Élaboration de nouveaux outils non conventionnels intelligents pour l'ordonnancement conjoint de la production et de la maintenance : application au cas d'un job shop

Amel Yahyaoui

UNIVERSITE DE TUNIS
ECOLE SUPERIEURE DES SCIENCES
ET TECHNIQUES DE TUNIS
ECOLE DOCTORALE : SCIENCES ET TECHNOLOGIES

Élaboration de nouveaux outils non conventionnels intelligents pour l'ordonnancement conjoint de la production et de la maintenance : application au cas d'un job shop

Thèse en vue de l'obtention du doctorat en

Génie Electrique

Option : Productique

par

Mme Amel Yahyaoui épouse Ben Taleb

Soutenue le 28 Juin 2011, devant le jury d'examen composé de :

Président :	**Abdelkader CHAARI**	Maître de Conférences, Ecole Supérieure des Sciences et Techniques de Tunis (ESSTT).
Rapporteurs :	**Noureddine ZERHOUNI**	Professeur, ENSMM Franche-Comté, Besançon, France.
	Noureddine LIOUANE	Maitre de conférences à l'Ecole Nationale d'Ingénieurs de Monastir (ENIM).
Examinateur :	**Moncef TAJINA**	Professeur, Ecole Nationale des Sciences de l'Informatique de l'Université la Manouba.
Directeur de thèse :	**Farhat FNAIECH**	Professeur, Ecole Supérieure des Sciences et Techniques de Tunis (ESSTT).

Résumé

ÉLABORATION DE NOUVEAUX OUTILS NON CONVENTIONNELS INTELLIGENTS POUR L'ORDONNANCEMENT CONJOINT DE LA PRODUCTION ET DE LA MAINTENANCE : APPLICATION AU CAS D'UN JOB SHOP

Résumé : *Les problèmes d'ordonnancement sont présents dans tous les secteurs de l'économie car ils présentent une fonction importante en gestion de production. L'objectif de cette thèse est la résolution du problème d'ordonnancement conjoint de la production et de la maintenance appliquée au cas de l'atelier de type job shop.*

Le problème d'ordonnancement job shop qui traite l'allocation des tâches aux machines tout en respectant un certain nombre de contraintes à savoir les contraintes de séquencement et les contraintes de ressources afin de minimiser un critère lié à la production qui est le Makespan C_{max} a été présenté dans ce manuscrit de thèse.

Deux contributions principales ont été proposées à savoir,

Une méthode d'initialisation des dates de début des opérations basée sur un réseau de neurone type Hopfield a amélioré une métaheuristique déjà proposée dans la littérature par Willems et al. [WIL94] et [WIL95] pour la résolution du problème d'ordonnancement à cheminement multiple type job shop. Cette méthode concerne uniquement les problèmes d'ordonnancement job shop sans tenir compte de l'indisponibilité des machines.

Une nouvelle méthode de décalage des opérations de production et de maintenance afin de résoudre le problème d'ordonnancement conjoint de la production et de la maintenance. Il s'agit d'optimiser deux critères l'un lié à la production le Makespan C_{max} et l'autre lié à la maintenance qui est le coût total de la maintenance CM. Cette méthode se résume dans un ensemble de règles proposées.

Ces deux contributions ont été validées expérimentalement sur des benchmarks usuels. Des comparaisons ont été élaborées avec des méthodes proposées dans la littérature.

Abstract

NEW INTELLIGENT TOOLS FOR COMBINED PRODUCTION SCHEDULING AND MAINTENANCE: APPLICATION TO THE CASE OF JOB SHOP

Abstract: *Artificial neural network models have been successfully applied to solve a job-shop scheduling problem (JSSP) known as a Non Polynomial (NP-complete) constraint satisfaction problem. Our first contribution is an improvement of the Willems and Brandts [WIL94] algorithm proposed in the literature. It consists in using an optimizing procedure of the initial value of the starting time. The aim is to speed a Hopfield Neural Network (HNN), and therefore to reduce the number of searching cycle.*

This new heuristic provides several advantages; mainly improves the searching speed of an optimal or near optimal solution of a deterministic JSSP using HNN and reduce the makespan. Simulation results of the proposed method have been performed on various benchmarks and compared with current algorithms such as genetic algorithm, constraint satisfaction adaptive neural networks, simulated annealing, threshold accepting, flood-method and priority rules such that shortest processing time (SPT) to mention but a few.

As the simulation results show, Willems et al. [WIL94] and [WIL95] algorithm combined with the proposed heuristic method, is efficient with respect to the resolution speed, quality of the solution, and the reduction of the computation time.

As a second contribution in this work, we have been concerned with a new heuristic based on a shifting method used to solve an integrated production and preventive maintenance strategy for a job shop scheduling problem. This shifting new method aims to reduce the makespan Cmax and the maintenance total cost caused by the advances or the delays of preventive maintenance tasks. In general, delayed maintenance tasks have a higher cost because they increase the risk of the breakdowns and generate a stop of production. This proposed method is based on a set of rules taking into account the job shop scheduling problem constraints and the maintenance tasks ranges. To underline the effectiveness of the method, some illustrative benchmarks are provided. The performances of the new algorithm concerns the makespan Cmax, the free time rate and some other statistics related to the production cost.

Keywords: Computer integrated manufacturing, Hopfield networks, manufacturing automation, manufacturing automation software, manufacturing planning, manufacturing scheduling, optimization methods, production management, resource management, Assignment, Availability, constraints, Maintenance, Optimization, Methods, Scheduling

Remerciements

Ce travail a été réalisé au sein de l'unité de recherche SICISI de l'Ecole Supérieure des Sciences et Techniques de Tunis, Université de Tunis.
Pendant toute la durée de ma thèse, j'ai eu la chance de côtoyer et parfois de rencontrer des personnalités réellement extraordinaires.

C'est en grande partie grâce à **Mr Farhat Fnaiech,** Professeur à l'ESSTT, que ce travail a abouti. Ses conseils, ses encouragements et sa grande disponibilité à ses chercheurs m'ont permis à mener à bien cette thèse. Qu'il trouve ici ma profonde gratitude et mon grand respect.

Je tiens à remercier en premier lieu les membres de jury, **Mr Abdelkader Chaari**, Maitre de conférences à l'ESSTT qui m'a fait l'honneur d'accepter la présidence du jury et l'évaluation de cette thèse.

J'exprime mes remerciements particuliers à Monsieur **Nader Fnaiech,** Maitre Assistant à Institut Supérieur d'Informatique de Mahdia pour ces longues et précieuses discussions pour ses conseils et tout ce qu'il m'a enseignés et pour l'aide qu'il m'a apportée.

J'adresse aussi mes remerciements à Monsieur **Noureddine Zerhouni,** Professeur, ENSMM Franche-Comté, Besançon, France. C'était lui qui m'a fourni au départ le sujet de recherche sur la productique. Je le remercie pour avoir rapporté cette thèse.

Je tiens à remercier Monsieur **Noureddine Liouane** Maitre de conférences à l'Ecole Nationale d'Ingénieurs de Monastir (ENIM) pour avoir accepter d'évaluer cette thèse et d'être rapporteur.

J'adresse mes vifs remerciements à Monsieur **Moncef Tajina** Professeur à l'Ecole Nationale des Sciences de l'Informatique de l'Université la Manouba qui a accepté de participer à l'évaluation de ce travail et membre examinateur du jury de cette thèse.

Un grand merci à tous les membres de l'unité de recherche **SICISI** pour leur accueil chaleureux. Je les remercie tous pour la bonne ambiance apportée et pour les discussions techniques et pour les bons moments partagés.

À la mémoire de ma mère

À mon père

À mon mari et à mes enfants

À mes frères et sœurs

À ma belle mère

À toute ma famille

Sommaire

Table des Figures

Liste des Tableaux

Introduction générale

Afin qu'une entreprise demeure compétitive, sa production doit être toujours de bonne qualité et au coût le plus bas. La minimisation du coût peut être réalisée par plusieurs façons entre autres la fabrication des produits plus vite et sans interruption et sans défaut. Cet objectif est l'un des buts qui touchent du près au problème d'ordonnancement qui sera traité dans nos travaux de thèse.

En effet, les problèmes d'ordonnancement ont suscité plusieurs travaux publiés dans la littérature et notamment des ouvrages dont nous pouvons citer : [BAK74], [BAK09], [BLA96], [ESQ99], [KIN06], [PIN08]. Et d'après [LOP01], « *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches* »

Ils sont considérés comme les problèmes les plus compliqués [GAR79] (NP-difficile au sens fort) des problèmes d'optimisation combinatoires souvent rencontrés dans le milieu industriel.

Comme les problèmes d'ordonnancements sont diversifiés, plus particulièrement, nous avons traité un cas particulier d'atelier qui est l'atelier job shop ou les ateliers à cheminement multiples qui concernent la production en discontinu qui est retenue lorsque l'on traite des quantités relativement faibles de nombreux produits variés. L'ordre de passage des tâches ou jobs sur les ressources dans la gamme opératoire peut différer d'un produit à l'autre (tous les produits ne passent pas nécessairement sur toutes les ressources).

Nos travaux présentés dans cette thèse concernent la résolution du problème d'ordonnancement conjoint de la production et de la maintenance dans le cas d'un atelier de type job shop. En effet, Les problèmes d'ordonnancement sont présents dans tous les secteurs de l'économie car ils présentent une fonction importante en gestion de production. De même comme la production et la maintenance sont deux fonctions qui agissent sur les mêmes ressources, alors l'ordonnancement conjoint de leurs opérations parait une tâche assez complexe. Cependant dans la majorité des cas qui traitent le problème d'ordonnancement de la production, les ressources sont toujours supposées disponibles et ne tiennent pas compte de leurs indisponibilités qui peuvent être liées soit à des opérations de maintenances (préventives) ou correctives (pannes…). Ce travail sera développé en trois étapes :

En première étape, nous allons présenter des notions préliminaires et définitions de base pour l'ordonnancement des systèmes de production. En deuxième étape, nous allons proposer des nouvelles procédures d'initialisation en vue de l'optimisation d'un réseau de neurone Hopfield appliqué aux problèmes d'ordonnancement type job shop au cours duquel nous allons résoudre le problème d'ordonnancement sans tenir compte des contraintes d'indisponibilités des machines pour optimiser un critère de production qui est le Makespan C_{max}. En troisième étape nous passons pour résoudre conjointement l'ordonnancement des tâches de la production et de la maintenance tout en optimisant deux critères, un critère lié à la production qui est le Makespan C_{max} et un deuxième critère lié à la maintenance qui est le coût de la maintenance *CM*.

Le premier chapitre, est un chapitre introductif à cette thèse. A travers duquel, nous allons situer notre problématique de recherche qui vise la résolution du problème d'ordonnancement de type job shop dans deux étapes.

L'objectif de ce chapitre introductif au problème d'ordonnancement, est de situer la problématique de notre recherche. Des notions préliminaires et quelques définitions de base pour l'ordonnancement des systèmes de production seront présentées dans ce chapitre qui sera répartie en trois parties.

La première partie est une initiation au problème de l'ordonnancement job shop. Une présentation générale introductive du problème d'ordonnancement visant un exposé des éléments de base à travers leurs définitions correspondantes au problème de l'ordonnancement du type job shop ainsi que sa modélisation utilisant quelques benchmarks usuels finissant par mettre l'accent sur sa difficulté de résolution. Cette partie a été explicitée pour une formulation générale du problème job shop considéré.

La deuxième partie consiste en une présentation de l'état de l'art et une synthèse bibliographique des méthodes de résolution de l'ordonnancement relatif au problème job shop.

