb*, Wb, Wb, RSb*)

		p_β = 0,2 (10 jobs)					p_β = 0,5 (24 jobs)					p_β = 0,8 (39 jobs)					p_β = 1 (48 jobs)				
		H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE	H1*	H2*	H3*	H4*	PLNE
	Pourcentage d'erreur	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	2,13	0,26	1,78	**0,12**	-	0,40	0,06	0,30	**0,04**	-	2,18	1,95	1,94	**0,02**	-
α=1	Ecart type	0,00	0,00	0,00	0,00	0,00	4,95	0,26	4,78	0,21	-	0,20	0,10	0,24	0,10	-	5,32	5,38	5,24	0,05	-
	N° des meilleures solutions	50	50	50	50	50	5	15	25	30	-	0	30	5	40	-	5	10	10	25	-
	Pourcentage d'erreur	0,56	**0,04**	0,05	0,05	**0,04**	2,82	**0,09**	0,34	0,38	-	4,04	**0,00**	1,03	0,52	-	4,24	**0,00**	1,28	0,80	-
α=0,75	Ecart type	0,71	0,12	0,12	0,12	0,12	0,45	0,14	0,55	0,53	-	0,90	0,00	0,74	0,59	-	0,59	0,00	0,81	0,70	-
	N° des meilleures solutions	30	40	40	40	40	0	35	30	15	-	0	50	5	10	-	0	45	5	5	-
	Pourcentage d'erreur	8,48	**0,09**	0,59	0,10	0,24	20,14	**0,02**	1,92	1,74	-	25,72	**0,01**	3,48	1,77	-	27,07	**0,10**	2,32	1,77	-
α=0,5	Ecart type	5,18	0,28	0,91	0,30	0,49	6,76	0,07	1,05	1,22	-	6,44	0,04	5,14	1,31	-	5,40	0,21	1,57	1,78	-
	N° des meilleures solutions	7	45	35	45	40	0	45	5	5	-	0	45	5	0	-	0	30	5	15	-

7. Conclusion

Ce chapitre étudie un problème de réordonnancement dans un système flowshop lorsque différentes contraintes de blocages sont mixées dans un seul système de production. Deux aspects sont simultanément considérés. En termes d'efficacité, le *TAMP* est utilisé comme critère. Et en termes de stabilité, la *DMDFP* est utilisée comme critère pour limiter l'écart par rapport à l'ordonnancement initial. A chaque période, l'ordonnancement en cours d'exécution peut être perturbé par l'arrivée ou l'annulation d'un job. En utilisant la stratégie prédictive-réactive, un réordonnancement est effectué en réponse à ces perturbations. Le problème décrit est, en premier lieu, résolu à travers la PLNE. Due à sa complexité *NP-difficile*, la résolution n'est possible que pour un nombre limité de jobs. Des heuristiques ont donc été proposées pour explorer plus de jobs en un temps plus court.

En premier lieu, deux heuristiques basées sur la PLNE ont été développées et testées. Leur implantation a permis de relever les conclusions suivantes :

- L'heuristique Hd2 consiste à diviser la partie à réordonnancer en deux sous-parties. Ensuite, elle commence à établir des améliorations en partant de la gauche vers la droite de la séquence. Cette heuristique fournit des meilleures solutions comparées à Hd1 qui commence de la droite vers la gauche.

- Le coefficient d'efficacité-stabilité α impacte le temps de résolution des heuristiques. Comme la stabilité consiste à fixer les jobs existants, Hd1 fait moins d'améliorations lorsque α diminue, contrairement à Hd2 qui fait plus d'améliorations dans le même cas.

- Lorsqu'un nouveau job est inclus dans la séquence, les jobs existants ne seront pas seulement décalés, mais leurs positions peuvent également être commutées. Ce phénomène de commutation de jobs est le résultat du changement de la structure du système flowshop après chaque perturbation, dû à la variation des durées de jobs.

Ensuite, quatre heuristiques basées sur la méthode NEH ont été développées et testées. Leur implantation a permis de relever les conclusions suivantes :

- Le temps de résolution des heuristiques dépend à la fois de la fréquence d'apparition des jobs et du type de contraintes de blocage entre les machines.