La troisième partie consiste en une présentation succincte de la maintenance des systèmes de production mettant l'accent sur la politique de la maintenance qui sera considérée dans notre travail.

Le deuxième chapitre vise à présenter notre contribution principale qui présente une nouvelle procédure d'initialisation en vue de l'optimisation d'un réseau de neurone Hopfield appliqué aux problèmes d'ordonnancement d'ateliers à cheminement multiples : job shop. Cette nouvelle procédure constitue en une amélioration des algorithmes proposés par Willems et

al. [WIL94] et [WIL95]. La démarche consiste en une heuristique de choix optimal d'initialisation des dates de début des opérations afin d'accélérer le réseau de neurones Hopfield (HNN).

L'avantage enregistré de cette nouvelle heuristique est de fournir une amélioration de la convergence du réseau de Hopfield vers une solution optimale ou proche de l'optimale pour le critère à minimiser qui est le Makespan du problème d'ordonnancement des ateliers de type Job Shop.

Les résultats de simulation utilisant cette méthode heuristique améliorée sur plusieurs benchmarks, sont performants. Une étude comparative avec d'autres algorithmes existants, tels que les algorithmes génétiques, les réseaux de neurones adaptatifs de satisfaction de contraintes et les approches combinées par heuristique pour le problème d'ordonnancement des ateliers du type Job Shop généralisé, le recuit simulé, les règles de priorité tel que (SPT) la plus courte durée opératoire, etc., a été élaborée.

A travers cette étude en simulation, il a été démontré que l'approche de Willems et al. [WIL94] et [WIL95] combinée à notre méthode heuristique est efficace en terme de résolution, de vitesse, de convergence de qualité de la solution escomptée ainsi que de la réduction du temps de calcul.

Le troisième chapitre traite une nouvelle heuristique basée sur une méthode de décalage employée pour résoudre le problème d'ordonnancement conjoint de la production et de la maintenance préventive systématique pour un problème d'ordonnancement de type job shop. L'heuristique proposée a pour objectif la réduction du Makespan *Cmax* ainsi que la réduction du coût total de la maintenance par cycle de production *CM* qui est due soit à l'avance ou soit au retard des tâches de la maintenance préventive systématique des machines. Généralement si les tâches de la maintenance sont exécutées en retard (tâches retardées), ceci peut induire un coût de maintenance plus élevé et cela est dû au fait qu'elles augmentent le risque de pannes et produisent un arrêt de production. La méthode proposée est basée sur un ensemble de règles tenant compte des contraintes de production ainsi que les contraintes de partage des machines et les contraintes de l'exécution des tâches de maintenance du problème d'ordonnancement job shop. Pour souligner l'efficacité de la méthode proposée, nous avons appliqué la nouvelle heuristique sur quelques benchmarks d'illustration. Les exécutions du nouvel algorithme concernent le Makespan *Cmax*, le temps libre de chaque machine et quelques autres statistiques liées au coût de la production et de la maintenance *CM*.

En définitive, cette thèse présente deux contributions essentielles soient l'accélération d'un réseau de neurone du type Hopfield en faisant un choix judicieux de son initialisation et une méthode de décalage des périodes de maintenance et de production dans le cas du problème conjoint.

Chapitre I : Notions préliminaires et définitions de base pour l'ordonnancement des systèmes de production

Résumé *: Les problèmes d'ordonnancement sont présents dans tous les secteurs de l'économie car ils présentent une fonction importante en gestion de production. L'objectif de ce chapitre introductif au problème d'ordonnancement est de situer notre problématique de recherche. Des notions préliminaires de quelques définitions de base pour l'ordonnancement des systèmes de production seront présentées au cours de ce chapitre qui est répartie en trois parties.*

La première partie est une initiation au problème de l'ordonnancement job shop. Une présentation générale introductive du problème d'ordonnancement visant un exposé des éléments de base à travers leurs définitions correspondantes au problème de l'ordonnancement du type job shop ainsi que sa modélisation utilisant quelques benchmarks usuels finissant par mettre l'accent sur sa difficulté de résolution. Cette partie a été explicitée pour une formulation générale du problème job shop considéré.

La deuxième partie consiste en une présentation de l'état de l'art et une synthèse bibliographique des méthodes de résolution de l'ordonnancement relatif au problème job shop.

La troisième partie consiste en une présentation succincte de la maintenance des systèmes de production.

Introduction du chapitre I

La théorie de l'ordonnancement est parue au début des années 50 [KIN06]. Les problèmes d'ordonnancement interviennent dans de nombreux domaines, dès qu'il est nécessaire d'organiser et /ou répartir le travail. L'ordonnancement peut donc intervenir dans les projets (construction), les systèmes industriels de production, les systèmes informatiques (problème à *m* machines). De même les problèmes d'ordonnancement se placent au niveau des systèmes administratifs tels que la planification des soins l'affectation des ressources ainsi que la réalisation des emplois du temps et les problèmes liés aux logistiques [FEI09], [YAH06]. C'est la raison pour laquelle, les problèmes d'ordonnancement sont généralement présents dans tous les secteurs de l'économie et constituent une fonction importante en gestion de production. En effet, un problème d'ordonnancement constitue la façon de programmer l'exécution de la réalisation des tâches. En d'autres termes c'est attribuer les ressources requises par les tâches tout en fixant les dates de leurs exécution sans violer aucune contrainte du problème.

Ainsi et à travers cette dernière définition du terme ordonnancement, nous pouvons évoquer plusieurs points intéressants. D'abord que les problèmes d'ordonnancement appartiennent aux problèmes d'optimisation combinatoire d'une part et d'autre part ils sont très variés. Leurs variétés viennent de la diversité des données, des contraintes et des critères d'optimisation qu'ils impliquent. Et ceci varie surtout en fonction du type d'atelier à ordonnancer. Au cours de ce travail nous allons viser en premier lieu l'ordonnancement des tâches dans un atelier de type job shop sans contraintes d'indisponibilité des machines. En deuxième lieu l'ordonnancement des tâches dans l'atelier job shop va prendre en considération la contrainte d'indisponibilité de machines telle que la maintenance préventive systématique qui sera traitée dans le chapitre III et qui constituerait l'ordonnancement conjoint de la production et de la maintenance dans l'atelier job shop.

Dans ce chapitre introductif et afin de situer la problématique de notre travail, nous allons l'exploiter pour mieux voir les particularités du problème d'ordonnancement de type job shop dans le but de formuler le problème à étudier. Ce chapitre est organisé en trois parties.

La première partie vise à présenter une initiation au problème d'ordonnancement job shop. Ceci commence par une présentation générale, suivi par une définition du problème d'ordonnancement. Les principaux éléments du problème d'ordonnancement sont définis afin de les expliciter pour illustrer les différents types d'organisation d'ateliers.

Une étude bibliographique et un état de l'art sur les méthodes de résolutions du problème d'ordonnancement job shop seront abordés dans la deuxième partie de ce chapitre. Ces méthodes de résolutions sont classées en trois types à savoir, les méthodes exactes, les méthodes heuristiques et les méthodes approchées ou métaheuristiques.

La troisième partie de ce chapitre sera réservée à la présentation de la maintenance des systèmes de production. Au cours de cette partie, nous rappelons quelques généralités et définitions normatives sur la maintenance qui sera suivie par une présentation des différentes politiques de la maintenance.

Partie A : Initiation au problème d'ordonnancement type job shop

1. Définitions et outils de modélisation pour l'ordonnancement

Les problèmes d'ordonnancement job shop ont suscité plusieurs travaux publiés dans la littérature et notamment des ouvrages dont nous pouvons citer : [BAK74], [BAK09], [BLA96], [ESQ99], [KIN06], [PIN08].

« Un ordonnancement constitue une solution au problème d'ordonnancement. Il décrit l'exécution des tâches et l'allocation des ressources au cours du temps, et vise à satisfaire un ou plusieurs objectifs» [LOP01]. Ainsi se présente la complexité réelle des problèmes d'ordonnancement, d'où la nécessité de l'existence d'un modèle standard sur lequel est basée la définition du problème. Dans la suite, nous nous intéressons à la description de ce modèle théorique du problème d'ordonnancement.

2. Définition du problème d'ordonnancement

Dans la littérature, nous rencontrons plusieurs définitions pour le problème d'ordonnancement ainsi que ses principales composantes. A travers ces définitions, nous retrouvons l'aspect commun de l'affectation de ressources aux tâches, dans le but d'optimiser un ou plusieurs critères. Parmi ces définitions, nous pouvons citer les suivantes :

« *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches* » [LOP01].

Suivant cette définition énoncée par Lopez, ordonnancer un ensemble de tâches, c'est programmer leurs exécutions en leur allouant les ressources requises et en fixant leurs dates de début. Ainsi dans un problème d'ordonnancement, plusieurs notions fondamentales interviennent entre autre : les tâches, les ressources, les contraintes, les objectifs et les critères. Généralement pour le problème d'ordonnancement de la production, on considère les machines des ressources et les tâches ou aussi « jobs » des activités. Un ordonnancement est la détermination du séquencement des tâches pour chaque machine du système de production correspondant, afin d'optimiser un ou plusieurs critères donnés [RON08].

La fonction ordonnancement est une fonction à court terme, même si elle est parfois utilisée à moyen terme [ESQ99].

Egalement, le problème d'ordonnancement vise à déterminer les dates de début et les dates de fin correspondant à l'exécution de chaque tâche ainsi que les ressources auxquelles elles sont affectées en respectant toutes les contraintes. Alors que nous réservons le terme de problème de séquencement au cas où nous chercherons uniquement à fixer un ordre relatif entre les tâches qui peuvent être en conflit pour l'utilisation des ressources.

L'ordonnancement revêt deux aspects :

- un aspect statique qui ne tient pas compte du temps.
- Un aspect dynamique lorsqu'il s'agit de prendre des décisions en temps réel, compte tenu de l'état des ressources et de l'avancement dans le temps des différents travaux (tâches).

Dans le chapitre II de ce mémoire, nous traitons un problème d'ordonnancement d'un atelier de type job shop statique sans contraintes d'indisponibilités de ressources en utilisant les réseaux de neurones du type Hopfield. Alors que dans le chapitre III, nous étudions l'ordonnancement conjoint d'un problème job shop (statique et dynamique).

3. Définition des éléments du problème d'ordonnancement

Les cinq notions fondamentales intervenant dans un problème d'ordonnancement constituent essentiellement les tâches ou « jobs », les ressources, les contraintes, les objectifs à atteindre et les critères. Dans la suite, nous allons essayer de définir brièvement ces termes et ces terminologies qui seront utilisés dans les autres chapitres de ce mémoire.

A. Les tâches

Une tâche est un travail que nous l'appelons couramment « job » dont la réalisation nécessite un nombre d'opérations élémentaires. Ces opérations élémentaires nécessitent un certain nombre d'unités de temps (sa durée) et l'unité de ressources. Tout au long de ce mémoire, nous utiliserons le terme « job » pour désigner une tâche.

La tâche est aussi définie dans l'**A**ssociation **F**rançaise de **Nor**malisation ainsi : L'AFNOR, a défini sous la norme « NF X50-310 ». Une tâche est le processus le plus élémentaire. Elle est constituée d'un ensemble d'actions à accomplir, dans des conditions fixées, pour obtenir un résultat attendu et identifié, en termes de performances, de coûts et de délais.

Dans le présent travail, nous allons considérer uniquement les tâches non préemptives. Une tâche est dite préemptive, lorsqu'elle est morcelable c'est à dire une fois interrompue suite à un ou plusieurs indisponibilités des ressources, elle peut être reprise sans causer des dégâts ou plutôt sans pénalités. Par contre une tâche est dite non préemptive, lorsqu'elle n'est pas

morcelable c'est-à-dire en d'autres termes l'interruption est interdite dans ce cas (pénalisée). Comme exemple concret, nous pouvons citer le cas de la production laitière ou bien aussi la production des peintures…

La Figure I. 1 illustre les principales contraintes temporelles relatives à une tâche.

Une tâche « *job i* » est localisée dans le temps par les notations suivantes:

t_i : La durée opératoire de la tâche *i* (processing time).

r_i : La date de début au plutôt ou date de disponibilité de la tâche *i* (release time).

d_i : La date de fin au plus tard ou échéance de la tâche *i*.

S_i : La date de début d'exécution de la tâche *i*.

c_i : La date de fin d'exécution de la tâche *i* (completion time).

$l_i = c_i - d_i$: Le retard algébrique.

T_i : Le retard vrai de la tâche $i / T_i = \max(0, l_i)$.

$f_i = c_i - r_i$: La durée de séjour de la tâche *i* depuis sa disponibilité jusqu'à la fin de son exécution (flow time).

De même nous pouvons définir aussi le paramètre ω_i indiquant le poids relatif (importance), (weight), en d'autres termes il définit la priorité de l'exécution de la tâche i pour les clients.

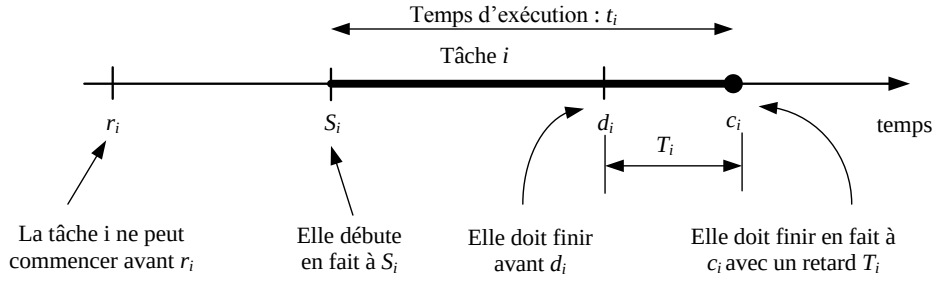


Figure I. 1: Caractéristiques d'une tâche *i* (principaux paramètres)

B. Les ressources

Dans un problème d'ordonnancement interviennent deux éléments principaux : les tâches déjà définies et les ressources. « *Une ressource k est un moyen technique ou humain requis pour la réalisation d'une tâche et disponible en quantité limitée* [ESQ99] ».

Dans le domaine de l'ordonnancement d'atelier les ressources peuvent être des machines, des équipes, ou encore de la matière première, de l'argent, etc. La source la plus importante en plus de machines est habituellement le personnel.

La répartition de la disponibilité d'une ressource *k* par rapport au temps est exprimée par la fonction $a_k(t)$. Cette dernière fonction est une caractéristique qui détermine la capacité de la ressource.

Généralement ces ressources sont de deux types : *renouvelables* et *consommables.* Une ressource est dite renouvelable, si elle redevient disponible avec la même capacité après son utilisation (tels que les machines et les hommes). Par contre une ressource et dite *consommable* si elle disparait après son usage (argent, matière première…).

Parmi les ressources *renouvelables*, il y a ceux qui sont capables de réaliser plusieurs tâches au même temps elles sont appelées des ressources *cumulatives* (nous les rencontrons dans une station de travail composée de plusieurs machines), ce type de ressources est de plus en plus étudié dans les problèmes d'ateliers. D'autres ressources renouvelables sont non partageables c'est-à-dire, elles ne permettent de réaliser qu'une seule tâche à la fois dans ce cas elles sont dites des ressources *disjonctives*.

Nous traitons dans ce mémoire des problèmes à ressources *disjonctives*.

C. Les contraintes

« Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement les variables de décision [ESQ99] ».

La construction d'un ordonnancement admissible (réalisable), doit respecter plusieurs types de contraintes. C'est pour cette raison la résolution des problèmes d'ordonnancement se complique de plus en plus que le nombre de contraintes s'accroit.

L'étude des contraintes rencontrées dans la théorie d'ordonnancement, nous permet de distinguer deux grandes catégories de contraintes : les contraintes temporelles et les contraintes de ressources.

1. *Les contraintes temporelles*

Il s'agit des contraintes les plus intuitives, analogues à celles définissant la succession des tâches dans un atelier. Elles intègrent trois formes

- Les contraintes de localisation temporelle qui concernent les délais de fabrication imposés relatives aux dates limites des tâches. Une tâche B ne peut être exécutée avant qu'une tâche A soit terminée.

Parmi les dates à respecter pour ces dernières contraintes, la date de disponibilité d'une tâche i r_i (déjà définie) ainsi que la date d'échéance d_i.

- Les contraintes technologiques décrivant le positionnement relatif de certaines tâches (gammes opératoires dans le cas d'ateliers). La relation de précédence entre deux tâches i et j (lorsque i précède j), peut être formalisée par l'inégalité ci après : $S_i + t_i \leq S_j$. Par exemple la tâche de couture d'un habillement ne peut pas débuter avant la coupe du tissu.

Une gamme de fabrication précise l'ordre de passage des opérations permettant la réalisation du produit sur l'ensemble des machines ainsi que le nombre d'opérations relatives au produit [TAN07].

Ces deux dernières contraintes seront prises en considération aux chapitres II et III.

- Les contraintes de calendrier liées au respect d'horaire de travail [ESQ99], etc.

2. *Les contraintes de ressources et les contraintes de séquencement*

Ces contraints traduisent le fait que les ressources sont disponibles en quantité limitée (leurs capacité) ; nous parlons également de contraintes de partage [ESQ99]. Ainsi, nous pouvons envisager deux types de contraintes pour les ressources renouvelables :

- Les contraintes de partages de ressources (nous l'expliquerons ultérieurement pour l'atelier de type job shop). Ces contraintes permettent de limiter la capacité des ressources. A ce niveau nous pouvons envisager encore deux types de contraintes de partages de ressources, les contraintes cumulatives et les contraintes disjonctives.
  i. Les contraintes cumulatives doivent être prises en considération, lorsque les ressources nécessaires à un moment d'ordonnancement d'un atelier dépassent les ressources disponibles.
  ii. Les contraintes disjonctives traduisent le fait que deux tâches A et B ne peuvent être réalisées en même temps [ROG92]. Une même machine ne pourra être utilisée simultanément pour deux tâches différentes. Ce type de contrainte sera pris en considération au cours de ce travail.

- Les contraintes de disponibilité de ressources et cela se traduit surtout par la nature et la quantité des moyens disponibles au cours du temps. Les pannes

des machines (arrêts imprévu des machines) ou bien les périodes de maintenances préventives ou correctives (arrêts programmés des machines) sont considérés des contraintes de disponibilité des machines. Dans le chapitre III nous traitons le problème d'ordonnancement conjoint de la production et de la maintenance, d'où nous prendrons en considération cette contrainte.

Plus généralement, il faut tenir compte de ces contraintes et décaler la réalisation de certaines tâches dans le temps.

Dans le cas d'une ressource consommable, seules les tâches ayant une consommation inférieure ou égale à la capacité de la ressource à l'instant t peuvent être exécutées [HAR03a].

Nous pouvons aussi distinguer d'autres types de contraintes suivant qu'elles soient strictes ou pas. Les contraintes strictes sont des obligations à respecter alors que les contraintes dites « relâchables» (aussi appelées préférences) peuvent éventuellement n'être pas satisfaites.

D. Les objectifs ou les critères

Dans la majorité de la littérature un ordonnancement doit optimiser un certain nombre de critères. De même, selon les circonstances du problème d'ordonnancement, les objectifs se trouvent plus ou moins contradictoires et dont l'importance relative est difficile à apprécier. De même, ces objectifs doivent remplir les exigences du credo des ateliers visant le triptyque coût/qualité/délais. La Figure 1.2 modélise ces principaux négociateurs.

Le premier négociateur de l'ordonnancement (délais) concerne la direction commerciale à travers la connaissance régulière de:

- l'état d'avancement et des dates les plus probables de fin de fabrication tout en considérant les retards ou les marges éventuelles.
- la disponibilité des capacités c'est-à-dire évité de donner aux clients des dates incertaines.

Cet objectif correspond à des exigences qualitatives et peut se présenter sous forme de contraintes à respecter comme il a été illustré dans la section précédente.

Le deuxième objectif (négociateur) étant la réduction des en-cours qui se manifestent au niveau de la direction de production dont la fonction principale est la planification et la gestion des ressources industrielles. (Les encours représentent les produits inachevés pendant une période de production).

Le plein emploi des ressources étant le troisième objectif qui vise essentiellement la réduction des coûts de production par une utilisation optimale des ressources par un taux d'occupation élevé tout en réduisant les files d'attente. Ce phénomène est illustré par l'organisation générale des contradictions entre les objectifs de l'ordonnancement à travers la Figure I. 2. Parmi les critères utilisés dans ce sens c'est la minimisation du Makespan $C_{\max}$. Le Makespan étant la date d'achèvement de la dernière tâche de l'ordonnancement.

A fin de récapituler formellement les problèmes d'ordonnancement dans un atelier nous les définissons ainsi: un atelier de production qui constitue un ensemble de ressources sur les quelles un ensemble de jobs ou de tâches devront s'exécuter. Chaque tâche est constituée d'une suite d'opérations nécessitant chacune sa propre ressource. Ainsi une tâche pour qu'elle se réalise, elle doit passer par plusieurs ressources [GHE06].

Le fait de trouver une séquence des opérations élémentaires des différentes tâches sur les différentes ressources disponibles de l'atelier considéré tout en respectant les contraintes (ressources et tâches) ça constitue une solution du problème d'ordonnancement considéré.

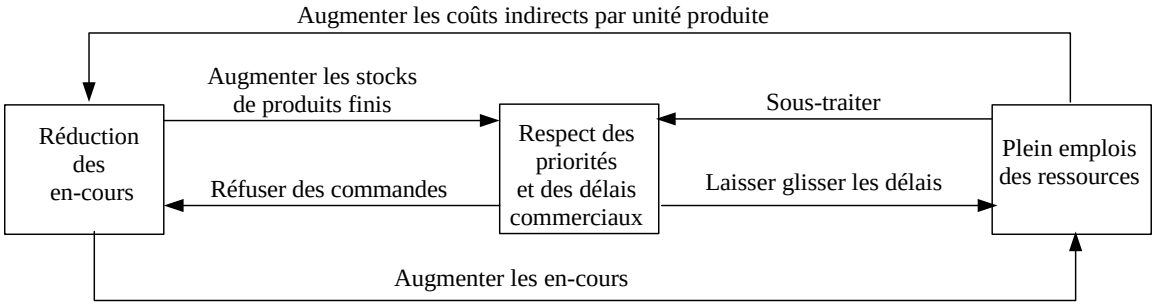


Figure I. 2 : Organisation générale des contradictions entre les objectifs de l'ordonnancement

Pour évaluer la qualité d'un ordonnancement en termes de coût/qualité/délais, nous présentons ci-dessous les principales mesures considérées le plus souvent dans la littérature pour une tâche *i*. Ces critères peuvent être de type *Max* ou de type *Somme*.

- ***Date de fin (makespan) :*** c_i, le critère est $C_{\max} = \max c_i$. Nous chercherons à minimiser la durée globale de l'ordonnancement exprimé généralement par le Makespan.
- ***Durée de séjour :*** $f_i = c_i - r_i$. Il s'agit de minimiser la somme des temps de séjour des tâches, $F = \sum_i f_i$.