- Lorsque la stabilité n'est pas prise en compte, H4* qui séquence les jobs par ordre décroissant de w_j et utilise la méthode NEH pour réordonnancer ces jobs, fournit des meilleurs résultats puisque les poids des jobs ont un impact majeur sur la performance de la solution. Cependant, lorsque la stabilité est considérée, H2* qui consiste à maintenir l'ordre de la séquence précédente et utilise la méthode NEH pour réordonnancer ces jobs, fournit des meilleurs résultats puisque la déviation par rapport à la séquence précédente est limitée par le critère de stabilité. L'un des meilleurs compromis qu'un décideur peut choisir, c'est d'utiliser, pour chaque cas, une heuristique parmi celles proposées. Idéalement, H4* lorsque $\alpha = 1$, et H2* lorsque $\alpha = 0{,}5$ ou $\alpha = 0{,}75$.

- Considérer les contraintes de blocage mixtes dans un système flowshop réduit l'espace des solutions réalisables puisqu'il y a beaucoup de contraintes à satisfaire, et réduit le pourcentage d'erreur, mais, augmente le temps de résolution des heuristiques qui

doivent vérifier les contraintes à chaque itération. La méthode basée sur la PLNE et les heuristiques convergent, dans ce cas, souvent vers la même solution. Cette situation a été clairement illustrée lorsque la contrainte *RCb** a été introduite en premier dans le système flowshop.

Ce travail est d'une grande utilité pour les preneurs de décision dans le domaine hospitalier ou industriel. Il permet de fournir, à chaque arrivée d'une perturbation, un ordonnancement qui optimise à la fois l'efficacité et la stabilité dans un système flowshop avec des contraintes de blocage mixtes. Il a fait l'objet de plusieurs contributions :

- Un article publié dans une revue internationale [AP1] ;
- Un article présentant les heuristiques basées sur la PLNE est publié dans une conférence internationale [CP3] ;
- Un article présentant les heuristiques basées sur la PLNE, intégrant les contraintes de blocage mixtes, publié dans une conférence internationale [CP2] ;
- Un article présentant les heuristiques basées sur la méthode NEH, intégrant les contraintes de blocage mixtes, rédigé et prêt à être soumis dans une revue internationale [AS1].

Conclusion générale et perspectives

Dans le cadre de cette thèse, nous avons étudié, dans des problèmes de réordonnancement, une nouvelle mesure de performance. Celle-ci a été proposée grâce à une synthèse bibliographique dans laquelle nous avons constaté d'une part, que les poids des jobs n'ont pas été assez pris en compte dans les mesures d'efficacité et de stabilité des problèmes de réordonnancement, et d'autre part, que les contraintes de blocage entre les machines ont reçu peu d'attention de la part des chercheurs qui traitent des travaux de réordonnancement.

La nouvelle mesure de performance qui a été définie combine simultanément l'efficacité et la stabilité de l'ordonnancement. Premièrement, en termes d'efficacité, le temps d'attente moyen pondéré par les poids des jobs, a été considéré comme mesure. Dans les systèmes de production, ce critère correspond à l'attente des jobs devant un poste de travail, en considérant les poids comme étant les priorités des clients. Dans les systèmes hospitaliers, celui-ci correspond à l'attente des patients avant d'être opérés, en considérant les poids comme étant les niveaux d'urgences des patients. Deuxièmement, en termes de stabilité, la déviation moyenne des dates de fin pondérées par les poids des jobs, a été considérée comme mesure. Ce critère évalue la différence entre la date de fin d'un job lorsqu'il est ordonnancé pour la première fois et la date de fin de celui-ci après le réordonnancement. Les poids permettent alors de pénaliser encore plus les mouvements des jobs prioritaires. Ces deux critères décrits sont associés par un coefficient α, nommé coefficient d'*efficacité-stabilité* qui représente la pondération liée à chaque critère. Cette nouvelle mesure de performance a été appliquée sur plusieurs environnements de machines, à savoir, une machine unique, des machines parallèles identiques et dans un atelier flowshop. Ce dernier cas, a été étudié en considérant des contraintes de blocage mixtes entre les machines.

Contributions

Cette thèse a permis d'étudier une nouvelle mesure de performance inspirée d'une situation réelle, où le temps d'attente moyen pondéré est considéré comme critère d'efficacité, et la déviation moyenne des dates de fin des jobs pondérée, est considérée comme critère de stabilité. A cet égard, nous avons implémenté des modèles basés sur la PLNE et des stratégies prédictive-réactive pour le réordonnancement des jobs en réponse aux perturbations dues à l'arrivée de nouveaux jobs et à l'annulation de jobs.