- ***Retard :*** $T_i = \max(0, c_i - d_i)$. Il s'agit de minimiser la somme des retards $\sum_i T_i$.
- ***Avance :*** $E_i = \max(0, d_i - c_i)$. Il s'agit de minimiser la somme des avances $\sum_i E_i$.
- ***Indicateur de retard :*** $U_i = \begin{cases} 0 \, si \, c_i \le d_i \\ 1 \sin on \end{cases}$. Le critère à minimiser est le nombre de tâches en retard $\sum_i U_i$ ce qui indique la pénalité unitaire de retard.

Une caractéristique intéressante pour un critère est sa régularité. Un critère est dit régulier, si pour deux ordonnancements S_1, S_2 le fait que chaque tâche de S_1 se termine au plus tard ou en même temps que la tâche correspondante dans S_2, entraine que la valeur du critère de S_1 est inférieure ou égale à la valeur du critère de S_2.

Un ordonnancement est dit admissible ou encore acceptable s'il respecte toutes les contraintes du problème, toutefois il n'est pas optimal.

Dans un ordonnancement semi-actif, aucune opération ne peut être exécutée au plutôt sans changer ou altérer l'ordre relatif de deux opérations sur les machines.

Un ordonnancement est dit actif, si aucune opération ne peut se terminer plutôt sans retarder au moins une autre opération.

4. Différents types d'organisation d'ateliers

Les problèmes d'ordonnancement sont très diversifiés vue l'existence de multiples organisations d'ateliers. Cette diversité implique nécessairement une classification des problèmes d'ordonnancement en deux grandes catégories selon le nombre d'opérations nécessaires à la réalisation de chaque tâche (job).

Les organisations à ressource unique constituent la première catégorie et regroupent les problèmes pour lesquels chaque tâche nécessite une seule opération.

La deuxième catégorie est présentée par les organisations à ressources multiples pour lesquels chaque tâche requiert plusieurs opérations c'est le cas du job shop, flow shop et open shop.

- **Organisations à ressource unique:** l'ensemble des tâches à réaliser qui nécessitent une seule machine. Une tâche est exécutée sous forme d'une seule opération nécessitant ainsi la même machine.
- **Organisations à ressources multiples:** Ces organisations sont caractérisées par l'existence de plusieurs ressources sur lesquelles seront exécutées plusieurs tâches. De ce fait ces organisations sont les plus répandues dans les entreprises

industrielles. Parmi les ateliers les plus intéressants dans l'étude du problème d'ordonnancement, on peut citer 4 types : les Machines Parallèles, le Flow Shop, le Job Shop, et l'Open Shop.

- **Ateliers à cheminement parallèle : Machines parallèles :** Les ateliers à machines parallèles ou non dédiées constituent les machines identiques réalisant les même tâches ou jobs.
- **Ateliers à cheminement unique : Flow Shop :** Les ateliers à cheminement unique, sont caractérisés par un flux de production linéaire dit aussi une production continue. Ce flux de production est souvent nommé « *Flow Shop* » par les anglo-saxons.
- **ateliers à cheminement libre: Open Shop:** Dans les organisations de type *Open Shop*, le cheminement des jobs est quelconque à la différence de l'atelier de type *Job Shop.*
- **Les ateliers à cheminement multiples: Job Shop:** Les ateliers à cheminement multiples concernent la production en discontinu qui est retenue lorsque l'on traite des quantités relativement faibles de nombreux produits variés. L'ordre de passage des tâches ou jobs sur les ressources dans la gamme opératoire peut différer d'un produit à l'autre (tous les produits ne passent pas nécessairement sur toutes les ressources). Un autre cas de passage ou d'exécution des tâches, pour le même type d'atelier, si une tâche doit visiter certaines machines plus qu'une fois, elle est (la tâche) dite recyclée [PIN08]. Le recyclage des tâches est un phénomène souvent rencontré dans la vie courante. Par exemple, à la fabrication de semi-conducteur les tâches doivent recycler plusieurs fois avant qu'ils n'accomplissent tout leur traitement (ce cas de tâches n'est pas traité dans ce travail).

5. Exemples et particularités d'un atelier type job shop

A titre d'exemples, nous pouvons commencer par présenter deux problèmes introductifs illustrant des problèmes réels de type job shop de la vie courante. Le premier exemple est tiré de la littérature [BEN65]. Il concerne le problème d'emploi de temps de quatre auditoires distincts A, B, C, D qui doivent assister chacun, au cours d'une après midi, à quatre conférenciers parmi huit, de la manière suivante telle qu'elle est illustrée par le Tableau I.1.

Conférenciers									
Auditoires		1	2	3	4	5	6	7	8
	A	X	X					X	X
	B	X		X	X		X		
	C		X			X	X		X
	D	X		X				X	X

Tableau I. 1 : Premier exemple d'un problème job shop

A titre d'exemple, la première ligne du Tableau I.1 se lit comme suit: l'auditeur A doit entendre les conférenciers 1, 2,7, 8…en deuxième ligne, l'auditeur B doit entendre les conférenciers 1, 3, 4…

Le problème posé au lecteur est d'établir l'horaire de la session de manière à minimiser la durée.

Nous proposons un deuxième exemple dans le domaine médical. Il s'agit de onze patients venus à une intervention chirurgicale dirigée par deux médecins, *A* et *B*. Leurs consultations sont dotées de plusieurs contraintes. Soit le premier ensemble de patients formé par $\{1,2,3,4\}$ qui veulent voir le médecin *A* avant le médecin *B*. Les patients de l'ensemble $\{5,6,7\}$ veulent voir le médecin *B* avant le médecin *A*. Les patients de l'ensemble $\{8,9\}$ veulent voir uniquement le médecin *A*. Les patients de l'ensemble $\{10,11\}$ veulent voir uniquement le médecin *B*. Les temps de chaque visite (en unités de temps approprié) sont donnés dans le Tableau I. 2.

Le travail demandé est d'établir l'ordonnancement des consultations des onze patients tout en respectant les contraintes imposées.

j	1	2	3	4	5	6	7	8	9	10	11
a_j (médecin *A*)	3	2	1	1	2	4	3	1	2		
b_j (médecin *A*)	2	1	2	1	4	8	9			2	1

Tableau I. 2 : Gamme opératoire des consultations des patients

D'autres particularités caractérisant l'atelier de type job shop sont marquées par l'absence d'un flux dominant de produits, ainsi le système devient producteur d'articles variés exigeant des séquences d'opérations différentes [CRA03]. A titre d'exemple on peut citer :

- atelier traditionnel (mécanicien, menuisier),

- sous-traitance de pièces mécaniques,
- cuisine de restaurant traditionnelle,
- laboratoire d'analyses médicales,
- hôpital qui concerne la planification des salles chirurgicales [PHA08].

De même, ce problème peut modéliser des situations réelles telles que la planification des emplois du temps : des classes pour les salles de classe, des cours de professeurs, les patients à leur lit d'hôpital, et ainsi de suite [FOO88].

En effet, l'atelier job shop peut se différer des autres ateliers par les caractéristiques suivantes à savoir, la grande variété de produits (adaptés aux exigences spécifiques de chaque client); des équipements non spécialisés, peu automatisés et très flexibles (temps de réglage faibles) ce qui implique forcément des investissements en équipements relativement peu élevés et même une main d'œuvre importante. Il a été mentionné aussi à travers des études statistiques du cours de [CRA03] que l'ordonnancement de type job shop revêt un taux d'utilisation des équipements très faible (souvent moins de 50%), ce problème nous allons l'améliorer au troisième chapitre en proposant une nouvelle méthode d'optimisation du problème de l'ordonnancement conjoint de la production et de la maintenance dans le cas d'un atelier de type job shop.

Vu les particularités évoquées sur le problème d'ordonnancement de l'atelier de type job shop, ce problème peut se présenter comme étant le plus complexe parmi les autres types d'ordonnancement d'ateliers. C'est pourquoi ils sont considérés parmi les problèmes d'ordonnancement les plus étudiés dans la littérature et ce, vu leur rapprochement de la réalité industrielle.

La Figure I. 3 illustre un exemple d'un atelier job shop avec trois tâches et six machines indiquant leur séquencement.

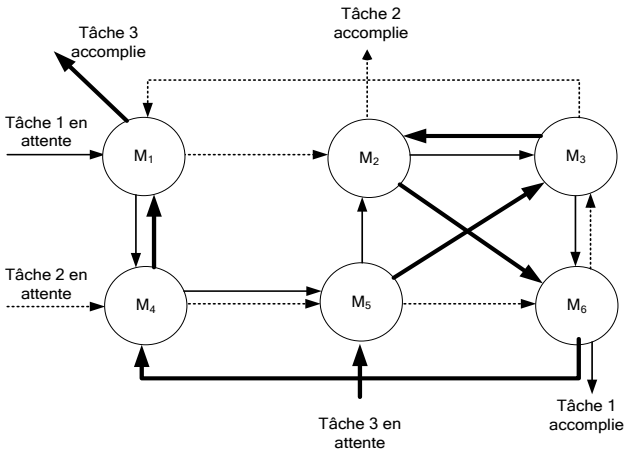


Figure I. 3 : Exemple d'un atelier de type job shop de 3 tâches et de 6 machines

Le job shop est l'atelier ciblé au cours de ce travail, nous allons le traiter avec plus de détail dans la section suivante.

6. Formulation générale d'un problème d'ordonnancement job shop

A. Définitions et hypothèses

Le problème d'ordonnancement de l'atelier job shop (JSSP) représente un cas particulier du problème d'ordonnancement général. Il est le plus étudié dans la littérature. En effet il appartient à la catégorie des organisations à ressources multiples et plus particulièrement il constitue l'ordonnancement dans les ateliers à cheminement multiples. Vu cette multitude de ressource et de cheminement ce problème est considéré le plus complexe à traiter par rapport aux autres ateliers.

La résolution des problèmes d'ordonnancements en général et en particulier ceux de l'atelier de type job shop sont considérés parmi les problèmes les plus difficiles et les plus complexes. Leur difficulté relative se manifeste en fonction des données telles que leurs tailles (nombre de machines et nombre de tâches) ainsi que la multitude de leurs contraintes (conjonctives, disjonctives, disponibilité etc.).

En effet, les données et les contraintes constituent les éléments de bases servant à la modélisation d'un problème d'ordonnancement dans l'atelier de type job shop.

1. *Les données :*

Pour la commodité de formulation du problème d'ordonnancement job shop (JSSP) de taille $(n \times m)$, quelques symboles sont définis comme suit. Il s'agit d'ordonnancer un ensemble de *n* tâches ou jobs $J = \{J_1, ..J_i ..J_n\}$ dans un atelier de type job shop à un ensemble de *m* machines dites aussi ressources $M = \{M_1, ..M_k ..M_m\}$. Ces ressources sont groupées sur la base des opérations qu'elles réalisent: le processus est organisé par fonction. Chaque tâche constitue au plus *m* séquences d'opérations notée ainsi $J_i = \{O_{i1}, O_{i2}, ...O_{im}\}$. O_{ijk} représente la $j^{ème}$ opération de la tâche J_i exécutée sur la machine M_k débutant à l'instant S_{ijk} et lui correspondant une durée opératoire t_{ijk} .

2. Les contraintes

L'objectif de l'ordonnancement d'un atelier de type Job Shop est de trouver l'ensemble des dates de début des opérations S_{ijk} qui respectent les contraintes qui tiennent compte de plusieurs critères.

D'après les principales recherches concernant les problèmes d'ordonnancement classiques notamment le problème de job shop doit respecter des contraintes de deux types :

- Les contraintes de précédences

Ces contraintes définissent un ordre partiel pour l'ensemble de tâches à ordonnancer. En effet si pour deux opérations de la même tâche i, O_{ij} et $O_{i(j+1)}$ sont soumises à une contrainte de précédence ; ceci implique automatiquement que l'opération $O_{i(j+1)}$ doit s'ordonnancer après que l'opération O_{ij} ait été ordonnancée et ceci d'une manière séparée.

- Les contraintes de ressources

Une machine M_k ne peut exécuter sans préemption qu'une seule tâche à la fois. De même une tâche i ne peut s'exécuter que sur une machine unique à la fois. C'est-à-dire une opération de la tâche i est exécutée par la machine k à un instant t, une autre opération de la même tâche ne peut pas être exécutée sur une autre machine.

Nous allons par la suite illustrer quelques hypothèses relatives à l'atelier job shop étudié au sein de ce travail.

- Les machines ne sont pas identiques, chaque machine se présente en un unique exemplaire.
- La préemption des opérations n'est pas autorisée : une fois commencée, l'exécution d'une opération ne peut s'interrompre avant son achèvement.
- Les données relatives au problème d'ordonnancement job shop sont connues d'avance dans le cas d'un ordonnancement statique.
- La disponibilité des machines dépend du cas. Au chapitre II, nous considérons des machines de disponibilité infinie, supposée fiables et ne subissent aucune panne qui peut causer l'exécution des tâches. Dans le chapitre III, nous traitons l'ordonnancement conjoint de la production et de la maintenance c'est-à-dire en considérant le cas de machines qui peuvent présenter plusieurs périodes d'indisponibilité durant l'horizon d'ordonnancement pour des raisons de maintenance préventive systématique.

3. Les critères

Les critères à optimiser dépendent du cas. Au chapitre II, nous considérons la minimisation du Makespan $C_{\max}$ du problème d'ordonnancement de type job shop correspond à la recherche du chemin le plus court entre l'état initial du système, décrit par la liste des tâches (jobs) à exécuter, et l'état final, dans lequel les tâches sont toutes achevées. Au chapitre III, nous ajoutons en plus de ce critère de production ($C_{\max}$) un deuxième critère de la maintenance : la minimisation du coût de la maintenance préventive notée MC .

L'ensemble de ces données, de contraintes et de critères peuvent être explicités pour modéliser le problème d'ordonnancement job shop.

B. Modélisation et représentation graphique des ordonnancements

La modélisation des problèmes d'ordonnancement parait difficile du fait de la multitude de leurs données, contraintes etc. Notamment les difficultés envisagées pour résoudre le problème d'ordonnancement de type job shop et ce, vu leurs caractères fortement combinatoire.

Dans la littérature, les problèmes d'ordonnancement constituent une application concrète dans le domaine de la production industrielle. Cette application est connue par la recherche de l'optimum d'une fonction, parmi un nombre fini de choix, souvent très grand d'une part et d'autre part à travers l'intervention d'un grand nombre de paramètres. Ils existent d'une part diverses façons de modéliser le problème d'ordonnancement job shop conduisant à choisir un mode de description appropriée et d'autre part une grande variété d'approches élaborées pour sa résolution.

La modélisation par graphe disjonctif étant la plus classique et la plus commune et par conséquent elle est la plus utilisée dans la littérature. En contre partie, la modélisation en programmation mathématique qui se présente sous forme de représentation analytique constitue la plus ancienne [BAR99], cette méthode sera étudiée au prochain chapitre. Dans le même contexte il existe également, des modélisations algébriques, polyédrales, par graphes potentiels-tâches, par les réseaux de pétri etc.

Il est à noter pour la modélisation par graphe potentiel-tâche ne peut pas prendre en considération les contraintes disjonctives d'où la nécessité de l'utilisation d'un graphe disjonctif pour modéliser les problèmes d'ordonnancement et en particulier pour l'atelier de type job shop.

1. Modélisation par un graphe disjonctif

La modélisation par un graphe disjonctif a été introduite la première fois par Roy et Sussman en 1964 [ROY64]. La notion de graphe disjonctif a été la base de plusieurs approches de résolution du problème d'ordonnancement job shop. Dans cette sous-section, nous passons en revue sur la modélisation d'un problème d'ordonnancement job shop par la technique des graphes disjonctifs appelés aussi graphes de précédences. De même nous adoptons les notations et les descriptions du problème utilisé par Adams et al [ADA88].

Le graphe disjonctif G modélisant un problème d'ordonnancement job shop est illustré par : $G = (V, C \cup D)$ avec,

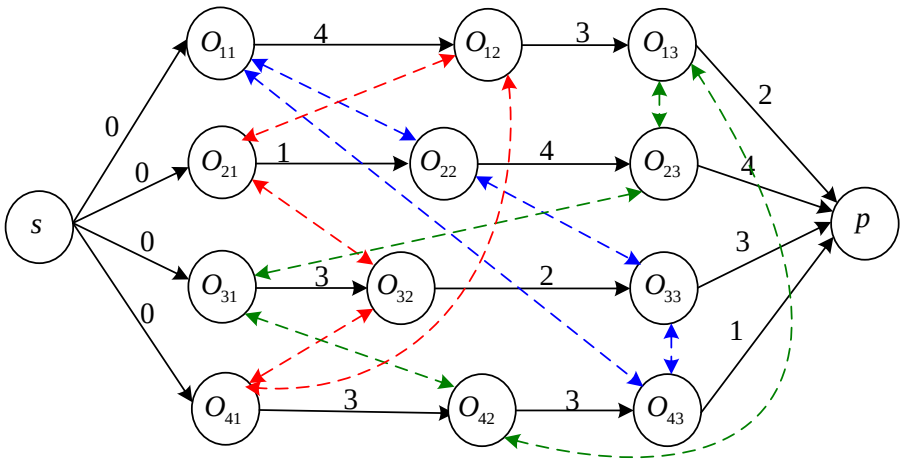
- V est l'ensemble des nœuds qui représentent les opérations des tâches avec deux opérations spéciales (fictives) : une source (s) et un puits (p) modélisant respectivement le début et la fin de l'ordonnancement.
- C est l'ensemble des arcs dits conjonctifs, reliant les opérations d'un même job. Cet ensemble modélise les contraintes de précédences (technologiques) entre les opérations et sont pondérés par les temps opératoires de opérations.
- D est l'ensemble des arcs disjonctifs. Ces arcs relient les opérations n'appartenant pas à la même tâche mais partagent la même machine.

Le Tableau I. 3 indique la répartition des tâches sur les machines (les durées opératoires et la gamme opératoire) d'un problème d'ordonnancement job shop d'un atelier composé de 3 machines de 4 tâches (produits).

La Figure I. 4 illustre une représentation par les graphes disjonctifs utilisant l'atelier décrit par le Tableau I. 3.

	Machine M_k (Temps opératoire : t_{ijk})		
Tâches	Opérations		
	1	2	3
1	M_1 (4)	M_2 (3)	M_3 (2)
2	M_2 (1)	M_1 (4)	M_3 (4)
3	M_3 (3)	M_2 (2)	M_1 (3)
4	M_2 (3)	M_3 (3)	M_1 (1)

Tableau I. 3 : Gamme opératoire d'un job shop de taille 4×3 (4 tâches, 3 machines)



Arcs conjonctifs indiquant les contraintes de séquencement (technologiques).

Arcs disjonctifs reliant les paires d'opérations exécutées sur la même machine.

Figure I. 4 : Graphe disjonctif d'un job shop de 4 tâches et de 3 machines (Tableau I. 3)

Les arcs conjonctifs sont représentés en trait plein, les arcs disjonctifs sont représentés en trait interrompu fin en bleu, en rouge et en vert et ce respectivement aux machines, M_1, M_2 et M_3.

Une fois un ordonnancement est réalisable, il peut constituer (fournir) une solution au problème posé et ceci en fixant un sens d'orientation aux arcs disjonctifs de façon à former un graphe acyclique.

Une telle solution est générée peut être représentée par un outil appelé diagramme de Gantt ce qui constitue la sous-section suivante.

2. Représentation de l'ordonnancement par le diagramme de Gantt

Le diagramme de Gantt peut être considéré comme étant la technique la plus ancienne et la plus répandue (1918) pour visualiser graphiquement l'exécution des tâches d'un problème d'ordonnancement.

En effet, et d'après [PIN08], Henry Laurence Gantt était un ingénieur industriel qui a adopté la discipline de Frederick W. Tailleur. Il a développé ses diagrammes maintenant célèbres pendant la Première Guerre Mondiale pour comparer les ordonnancements de la production avec leurs réalisations. Gantt a présenté son diagramme ainsi que ses principes fondamentaux pour la première fois dans l'annuaire réunion de la société américaine des ingénieurs mécaniciens en 1918. Les diagrammes de Gantt actuellement en service sont typiquement une simplification des originaux, dans le but et dans la conception.

Au cours de ce travail, nous adoptons cet outil (diagramme de Gantt) afin de visualiser et commenter les solutions correspondantes aux problèmes étudiés. Le diagramme de Gantt montre des unités de temps en l'abscisse et les machine indices dans l'axe des ordonnées. Par conséquent, ce diagramme peut être un outil de visualisation des périodes d'occupation et d'inoccupation des machines ou des ressources et également pour mentionner l'ordre de passage des opérations qui constituent les tâches ainsi que la durée totale d'un ordonnancement.

Dans ce diagramme, à chaque opération (O_{ijk}) d'une tâche est associée un rectangle de longueur proportionnelle à la durée opératoire de l'opération (t_{ijk}). Une visualisation de l'exemple précédent décrit par le Tableau 1.3 est illustrée par la figure 1.5. A titre d'exemple l'opération O_{111} est la première opération de la première tâche (J_1) réalisée sur la machine M_1, elle est visualisée sous la forme d'un rectangle ayant comme longueur son temps opératoire. Ainsi son exécution sur la machine M_1 débute à la date $t=0$ et finit à la date $t=4$. La deuxième opération à illustrer est O_{211} qui constitue la première opération de la tâche 2 (J_2), son exécution sur la machine M_1 débute à la date $t=4$ et finit à la date $t=8$.

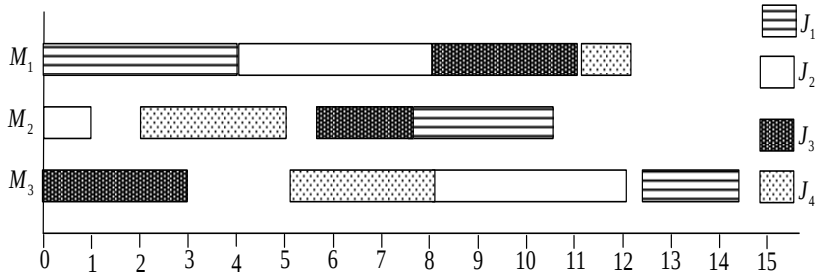


Figure I. 5 : Diagramme de Gantt visualisant un job shop de 4 tâches et de 3 machines

D'après le diagramme de Gantt fourni par la Figure I. 5 le Makespan correspondant à cet ordonnancement est $C_{\max}=14.3$.

C. Validation des modèles sur des benchmarks

Afin de valider les modèles développés et surtout les méthodes et les algorithmes dédiés à la résolution des problèmes d'ordonnancement de type job shop, il est souvent utile de les tester en utilisant des benchmarks.