Dans le chapitre 2, une première application de ce modèle a été établie sur une machine unique, dont nous avons constaté que l'augmentation du temps d'attente moyen pondéré *TAMP* est due essentiellement aux jobs qui ont un poids faible. Ces jobs sont reportés à chaque étape de réordonnancement, car les nouveaux jobs qui arrivent s'insèrent au milieu des séquences. Par conséquent, nous avons développé un nouveau concept qui consiste à augmenter les poids des jobs en fonction du temps pour aider les jobs de faible poids à ne pas être ignorés.

Ensuite, une deuxième application a été effectuée sur des machines parallèles identiques, pour lesquelles nous avons aussi mis en œuvre une méthode d'optimisation basée sur la lexicographie pour minimiser le nombre de jobs qui changent de machine après le réordonnancement, et puis nous avons analysé l'effet de cette méthode sur la performance des

solutions obtenues. Dans l'application de ce deuxième modèle, nous avons constaté que le critère de stabilité génère très souvent un effet proactif lorsque α est compris entre 0,7 et 0,9, fournissant des meilleurs résultats comparés à un monocritère qui ne considère que l'efficacité.

Enfin une troisième application a été établie dans un système flowshop avec l'intégration des contraintes de blocage mixtes dans le problème. L'implantation de ce modèle a permis de constater que le temps de résolution de la méthode basée sur la PLNE dépend à la fois, du nombre des jobs initiaux, de la fréquence d'apparition des jobs, et des types de contraintes de blocages entre les machines. Ainsi, des méthodes heuristiques adaptées pour ce problème ont été développées permettant de parcourir un nombre important de jobs. En implémentant les heuristiques basées sur la PLNE, nous avons conclu que l'heuristique Hd2, qui consiste à diviser la partie à réordonnancer en deux sous-parties et commence à établir des améliorations en partant de la gauche vers la droite de la séquence, fournit des meilleures solutions comparées à Hd1 qui va de la droite vers la gauche. D'autre part, le coefficient d'efficacité-stabilité α impacte le temps de résolution des heuristiques. Il a aussi été constaté que lorsqu'un nouveau job est inclus dans la séquence, les jobs existants ne seront pas seulement décalés, mais leurs positions peuvent également être commutées. En implémentant les heuristiques basées sur la méthode NEH, nous avons conclu que l'un des meilleurs compromis qu'un décideur peut choisir, c'est d'utiliser, pour chaque cas, une heuristique parmi celles proposées. Idéalement, H4* lorsque $\alpha = 1$, et H2* lorsque $\alpha = 0{,}5$ ou $\alpha = 0{,}75$.

Perspectives

Ce travail est d'une grande utilité pour les preneurs de décision qui gèrent les activités d'ordonnancement dans les systèmes industriels ou hospitaliers. Il aide à fournir une solution, à chaque étape de réordonnancement, en réponse à des perturbations dues à l'arrivée ou à l'annulation de jobs. Cependant, la principale limite de ce travail, c'est qu'il ne gère que l'arrivée ou l'annulation d'un seul job par période. Dans ces dernières années, bien avant la pandémie du COVID-19, la pression sur les services hospitaliers n'a cessé d'augmenter. En effet, les preneurs de décision doivent très rapidement réagir afin de fournir un nouveau planning en réponse aux perturbations dues à l'arrivée de nouveaux patients urgents, même si plusieurs d'entre eux arrivent en même temps. Il faudrait donc adapter nos heuristiques pour prendre en compte l'arrivée simultanées de plusieurs jobs.

Les heuristiques proposées permettent d'accélérer l'obtention des solutions en un temps raisonnable. Cependant, le manageur doit choisir parmi les heuristiques proposées, celle qui est adaptée à son cas, en fonction du coefficient d'efficacité-stabilité. Comme perspective pour ce travail, il sera judicieux, dans les travaux futurs, de développer une méthode métaheuristique intelligente qui peut s'adapter aux différentes situations en fonction de la valeur du α choisie.

Il est aussi intéressant d'étudier le comportement du *TAMP* combiné avec la *DMDFP* dans d'autres types d'environnement machines, notamment les ateliers flowshop hybrides qui peuvent être une illustration des patients qui doivent être traités, dans le même ordre, dans des unités de soins, et que chaque unité est composée de plusieurs salles opératoires identiques.

Malgré les technologies qui peuvent être mises en place, la réalisation des opérations ne peut aboutir que par l'implication des personnels. Le processus de réordonnancement nécessite des changements réguliers des calendriers de production. Cela entraine plusieurs phénomènes humains qui exercent une influence sur la durée de réalisation de certaines tâches, comme la

fatigue mentale, ou la résistance aux changements, etc. Nous proposons, dans le long terme, d'intégrer ces aspects humains à nos futurs travaux.