Un benchmark est un modèle de référence de problèmes-types construits par plusieurs auteurs permettant de mesurer les performances d'une approche de résolution pour la comparer à

d'autres pour le même benchmark appelé aussi instance. Chaque benchmark est décrit par un ensemble de *n* tâches et *m* machines et par conséquent il constitue un problème $(n \times m)$.

Parmi les sources de ces benchmarks disponibles dans la littérature, un site web fourni par Beasley [BEA98] qui constitue une collection d'un ensemble de données pour les problèmes de la recherche opérationnelle (OR-library).

Il existe dans la littérature plusieurs benchmarks de différentes tailles dédiées à tester le problème d'ordonnancement job shop. En effet le plus petit est le *2/3/J/*C_{max} connu par l'utilisation de 2 tâches et de 3 machines dont la valeur optimale du Makespan est $C_{\max} = 22$.

Le Tableau I. 4 présente la gamme opératoire des tâches et les durées opératoires pour le problème *2/3/J/*C_{max}.

Machine (durée opératoire)			
Tâche	opérations		
	1	2	3
1	1(5)	2(8)	3(2)
2	3(7)	1(3)	2(9)

Tableau I. 4 : Gamme opératoire (machines) et durée opératoire des tâches du problème *2/3/J/*C_{max}.

Parmi les benchmarks les plus connus et les plus utilisés pour le problème du job shop, on peut citer les 3 instances nommées mt06, mt10 and mt20 de taille respectivement de, 6×6, 10×10 et 20×5 formulés par Muth et Thompson [MUT63].

Le Tableau I. 5, présente, la gamme opératoire des tâches et les durées opératoires pour le benchmark mt06.

Tâches	Machine (durée opératoire)					
	1	2	3	4	5	6
1	3 (1)	1(3)	2 (6)	4 (7)	6 (3)	5 (6)
2	2 (8)	3 (5)	5 (10)	1 (9)	1 (10)	4 (4)
3	3 (5)	4 (4)	6 (8)	4 (3)	2 (1)	5 (7)
4	2 (5)	1 (5)	3 (5)	6 (4)	5 (8)	6 (9)
5	3 (9)	2 (3)	5 (5)	6 (1)	1 (3)	4 (1)
6	2 (3)	4 (3)	6 (9)	1 (10)	5 (4)	3 (1)

Tableau I. 5 : Gamme opératoire (machines) et durée opératoire des tâches du problème (6×6)

Ces benchmarks présentent des caractéristiques communes telles que :

- Toutes les tâches contiennent un nombre d'opérations inférieur ou égal au nombre de machines *m.* Chaque tâche ne passe qu'une seule fois sur chaque machine.
- Le temps d'exécution de chaque opération est un nombre entier et il en ait de même pour la spécification de la machine sur laquelle a été exécuté.
- Les solutions optimales de ces benchmarks existent dans la littérature d'où la facilité de leurs utilisations dans le but de les expérimenter et de les commenter.

Dans le prochain chapitre, nous utilisons les trois benchmarks décrits précédemment comme problèmes de test à fin de valider la nouvelle procédure d'initialisation pour accélérer la convergence des réseaux de neurones utilisés pour l'ordonnancement Job Shop. De même d'autres benchmarks ont été générés à partir de ces instances en créant des périodes d'indisponibilité pour les machines pour étudier l'efficacité de la nouvelle méthode de décalage pour la résolution du problème d'ordonnancement conjoint de la production et de la maintenance dans un atelier job shop qui sera traitée dans le chapitre III.

Le Tableau I. 6 présente la gamme opératoire des tâches et les durées opératoires pour le problème (10×10).

Tâches	Machines (durée opératoire)									
	1	2	3	4	5	6	7	8	9	10
1	3 (1)	1(3)	2 (5)	4 (8)	6 (3)	5 (7)	7 (5)	8 (8)	9 (8)	10 (4)
2	2 (8)	3 (5)	5 (10)	6 (9)	7 (10)	8 (4)	1 (5)	4 (3)	10 (5)	9 (7)
3	3 (5)	4 (4)	7 (8)	8 (9)	2 (1)	5 (8)	6 (3)	10 (7)	9 (10)	1 (3)
4	7 (5)	8 (5)	2 (5)	1 (4)	3 (8)	4 (10)	10 (7)	9 (4)	5 (7)	6 (10)
5	3 (8)	7 (4)	8 (5)	2 (4)	5 (1)	10 (1)	9 (7)	6 (7)	1 (8)	4 (7)
6	2 (3)	4 (3)	7 (8)	9 (10)	10 (4)	6 (1)	8 (7)	1 (9)	5 (7)	3 (5)
7	5 (7)	6 (7)	3 (7)	10 (5)	9 (1)	4 (10)	7 (10)	8 (4)	2 (3)	1 (9)
8	4 (5)	9 (7)	10(10)	6 (4)	3 (4)	5 (8)	1 (5)	2 (10)	8 (4)	7 (5)
9	5 (3)	10 (8)	9 (4)	6 (7)	4 (7)	1 (5)	2 (9)	3 (5)	7 (10)	8 (10)
10	6 (8)	2 (1)	1 (5)	5 (7)	8 (9)	3 (3)	4 (7)	7 (5)	10 (9)	9 (7)

Tableau I. 6 : Gamme opératoire (machines) et durée opératoire des tâches du problème (10×10)

Dans la partie B qui va suivre, nous allons rappeler quelques méthodes de résolution de l'ordonnancement de l'atelier job shop. Pour cela, nous allons donner un état de l'art des

différentes méthodes dans la littérature ainsi qu'une synthèse bibliographique non exhaustive de ces différentes méthodes.

Partie B: Présentation des méthodes de résolution de l'ordonnancement au sein de l'atelier job shop

1. Introduction et généralités

De nombreuses années de recherche sur les méthodes d'optimisation ont engendré un grand nombre d'algorithmes qui peuvent être appliqués pour générer des ordonnancements, de la programmation mathématique aux recherches qui utilisent des concepts d'intelligence artificielle [JEF06]. Le problème d'ordonnancement job shop a été connu par la communauté de recherche opérationnelle depuis le début des années 50 [JAI98a], [JAI98b]. La première discipline à s'être intéressée à ce type de problème a été la recherche opérationnelle. Il est considéré comme le problème le plus compliqué (NP-difficile au sens fort) des problèmes d'optimisation combinatoires souvent rencontrés dans le milieu industriel. La complexité d'un algorithme de résolution réside soit au niveau du coût correspondant au temps du calcul, ou à l'espace mémoire réservée à cette même tâche (calcul) [KIN06]. Ainsi l'aspect combinatoire résulte du fait que l'ensemble des solutions représente un sous ensemble, ou tout l'ensemble des solutions possibles. En effet, les premiers travaux théoriques pour le problème d'ordonnancement de l'atelier job shop ont débutés depuis les années 50 à travers la création de certains algorithmes important. Pour les ateliers à deux machines, une solution est connue et réalisable en un temps polynomial par l'algorithme de Jackson qui est basé sur les règles de Johnson [JAC56], [JOH56]. Pour les ateliers à deux tâches, le problème de job shop a été résolu optimalement en un temps polynomial par [AKE55] et [HAR63] par une méthode graphique. Pour des problèmes à m machines et n tâches, le problème devient NP-difficile au sens fort [GAR79], c'est-à-dire qu'un résultat exact ne peut être obtenu qu'avec une complexité exponentielle. A titre d'exemple, concernant l'importance de l'espace de solutions correspondant à un benchmark constitué de n tâches et m machines, il existe $(n!)^m$ solutions différentes. Mais le fait d'énumérer toutes ces solutions pour aboutir à une solution optimale n'est pas une tâche triviale. Ainsi si nous prenons par exemple le cas d'une instance de taille (10×10), le nombre de solutions différents correspondante est égale à 39594×10^{65}. La difficulté de la résolution du problème d'ordonnancement job shop ne réside pas au niveau du nombre de machines et du nombre de tâches mais aussi dépend fortement des temps alloués aux opérations correspondantes aux tâches ainsi que le temps de calcul pour trouver ces solutions optimisant un ou plusieurs critères, et par conséquent ce facteur engendre des coûts de calcul énorme ce qui influe le choix d'une méthode de résolution.

2. Méthodes de résolution du problème job shop

Dans cette section, nous allons présenter un bref aperçu des techniques utilisées pour résoudre le problème d'ordonnancement job shop ainsi que les chercheurs impliqués.

L'optimisation combinatoire intègre plusieurs méthodes de résolution des problèmes d'ordonnancement job shop: les méthodes exactes et les méthodes approchées qui intègrent simultanément les méthodes heuristiques et les métaheuristiques. Un état de l'art satisfaisant sur les techniques de résolution du job shop a été publié dans plusieurs références bibliographiques à savoir, Blazewicz et al [BLA96], Jones et Rabelo [JON98], Käschel et al [KÄS99], Brucker [BRU07], Jain et Meeran [JAI99], [JAI98a], [JAI98b], Akyol [AKY07] pour les réseaux de neurones, et récemment Jean-Marie [PRO07] a rappelé brièvement l'évolution de la planification des activités en milieu industriel, il a aussi décrit la situation actuelle et a souligné les problèmes qui appellent des solutions. De même Zobolas et al. [ZOB08] ont aussi présenté les différents types d'algorithmes d'ordonnancement (exacts et approchés) dédiés à plusieurs types d'ateliers et en particulier pour le job shop.

Ainsi, les méthodes exactes visent l'obtention de la solution optimale dans un temps qui augmente exponentiellement avec la taille du problème excepté des problèmes de taille limitée. Ces méthodes sont souvent reliées aux approches d'énumération implicite, telles que la programmation linéaire en nombre mixte, la procédure par séparation et d'évaluation et la programmation dynamique.

Les méthodes approchées sont souvent utilisées pour résoudre des problèmes plus complexes ou de taille plus grande que ceux traités par les méthodes exactes. Les méthodes approchées ne garantissent pas de solutions optimales comme le cas des méthodes exactes, mais plutôt souvent proches de l'optimum.

La flexibilité et l'adaptabilité des méthodes heuristiques aux problèmes complexes ont favorisé la naissance de nouvelles méthodes de cette famille dites métaheuristiques telles que le recuit simulé, la méthode tabou, les algorithmes génétiques et les réseaux de neurones.

A. Les méthodes exactes pour la résolution du job shop

Les méthodes exactes font partie des approches traditionnelles, c'est-à-dire développées avant 1985. Elles sont souvent utilisées pour résoudre des problèmes de petite taille. Parmi ces méthodes, nous allons examiner la programmation mathématique linéaire en nombre mixte, la procédure par évaluation et séparation progressive (PSE) Branch and Bound (BB), et la programmation dynamique.

La programmation mathématique a été appliquée intensivement aux problèmes d'ordonnancement job shop [JON98].

L'approche de la théorie de graphe qui fait partie elle aussi de ces méthodes classiques a été exposée à la partie A. La combinaison de la programmation linéaire avec l'approche de la théorie de graphe et les réseaux de neurones Hopfield constitue une mise au point d'initialisation du réseau de neurone de Willems et al [WIL93], [WIL94], [WIL95], pour le modèle d'un problème d'un job shop déterministe qui sera proposé au chapitre II.

1. *La programmation linéaire en nombre mixte*

Les premières modélisations du problème de job shop ont été des programmes linéaires en nombre entiers ou à variables mixtes. Les premiers travaux datent de 1959 avec les travaux de Wagner [WAG59], Bowman [BOW59] et Manne [MAN60].

D'après Jones [JON98], le problème job shop a été formulé en utilisant la programmation linéaire en nombre entier, [BAL67], la programmation mixte en nombre entier [BAL69], [BAL70] et également la programmation dynamique [SRI71].

Dans la littérature il existe cinq modèles de programmation linéaire en nombres entiers : Le modèle de Manne [MAN60], le modèle de Bowman [BOW59], le modèle de Wagner [WAG59] et les deux modèles de Liao [LIA92].

Ces modèles se caractérisent par l'utilisation de variables binaires en 0 et 1. Le chapitre II illustre les caractéristiques et les détails de la programmation linéaire en nombre entier.

Jusqu'à présent, l'utilisation de ces approches est devenue limitée vue la complexité du problème job shop.

2. *La Procédure par séparation progressive et évaluation (PSE) Branch and Bound (BB)*

La Procédure par évaluation et séparation progressive notée souvent (PSE) ou aussi la méthode de Branch and Bound (B&B) est une technique de recherche qui est paru vers les années 60 [LAN60]. Pour le problème d'ordonnancement job shop de 10 tâches traité sur 10 machines et formulé depuis 1963 par Muth and Thompson [MUT63] est resté ouvert le long de 25 ans. Il est finalement résolu par la méthode de branch and bound par Carlier et Pinson [CAR89]. Egalement McMahon et Florian [MCM75], ont présenté la première application réussie d'une décomposition populaire d'une machine. De même cette méthode est considérée la base des méthodes exactes proposées pour la résolution du problème de type job shop. Ces procédures garantissent l'optimalité mais en une durée exponentielle

L'algorithme de branch and bound est une méthode générique de résolution des problèmes d'optimisation, basée sur l'énumération implicite de l'ensemble des solutions [LAW66]. Elle permet de limiter l'espace de recherche aux régions les plus pertinentes [GHE06]. Cet intérêt de limitation de l'espace de recherche se traduit à travers l'énumération de larges classes de mauvaises solutions. A titre d'exemple, nous pouvons citer quelques travaux d'ordonnancement de type job shop utilisant la présente technique à savoir, [CAS96], [CLA99] et [ART08].

Une description des principes de la méthode Branch and Bound peut être trouvée dans [CLA99]. Un état de l'art sur les différentes méthodes de résolutions a été réalisé dans la référence [JAI98a], [JAI98b] et en particulier pour la méthode de Branch and Bound.

3. *La programmation dynamique*

La technique de la programmation dynamique fait partie aussi des méthodes exactes. Elle est basée sur en une approche récursive qui s'applique généralement aux problèmes dont le temps de résolution est polynomial (NΠ-*difficile*). Cette technique repose essentiellement sur le principe d'optimalité de Bellman [BEL74]. Cette technique est applicable à n'importe quel problème sachant qu'il est décomposable en une séquence de sous-problème. La procédure de résolution se fait itérativement d'une manière ascendante par détermination d'une solution optimale à chaque sous-problème et ce, en tenant compte des informations des sous-problèmes précédents. Cette résolution s'arrête jusqu'à ce qu'elle trouve une solution pour le problème initial, sachant que le critère à minimiser par exemple puisse être mis sous forme d'une relation de récurrence entre n'importe quels pairs de niveaux successifs. Et selon [DEN82], La programmation dynamique est caractérisée par trois types d'équations, à savoir,

- des conditions initiales;
- une relation de récurrence
- une fonction de valeur optimale.

Bien que l'utilisation de cette méthode ait été utilisée pour résoudre des problèmes d'ordonnancement depuis les années soixante, elle n'est pas très répandue pour résoudre le problème job shop, à part quelques travaux, dont on peut citer Chen et al. [CHE98], et ceci est due à son besoin de nombreuses heures de temps de calcul [PIN05].

A travers cette présentation des méthodes exactes, on ne peut pas trouver un ordonnancement optimal en un temps raisonnable pour des problèmes ayant une taille industrielle. D'où l'apparition des méthodes approchées par des chercheurs dont leurs but n'est pas de garantir

l'optimalité mais plutôt de trouver des ordonnancements plus au moins performants selon le type de problème considéré.

B. Les méthodes heuristiques

Les méthodes heuristiques, désormais plus utilisées, leur exécution est un domaine très important de recherche empirique et expérimentale surtout pour le problème job shop. En effet, les méthodes heuristiques sont basées sur une investigation calculée mais qui ne garantit pas l'obtention de la solution optimale. Elles visent la recherche de bonnes solutions dans un temps de calcul raisonnable. Elles sont généralement utilisées pour améliorer l'efficacité d'un processus de recherche sachant qu'elles puissent offrir l'exactitude qui peut induire à l'optimalité de la solution. Compte-tenu du caractère exponentiel des problèmes d'optimisation, elles sont plus adaptées à la solution des problèmes issus de la réalité industrielle. Elles répondent bien au pragmatisme des industriels qui préfèrent plutôt une solution approximative pour un modèle exact à une solution exacte pour un modèle approximatif. Les règles de priorité constituent une classe des méthodes heuristiques dédiée à l'ordonnancement en général et en particulier à l'atelier job shop. Elles ont été intensivement étudiées au cours des 30 dernières années au moyen d'expériences de simulation [HAU89]. De même, et parmi les techniques heuristiques les plus utilisées pour le problème job shop, nous pouvons citer l'approche simulatoire que nous avons opté pour résoudre le problème d'ordonnancement conjoint du chapitre III. L'heuristique de la machine goulet (The shifting bottleneck procedure) constitue aussi une méthode performante pour la résolution des problèmes de job shop [ADA88] et [PEZ00].

1. *Les règles de priorité*

Les règles de priorité sont probablement les heuristiques les plus utilisées pour la résolution des problèmes job shop, et ce pour deux raisons, d'une part vue la facilité (simplicité) de leur implémentation et d'autres part la taille réduite du temps alloué au calcul des problèmes correspondants [BLA07]. Dans une étude qui a été réalisée par Haupt [HAU89] sur l'état de l'art des règles de priorité utilisées pour l'ordonnancement, une règle de priorité est définie comme étant une politique décrivant une décision de séquencement spécifique à chaque fois qu'une machine est libre (n'est pas occupée par le traitement d'une opération). Ainsi une règle de priorité permet à une machine libre de sélectionner l'exécution de la prochaine opération parmi celles disponibles ou en attente. Au cours du chapitre III, nous avons utilisé une règle de priorité pour pouvoir sélectionner une opération parmi celles en attente des différentes tâches. Toutes ces différentes règles de priorités sont bien classées et définies dans [BLA07]

et [HAU89]. A titre d'exemple nous pouvons rappeler parmi les 26 règles de priorités souvent utilisées pour résoudre le problème du job shop, à savoir la règle FIFO : premier arrivé premier servi, la règle SPT : plus courte durée opératoire sur la machine etc.

2. *L'approche simulatoire*

C'est une approche qui fait partie elle aussi des méthodes heuristiques dites à construction progressive d'un ordonnancement. Cette approche a été largement utilisée pour modéliser des systèmes de production suffisamment compliqués et ce en raison de la diversité des paramètres à prendre en compte et de la multitude des contraintes. Parmi ces systèmes de production, celui de l'atelier job shop. Dans la littérature, la simulation à événement discret ou aussi l'approche simulatoire a été fréquemment utilisée pour la modélisation des unités complètement automatisées de fabrication de wafers semi-conducteur (fab) [LEE09], [SUG10]. En effet dans l'ouvrage de [ESQ99] et [CER88], l'approche simulatoire utilise les modèles à événements discret pour résoudre les problèmes d'ordonnancement et en particulier l'ordonnancement job shop. Cette approche permet de modéliser et de simuler le comportement dynamique d'un système de production à travers la circulation de l'ensemble des opérations des tâches correspondantes.

Ainsi, dans le chapitre 6 de l'ouvrage [ESQ99], est définie comme suit: «La simulation à événements discrets consiste à reproduire, à l'aide du modèle précédent, l'évolution de l'état du système au cours du temps sur l'horizon donné». Compte tenue de cette définition, pour pouvoir bien avancer d'une façon progressive dans le temps, il faut tenir compte de l'état de deux types d'évènements. Les évènements exogènes imposés par l'extérieur par exemple l'indisponibilité d'une machine quelconque, et les évènements endogènes qui sont les conséquences des changements d'état du système par exemple le début ou la fin d'une opération. Généralement pour ce type de simulation il ya deux types de moteurs de simulation. Le premier moteur, recherche quelques dates souvent appelées « échéancier » correspondants généralement à des évènements (endogènes ou exogènes) en faisant avancer le temps pour prendre une telle décision correspond à une modification de l'état du système. Le deuxième moteur de simulation, sa tâche consiste à faire avancer le temps par petits incréments en testant à chaque itération (incrémentation) si les conditions sont vraies pour pouvoir passer à l'événement suivant. Cette approche a été utilisée au chapitre III pour modéliser le problème d'ordonnancement conjoint de la production et de la maintenance.

3. Heuristique de la machine goulet « Shifting Bottleneck procedure »

L'heuristique de la machine goulet appelée souvent aussi par la procédure de goulot d'étranglement de déplacement (SPB), a été décrite par Adams et al. [ADA88]. Elle a été parmi les heuristiques les plus puissantes et efficaces pour la résolution du problème du job shop ayant pour objectif la minimisation du Makespan. C'est une heuristique itérative.

Avant de passer pour présenter les principes est les différentes étapes de cette heuristique, il serait intéressant tout d'abord d'expliquer ce qu'on appelle goulot d'étranglement dans le domaine de production. Une ressource ou même une machine est dite goulot est définie par l'étape de production qui a la plus faible cadence dans un flux de production. Ceci nous amène donc à présenter cet aperçu résumé sur cette heuristique qui réside en trois sous-problèmes d'identification. Le premier c'est la sélection du goulot d'étranglement, quand au deuxième c'est l'attribution de la solution à la machine goulet et enfin c'est la réoptimisation de l'ordonnancement. Par conséquent si nous revenons pour expliquer ces étapes, l'heuristique de la machine goulet consiste tout d'abord à diviser le problème initial d'ordonnancement en « *m* » problèmes à une seule machine et résoudre ainsi chacun comme étant un sous-problème d'une manière itérative. Et ceci est basé essentiellement sur la supposition du chevauchement entre les solutions attribuées aux problèmes simples à une seule machine et le problème d'ordonnancement job shop. Par la suite chaque solution d'un sous problème est comparée aux autres solutions et les machines sont ainsi classées en fonction de la comparaison. La machine goulot d'étranglement est ainsi la machine non séquencée avec la plus grande valeur de la solution (exemple un Makespan maximale). Sans tenir compte des autres machines non séquencées, la machine goulet d'étranglement, est prévu sur la base des machines séquencées. Ainsi s'achève cette heuristique par sa dernière étape de réoptimisation locale des machines qui ont été déjà ordonnancées sous forme de problèmes à une seule machine. Récemment Wenqui et Aihua [WEN04] ont amélioré cette procédure heuristique de décalage pour l'ordonnancement job shop dont le but d'améliorer le Makespan qui a été testé sur plusieurs benchmarks.