Références bibliographiques

Abdel-Jabbar, M. A. H., Kacem, I., & Martin, S. (2014, October). Unrelated parallel machines with precedence constraints: application to cloud computing. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)* (pp. 438-442). IEEE.

Abedini, A., Ye, H., & Li, W. (2016). Operating room planning under surgery type and priority constraints. *Procedia Manufacturing*, *5*, 15-25.

Abumaizar, R. J., & Svestka, J. A. (1997). Rescheduling job shops under random disruptions. *International journal of production research*, *35*(7), 2065-2082.

Addis, B., Carello, G., Grosso, A., & Tànfani, E. (2016). Operating room scheduling and rescheduling: a rolling horizon approach. *Flexible Services and Manufacturing Journal*, *28*(1-2), 206-232.

Ahmadi, R. H., & Bagchi, U. (1990). Lower bounds for single-machine scheduling problems. *Naval Research Logistics (NRL)*, *37*(6), 967-979.

Akkan, C. (2015). Improving schedule stability in single-machine rescheduling for new operation insertion. *Computers & Operations Research*, 64, 198-209.

Aktürk, M. S., Atamtürk, A., & Gürel, S. (2010). Parallel machine match-up scheduling with manufacturing cost considerations. *Journal of Scheduling*, 13(1), 95-110.

Alagöz, O., & Azizoğlu, M. (2003). Rescheduling of identical parallel machines under machine eligibility constraints. *European journal of operational research*, *149*(3), 523-532.

Angel-Bello, F., Vallikavungal, J., & Alvarez, A. (2021). Fast and efficient algorithms to handle the dynamism in a single machine scheduling problem with sequence-dependent setup times. *Computers & Industrial Engineering*, 152, 106984.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, *161*(1), 86-110.

Baker, K. R., & Keller, B. (2010). Solving the single-machine sequencing problem using integer programming. *Computers & Industrial Engineering*, *59*(4), 730-735.

Ballestín, F., Pérez, Á., & Quintanilla, S. (2019). Scheduling and rescheduling elective patients in operating rooms to minimise the percentage of tardy patients. *Journal of Scheduling*, *22*(1), 107-118.

Beezão, A. C., Cordeau, J. F., Laporte, G., & Yanasse, H. H. (2017). Scheduling identical parallel machines with tooling constraints. *European journal of operational research*, *257*(3), 834-844.

Bhaskaran, K., & Pinedo, M., (1991). "Handbook of Industrial Engineering", John Wiley, New York.

Bouguerra, A., Sauvey, C., & Sauer, N. (2015). Mathematical model for maximizing operating rooms utilization. *IFAC-PapersOnLine*, 48(3), 118-123.

Bouguerra, A., Sauvey, C., & Sauer, N. (2016). Online assignment strategies for emergent, urgent and work-in-cases surgeries in an operating theatre. In *2016 International Conference on Control, Decision and Information Technologies (CoDIT)* (pp. 438-443). IEEE.

Bouguerra, A., Sauvey, C., & Sauer, N. (2020). A decision support tool for the urgent surgeries assignment problem. *Journal of Engineering Research*, 8(2).

Cardoen, B., Demeulemeester, E., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European journal of operational research*, *201*(3), 921-932.

Chaari, T., Chaabane, S., Loukil, T., & Trentesaux, D. (2011). A genetic algorithm for robust hybrid flow shop scheduling. *International Journal of Computer Integrated Manufacturing*, *24*(9), 821-833.

Chacon, G. R. (1998). Using simulation to integrate scheduling with the manufacturing execution system. *Future Fab International*, 63-66.

Chang, K. H. (2014). *Design theory and methods using CAD/CAE: The computer aided engineering design series*. Academic Press.

Cheng, M., Xiao, S., & Liu, G. (2018). Single-machine rescheduling problems with learning effect under disruptions. *Journal of Industrial & Management Optimization*, *14*(3), 967.

Choi, S., & Wilhelm, W. E. (2012). An analysis of sequencing surgeries with durations that follow the lognormal, gamma, or normal distribution. *IIE Transactions on Healthcare Systems Engineering*, *2*(2), 156-171.

Church, L. K., & Uzsoy, R. (1992). Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, *5*(3), 153-163.

Da Silva, N. C. O., Scarpin, C. T., Pécora Jr, J. E., & Ruiz, A. (2019). Online single machine scheduling with setup times depending on the jobs sequence. *Computers & Industrial Engineering*, 129, 251-258.

Daniels, R. L., & Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, *41*(2), 363-376.

Dauzère-Pérès, S., C. Pavageau et N. Sauer. (2000). Modélisation et résolution par PLNE d'un problème réel d'ordonnancement avec contraintes de blocage. *3ème congrès ROADEF, Nantes*, pp. 216-217.

Denton, B., Viapiano, J., & Vogl, A. (2007). Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health care management science*, *10*(1), 13-24.

Dhingra, J. S., Musser, K. L., & Blankenship, G. L. (1992, December). Realtime operations scheduling for flexible manufacturing systems. In *Proceedings of the 24th conference on Winter simulation* (pp. 849-855).

Dou, J., Su, C., & Zhao, X. (2020). Mixed integer programming models for concurrent configuration design and scheduling in a reconfigurable manufacturing system. *Concurrent Engineering*, *28*(1), 32-46.

Duenas, A., & Petrovic, D. (2008). An approach to predictive-reactive scheduling of parallel machines subject to disruptions. *Annals of Operations Research*, 159(1), 65-82.

El-Bouri, A. (2012). A cooperative dispatching approach for minimizing mean tardiness in a dynamic flowshop. *Computers & Operations Research*, *39*(7), 1305-1314.

Erdem, E., Qu, X., & Shi, J. (2012). Rescheduling of elective patients upon the arrival of emergency patients. *Decision Support Systems*, *54*(1), 551-563.

Eren, T. (2009). Minimizing the total weighted completion time on a single machine scheduling with release dates and a learning effect. *Applied Mathematics and Computation*, *208*(2), 355-358.

Framinan, J. M., Fernandez-Viagas, V., & Perez-Gonzalez, P. (2019). Using real-time information to reschedule jobs in a flowshop with variable processing times. *Computers & Industrial Engineering*, 129, 113-125.

Gao, K. Z., Suganthan, P. N., Tasgetiren, M. F., Pan, Q. K., & Sun, Q. Q. (2015). Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. *Computers & Industrial Engineering*, 90, 107-117.

Ghezail, F., Pierreval, H., & Hajri-Gabouj, S. (2010). Analysis of robustness in proactive scheduling: A graphical approach. *Computers & Industrial Engineering*, *58*(2), 193-198.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Guo, Q., & Tang, L. (2019). Modelling and discrete differential evolution algorithm for order rescheduling problem in steel industry. *Computers & Industrial Engineering*, *130*, 586-596.

Guo, Y., & Xie, X. (2017, October). Two mixed integer programming formulations on single machine to reschedule repaired jobs for minimizing the total waiting-time. In *2017 Chinese Automation Congress (CAC)* (pp. 2129-2133). IEEE.

Guo, Y., Huang, M., Wang, Q., & Leon, V. J. (2016). Single-machine rework rescheduling to minimize maximum waiting-times with fixed sequence of jobs and ready times. *Computers & industrial engineering*, *91*, 262-273.

Hall, N. G., & Potts, C. N. (2004). Rescheduling for new orders. *Operations Research*, 52(3), 440-453.

Hall, N. G., & Potts, C. N. (2010). Rescheduling for job unavailability. *Operations Research*, 58(3), 746-755.

Hall, N. G., Kamoun, H., & Sriskandarajah, C. (1998). Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research*, *109*(1), 43-65.

Hall, N. G., Liu, Z., & Potts, C. N. (2007). Rescheduling for multiple new orders. *INFORMS Journal on Computing*, *19*(4), 633-645.

Hamzadayi, A., & Yildiz, G. (2016). Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. *Computers & Industrial Engineering*, *91*, 66-84.

Haupt, R., (1989). "A survey of priority rule-based scheduling", OR Spektrum 11, 3-16.

He, X., Dong, S., & Zhao, N. (2020). Research on rush order insertion rescheduling problem under hybrid flow shop based on NSGA-III. *International Journal of Production Research*, *58*(4), 1161-1177.

Herrmann, J. W. (2006). Decision-making systems in production scheduling. In *Handbook of production scheduling* (pp. 91-108). Springer, Boston, MA.

Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research*, *156*(3), 550-565.

Hoogeveen, H., Lenté, C., & T'kindt, V. (2012). Rescheduling for new orders on a single machine with setup times. *European Journal of Operational Research*, 223(1), 40-46.

Jensen, M. T. (2003). Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Transactions on evolutionary computation*, *7*(3), 275-288.

Jorge Leon, V., David Wu, S., & Storer, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE transactions*, 26(5), 32-43.

Josyula, S. P., Krasemann, J. T., & Lundberg, L. (2018). A parallel algorithm for train rescheduling. *Transportation Research Part C: Emerging Technologies*, *95*, 545-569.

Jun, S., & Lee, S. (2021). Learning dispatching rules for single machine scheduling with dynamic arrivals based on decision trees and feature construction. *International Journal of Production Research*, *59*(9), 2838-2856.

Kan, A. R. (1976). Problem formulation. In *Machine Scheduling Problems* (pp. 5-29). Springer, Boston, MA.

Katragjini, K., Vallada, E., & Ruiz, R. (2013). Flow shop rescheduling under different types of disruption. *International Journal of Production Research*, *51*(3), 780-797.

Kecman, P., Corman, F., D'Ariano, A., & Goverde, R. M. (2013). Rescheduling models for railway traffic management in large-scale networks. *Public Transport*, 5(1-2), 95-123.

Kooli, A., & Serairi, M. (2014). A mixed integer programming approach for the single machine problem with unequal release dates. *Computers & operations research*, *51*, 323-330.

Labetoulle, J., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G.(1984). Preemptive scheduling of uniform machines subject to release dates. In *Progress in combinatorial optimization* (pp. 245-261). Academic Press.

Lee, Y. C., & Zomaya, A. Y. (2010). Rescheduling for reliable job completion with the support of clouds. *Future Generation Computer Systems*, *26*(8), 1192-1199.

Lei, D. M., & Xiong, H. J. (2008, July). Job shop scheduling with stochastic processing time through genetic algorithm. In *2008 International Conference on Machine Learning and Cybernetics* (Vol. 2, pp. 941-946). IEEE.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. In *Annals of discrete mathematics* (Vol. 1, pp. 343-362). Elsevier.

Letsios, D., Mistry, M., & Misener, R. (2021). Exact lexicographic scheduling and approximate rescheduling. *European Journal of Operational Research*, *290*(2), 469-478.

Li, J. Q., Mirchandani, P. B., & Borenstein, D. (2009). A Lagrangian heuristic for the real-time vehicle rescheduling problem. *Transportation Research Part E: Logistics and Transportation Review*, *45*(3), 419-433.

Li, J. Q., Pan, Q. K., & Mao, K. (2015). A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Engineering Applications of Artificial Intelligence*, *37*, 279-292.

Li, R. K., Shyu, Y. T., & Adiga, S. (1993). A heuristic rescheduling algorithm for computer-based production scheduling systems. *The International Journal of Production Research*, *31*(8), 1815-1826.

Li, Z., & Ierapetritou, M. (2008). Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, *32*(4-5), 715-727.

Liu, F., Wang, S., Hong, Y., & Yue, X. (2017). On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. *IEEE Transactions on Engineering Management*, *64*(4), 539-553.

Liu, L. (2019). Outsourcing and rescheduling for a two-machine flow shop with the disruption of new arriving jobs: A hybrid variable neighborhood search algorithm. *Computers & Industrial Engineering*, *130*, 198-221.

Liu, Z., & Ro, Y. K. (2014). Rescheduling for machine disruption to minimize makespan and maximum lateness. *Journal of Scheduling*, 17(4), 339-352.

Liu, L., & Zhou, H. (2013). On the identical parallel-machine rescheduling with job rework disruption. *Computers & Industrial Engineering*, 66(1), 186-198.

Luo, W., Jin, M., Su, B., & Lin, G. (2020). An approximation scheme for rejection-allowed single-machine rescheduling. *Computers & Industrial Engineering*, *146*, 106574.

Luo, W., Luo, T., Goebel, R., & Lin, G. (2018). Rescheduling due to machine disruption to minimize the total weighted completion time. *Journal of Scheduling*, *21*(5), 565-578.

Lust, T., Meskens, N., & Monteiro, T. (2012, June). Ordonnancement multiobjectif du bloc opératoire avec une prise en compte d'une affectation équilibrée des compétences des infirmières. In *9th International Conference on Modeling, Optimization & SIMulation*.

Macario A., T.S. Vitez, B. Dunn and T. Mc Donald, 1995, Where are the costs in perioperative care? Analysis of hospital costs and charges for inpatient surgical care, *Anesthesiology*, vol. 83, pp. 1138-44.

Martinez, S. Ordonnancement de systèmes de production avec contraintes de blocage. *Thèse, Université de Nantes*, 2005.

Mavrotas, G. (2009). Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, *213*(2), 455-465.

Molaee, E., Sadeghian, R., & Fattahi, P. (2021). Minimizing maximum tardiness on a single machine with family setup times and machine disruption. *Computers & Operations Research*, *129*, 105231.

Nagasawa, K., Ikeda, Y., & Irohara, T. (2015). Robust flow shop scheduling with random processing times for reduction of peak power consumption. *Simulation Modelling Practice and Theory*, *59*, 102-113.

Nessah, R., Chu, C., Yalaoui, F., & Kacem, I. (2006, October). A Branch and bound for 1| ri| wiCi, Scheduling Problem. In *The Proceedings of the Multiconference on" Computational Engineering in Systems Applications"* (Vol. 1, pp. 1047-1053). IEEE.

Nie, L., Gao, L., Li, P., & Shao, X. (2013). Reactive scheduling in a job shop where jobs arrive over time. *Computers & Industrial Engineering*, 66(2), 389-405.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling*, *12*(4), 417-431.

Ozlen, M., & Azizoğlu, M. (2011). Rescheduling unrelated parallel machines with total flow time and total disruption cost criteria. *Journal of the Operational Research Society*, 62(1), 152-164.

Peng, K., Pan, Q. K., Gao, L., Li, X., Das, S., & Zhang, B. (2019). A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm and Evolutionary Computation*, *45*, 92-112.

Peng, K., Pan, Q. K., Gao, L., Zhang, B., & Pang, X. (2018). An Improved Artificial Bee Colony algorithm for real-world hybrid flowshop rescheduling in Steelmaking-refining-Continuous Casting process. *Computers & Industrial Engineering*, 122, 235-250.

Pfeiffer, A., Kádár, B., Monostori, L., & Karnok, D. (2008). Simulation as one of the core technologies for digital enterprises: assessment of hybrid rescheduling methods. *International Journal of Computer Integrated Manufacturing*, 21(2), 206-214.

Pinedo, M. L. (2012). Deterministic models: preliminaries. *In Scheduling* (pp. 13-33). Springer, Boston, MA.

Qiao, F., Ma, Y., Zhou, M., & Wu, Q. (2018). A novel rescheduling method for dynamic semiconductor manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(5), 1679-1689.

Rahmani, D., & Heydari, M. (2014). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems*, *33*(1), 84-92.

Rahmani, D., & Ramezanian, R. (2016). A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers & Industrial Engineering*, *98*, 360-372.

Rangsaritratsamee, R., Ferrell Jr, W. G., & Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering*, 46(1), 1-15.

Rossit, D. A., Tohmé, F., & Frutos, M. (2019). Industry 4.0: smart scheduling. *International Journal of Production Research*, *57*(12), 3802-3813.

Sabuncuoglu, I., & Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, *18*(4), 268-283.

Sahraeian, R., Rahmani, D., & Abtahi, Z. (2020). Predictive heuristics for generating robust and stable schedules in single-machine systems under disruption. *International Journal of Science and Technology*, Scientia Iranica, 27(5).

Salido, M. A., Escamilla, J., Barber, F., & Giret, A. (2017). Rescheduling in job-shop problems for sustainable manufacturing systems. *Journal of cleaner production*, *162*, S121-S132.

Spliet, R., Gabor, A. F., & Dekker, R. (2014). The vehicle rescheduling problem. *Computers & Operations Research*, *43*, 129-136.

Tang, D., Dai, M., Salido, M. A., & Giret, A. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry*, *81*, 82-95.

Tang, L., & Zhang, Y. (2009). Parallel machine scheduling under the disruption of machine breakdown. *Industrial & engineering chemistry research*, 48(14), 6660-6667.

Tao, Z., & Liu, X. (2019). Dynamic Scheduling of Dual-Resource Constrained Blocking Job Shop. *In International Conference on Intelligent Robotics and Applications* (pp. 447-456). Springer, Cham.

Tighazoui, A., Turki, S., Sauvey, C., & Sauer, N. (2019). Optimal design of a manufacturing-remanufacturing-transport system within a reverse logistics chain. *The International Journal of Advanced Manufacturing Technology*, *101*(5), 1773-1791.

Trabelsi, W., Sauvey, C., & Sauer, N. (2012a). Mathematical Model and Lower Bound for Hybrid Flowshop Problem With Mixed Blocking Constraints. *IFAC Proceedings Volumes*, *45*(6), 1475-1480.

Trabelsi, W., Sauvey, C., & Sauer, N. (2012b). Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems. *Computers & Operations Research*, *39*(11), 2520-2527.

Trabelsi, W. (2012). Ordonnancement des systèmes de production flexibles soumis à différents types de contraintes de blocage.(Scheduling of flexible manufacturing systems subject to mixed blocking constraints). *Doctoral dissertation, University of Lorraine, Nancy, France*.

Uhlmann, I. R., & Frazzon, E. M. (2018). Production rescheduling review: Opportunities for industrial integration and practical applications. *Journal of manufacturing systems*, *49*, 186-193.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2000a). Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies. *International journal of production research*, 38(8), 1899-1915.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2000b). Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of manufacturing Systems*, *19*(4), 256-266.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of scheduling*, *6*(1), 39-62.

Wang, B., Han, X., Zhang, X., & Zhang, S. (2015). Predictive-reactive scheduling for single surgical suite subject to random emergency surgery. *Journal of Combinatorial Optimization*, 30(4), 949-966.

Wang, L., Zhang, L., & Zheng, D. Z. (2006). An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research*, 33(10), 2960-2971.

Wang, D., Yin, Y., & Jin, Y. (2020). Parallel-Machine Rescheduling with Job Rejection in the Presence of Job Unavailability. In *Rescheduling Under Disruptions in Manufacturing Systems* (pp. 35-63). Springer, Singapore.

Wang, S., Wu, R., Chu, F., & Yu, J. (2020). Identical parallel machine scheduling with assurance of maximum waiting time for an emergency job. *Computers & Operations Research*, 118, 104918.

Wu, H. H., & Li, R. K. (1995). A new rescheduling method for computer based scheduling systems. *International journal of production research*, *33*(8), 2097-2110.

Wu, S. D., Byeon, E. S., & Storer, R. H. (1999). A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, *47*(1), 113-124.

Wu, S.D., & Wysk, R.A., (1989). "An application of discrete-event simulation to on-line control and scheduling of flexible manufacturing". *International Journal of Production Research*, 27(9).

Wu, S. D., Storer, R. H., & Pei-Chann, C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers & Operations Research*, *20*(1), 1-14.

Wullink, G., Van Houdenhoven, M., Hans, E. W., Van Oostrum, J. M., Van Der Lans, M., & Kazemier, G. (2007). Closing emergency operating rooms improves efficiency. *Journal of Medical Systems*, *31*(6), 543-546.

Yang, B. (2007). Single machine rescheduling with new jobs arrivals and processing time compression. *The International Journal of Advanced Manufacturing Technology*, 34(3-4), 378-384.

Yang, B., & Geunes, J. (2008). Predictive–reactive scheduling on a single resource with uncertain future jobs. *European Journal of Operational Research*, *189*(3), 1267-1283.

Yang, Z., Li, F., & Liu, F. (2021). On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. In *Applications of Artificial Intelligence in Process Systems Engineering* (pp. 381-416). Elsevier.

Yao, G., Ding, Y., Ren, L., Hao, K., & Chen, L. (2016). An immune system-inspired rescheduling algorithm for workflow in Cloud systems. *Knowledge-Based Systems*, *99*, 39-50.

Yin, Y., Cheng, T. C. E., & Wang, D. J. (2016). Rescheduling on identical parallel machines with machine disruptions to minimize total completion time. *European Journal of Operational Research*, *252*(3), 737-749.

Yuan, J., Mu, Y., Lu, L., & Li, W. (2007). Rescheduling with release dates to minimize total sequence disruption under a limit on the makespan. *Asia-Pacific Journal of Operational Research*, 24(06), 789-796.

Yue, Q., & Zhou, S. (2021). Due-window assignment scheduling problem with stochastic processing times. *European Journal of Operational Research*, *290*(2), 453-468.

Zakaria, Z., & Petrovic, S. (2012). Genetic algorithms for match-up rescheduling of the flexible manufacturing systems. *Computers & Industrial Engineering*, 62(2), 670-686.

Zhang, B., Pan, Q. K., Gao, L., Zhang, X. L., & Peng, K. K. (2019). A multi-objective migrating birds optimization algorithm for the hybrid flowshop rescheduling problem. *Soft Computing*, *23*(17), 8101-8129.

Zhang, S., & Wong, T. N. (2017). Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. *International Journal of Production Research*, *55*(11), 3173-3196.

Zhao, Y., & Wang, G. (2020). A dynamic differential evolution algorithm for the dynamic single-machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, *71*(2), 225-236.

Zhu, S., Fan, W., Yang, S., Pei, J., & Pardalos, P. M. (2019). Operating room planning and surgical case scheduling: a review of literature. *Journal of Combinatorial Optimization*, *37*(3), 757-805.

Zykina, A. V. (2004). A lexicographic optimization algorithm. *Automation and Remote Control*, *65*(3), 363-368.