C. Les méthodes approchées ou métaheuristiques

Les métaheuristiques sont des nouvelles approches qui semblent prometteuses pour la recherche en optimisation combinatoire, et en particulier le problème job shop [YAM03]. En effet, ces métaheuristiques sont des algorithmes d'optimisation qui constituent essentiellement le recuit simulé [YAM95], la méthode tabou [VEL07], [YON07], [GEO08], [ADI10] les algorithmes génétiques [KAM10], [OMB04], [SEN09] [GEO08] [ASA10] et les réseaux de

neurones [FOO88a], [FOO88b], [ZHO91]. D'autres travaux ont été menés avec d'autres métaheuristiques telles que la colonie de fourmi [MON06], les algorithmes mimétiques [HAS09], les méthodes hybrides [ZHA05], [BEN09]…

1. Le Recuit simulé

Le recuit simulé (RS) est une métaheuristique basée sur la recherche locale. L'origine du recuit simulé remonte des expériences de Metropolis et al. [MET53]. Leur algorithme (fonction de voisinage) consistait à étudier la stabilité thermique d'un système physique. Ce n'est que vers l'année 1983 que les trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi [KIR83] ont été les premiers à s'inspirer de la technique du recuit physique pour l'appliquer aux problèmes d'optimisation combinatoire. Ce principe consiste à établir une analogie entre un problème d'optimisation combinatoire et un système physique évoluant à une certaine température. Ce système physique est d'aspect thermodynamique défini par ses degrés de liberté et par son énergie globale *E* qui sera la fonction objectif à minimiser, ainsi qu'un paramètre fictif *T* décrivant la température du système. En effet le recuit simulé est une opération de transformation des métaux. Il est basé sur le chauffage brusque d'un métal jusqu'à fusion, puis le refroidissement lent afin d'obtenir une structure cristalline très pure avec une énergie minimale de sorte que le nombre de fractures et d'irrégularités devient minime [KÄS99]. Ainsi ceci n'est réalisé que si chaque température est maintenue à condition que la matière trouve un équilibre thermodynamique [HAO99].

La simulation du recuit simulé démarre par une solution initiale admissible et continue l'exploration de l'espace d'états en faisant subir à la solution courante des faibles perturbations aléatoires. Si la solution suivante (voisine) obtenue améliore le critère à optimiser, alors elle est retenue. Sinon, si elle provoque une dégradation ΔE du même critère, dans ces conditions la présente solution sera acceptée avec une probabilité : $\Delta E = \exp\left(-\Delta E / T\right)$, avec T : paramètre qui diminue lorsque le nombre d'itérations augmente.

Beaucoup de recherches ont été effectuées sur la résolution du problème d'ordonnancement job shop utilisant le RS. A titre d'exemple Sadeh et Nakakuki ont développé une approche focalisée du recuit simulé, pour résoudre le problème job shop dont le but est la minimisation des critères à savoir, les retards ainsi que les coûts d'inventaires. Steinhofel et al ont présenté deux algorithmes de recuit simulé pour résoudre le problème job shop ayant dont l'objectif est la minimisation Makespan.

Cependant, concernant l'utilisation du RS comme une technique générique pour la résolution du problème d'ordonnancement job shop, d'après [JAI99] il est incapable de converger

rapidement aux bonnes solutions. D'où pour les recherches actuelles, l'utilisation du RS est dirigée vers les méthodes hybrides c'est-à-dire la combinaison de cet algorithme avec d'autres méthodes tel que l'algorithme génétique, la méthode tabou… et ce afin d'améliorer les résultats et réduire le temps de calcul [JAI99].

2. *La méthode tabou*

La méthode tabou est aussi une technique de recherche locale et itérative d'optimisation combinatoire qui fait partie elle aussi des métaheuristiques. Elle a été introduite et développée vers l'année 1986 par Glover [GLO86]. La même idée a été aussi développée d'une manière indépendante par Hansen [HAN86]. Le principe de cette méthode est basé sur la mémorisation d'une fonction mémoire souvent appelée la liste tabou qui représente l'ensemble des états récemment visités et ce afin d'éviter le retour à une solution précédemment visitée. Ainsi de cette façon l'algorithme peut se mettre à jour à chaque itération en lui donnant suffisamment de temps pour lui permettre de sortir de toute vallée contenant un minimum local [CHE06].

Une des approches basées sur la méthode tabou, est celle proposée par Velmurugan et Selladurai [VEL07]. Dans leurs travail, ils ont utilisé des règles de priorités afin d'obtenir une solution initiale pour chercher des nouvelles solutions afin d'obtenir une solution initiale pour chercher des nouvelles solutions pour minimiser le Makespan. Et finalement, la méthode tabou pour les mêmes auteurs [VEL07] a été testée pour des problèmes d'ordonnancement dans les industries de fabrication des pièces d'automobiles.

3. *Les algorithmes génétiques*

Les algorithmes génétiques (AG) ont été développés dés 1950 par les biologistes qui utilisaient des ordinateurs pour simuler l'évolution des organismes. En 1975, John Holland [HOL75] et son équipe, suivis ensuite par Goldberg [GOL89] et Davis [DAV91] ont adaptés ces algorithmes pour la recherche de solutions à des problèmes d'optimisation. Ils développent une analogie entre un individu dans une population et une solution d'un problème dans un ensemble de solutions. Et depuis l'utilisation des AG est très répandu vu leurs efficacité dans divers domaines d'optimisation combinatoire, notamment le traitement d'image, l'optimisation analytique sans oublier l'ordonnancement dans différents types d'ateliers et en particulier le job shop etc.

De même et concernant ce type d'algorithme, l'efficacité de leurs utilisation est soulignée pour la résolution des problèmes d'optimisation multi objectif, ce qui n'est pas le cas pour d'autres algorithmes. Pour les principes fondamentaux des AGs et leurs applications dans une

variété de différents domaines de l'ingénierie et de la science le lecteur peut se référer à un livre récemment réalisé par R. L. Haupt, et S. E. Haupt, [HAU04].

Le principe de base des AGs s'appuie sur la simulation de l'évolution naturelle des *individus,* qui constitue la *population,* par succession de générations, tout en respectant les phénomènes d'hérédité et une loi de survie [CHE06], [ESQ99]. La mise en œuvre d'un AG doit se réaliser dans un espace de recherche composé essentiellement d'un ensemble d'individus. Chaque individu dans une population est caractérisé par son empreinte génétique appelée ***génome***, souvent formée par un ensemble de chromosomes, issue de la recombinaison des empreintes parentales. Cette empreinte est obtenue à l'aide d'opérateurs génétiques dont les plus connus sont le ***crossover*** et la ***mutation***. Le crossover représente un échange des sections des parents des chromosomes, la mutation est une modification aléatoire du chromosome. Elle représente la modification de l'empreinte génétique pouvant se produire sur des individus d'une génération à l'autre et qui évitent la dégénérescence de la population.

Ainsi, et afin d'espérer générer une solution proche de l'optimum, l'ensemble de solutions manipulé par un AG doit fonctionner conformément à un cycle de quatre étapes décrites comme suit :

- Génération d'une population initiale de chromosomes, en associant à chaque solution un code qui représente une empreinte génétique.
- Evaluation de chaque chromosome (calcul des valeurs des critères associées aux individus de la population considérée)
- Sélection des meilleurs chromosomes (l'individu le plus fort de la population c'est-à-dire ayant la meilleur valeur du critère considéré).
- Traitement génétique pour produire une nouvelle population de chromosomes.

Ce cycle d'AG est régi par la mise en place de mécanismes de croisement et de simulation des mutations. L'arrêt de ce cycle est conditionné par un test qui peut être soit par un nombre d'itérations ou bien l'absence de l'amélioration d'un critère à partir d'un nombre d'itérations fixées à l'avance.

Les AGs sont des techniques stochastiques de recherche applicables à de nombreux problèmes d'optimisation. De même et d'après Dorndorf et Pesch [DOR95], les AGs représentent la base des métaheuristiques qui ont servi comme une méta-stratégie permettant d'aboutir à une solution optimale d'ordonnancement job shop basée sur des règles de priorités. A propos des travaux traitant le problème job shop par application des AGs, nous avons trouvé un nombre important de références.

Deux études faites sur l'état de l'art sur l'utilisation des AGs pour les problèmes d'ordonnancement job shop en deux parties successivement en 1996 et 1999 par Cheng et al. Dans la première partie ils ont décrit la représentation d'un problème job shop [CHE96]. En deuxième étape ils ont passé pour décrire les stratégies de la génétique hybride pour le même problème et ce vue que les AGs dans la majorité des cas ne peuvent pas arriver à un résultat optimal dans certains cas difficile [CHE99]. C'est pourquoi les AGs autorisent des synergies avec d'autres méthodes pouvant s'intégrer dans les opérateurs génétiques tels que le recuit simulé et dans ce cas nous parlons plutôt des algorithmes génétiques hybrides.

La première application des AGs pour résoudre le problème d'ordonnancement job shop a été proposée par Nakano et Yamada en 1991 [NAK91]. Mais cette dernière approche n'a pas été assez performante et c'est pour cette raison un an plus tard les mêmes hauteurs ont poussé encore leurs recherche pour construire un ordonnanceur de base qui convient aux problèmes de taille moyenne. Nous pouvons citer à ce stade quelques travaux récents traitant le problème job shop soit avec les AGs seuls [KAM10], [ASA10] ou avec les AGs hybrides [OMB04], [ZHA05], [GEO08] et [THA09].

4. *Les réseaux de neurones*

Un deuxième type d'algorithme inspiré par le calcul dans les systèmes biologiques comme le précédent est le réseau de neurones. Les réseaux de neurones et les techniques connexes sont des outils puissants qui ont prouvé leurs efficacités dans des applications réelles telles que les problèmes d'ordonnancement [AKY07] et les problèmes de maintenance.

Un état de l'art exhaustif sur l'évolution de l'ordonnancement avec les réseaux de neurones a été publié en 2007 par Akyol et Bayhan [AKY07]. Au cours de leur étude, Akyol et Bayhan ont établit une vue d'ensemble sur les réseaux de neurones en classant les approches existantes qui ont paru entre 1988 jusqu'à 2005 en quatre groupes. De même une étude historique progressive a été soulignée à travers les trois types de réseaux qui seront envisagés et finissant par les méthodes hybrides.

Un certain nombre de réseaux de neurones a été développé pour résoudre les problèmes d'ordonnancement et en particulier le problème job shop. Les études existantes pour ce problème sont classées selon trois types de réseaux de neurones, à savoir le réseau de Hopfield, les réseaux de neurones compétitifs (Competitive networks) et les réseaux de neurones à rétropropagation. Mais la plupart des études existantes sont basées sur le réseau de Hopfield.

L'origine du réseau de neurone Hopfield a été introduite par le physicien J. Hopfield en 1982 [HOP82]. Il s'agit d'un modèle à réseau récursif qui est composé de neurones entièrement connectés, capables d'exécuter des tâches de calcul. L'idée de l'utilisation de ce réseau pour résoudre des problèmes d'optimisation NP-difficile a été lancée pour la première fois par Hopfield et Tank [HOP85]. L'application du réseau de neurone de type Hopfield pour la résolution du problème de voyageur de commerce par ces derniers avec succès, a motivé plusieurs chercheurs du domaine de l'ordonnancement à son utilisation.

La modélisation et la résolution du problème d'ordonnancement de type job shop par les réseaux de neurones de type Hopfield a été lancée pour la première fois par Foo et Takefuji [FOO88a], [FOO88a]. L'objectif visé lors de cette étude était la minimisation de la somme de toutes les dates de début de chaque dernière opération de chaque tâche. Le modèle considéré a été un modèle d'ordonnancement statique qui ne peut pas être appliqué pour des problèmes de grande taille.

Une présentation de la totalité des publications pour l'application des RN pour le job shop a été publiée dans [AKY07]. Pour alléger cet exposé, nous avons essayé de résumer cette partie par le Tableau 1.7 qui présente une synthèse bibliographique sur l'évolution de l'application des différents types de réseaux de neurones pour le problème d'ordonnancement job shop.

Références (auteurs)	**Réseau de neurone appliqué**	**Objectifs**
[FOO88a], [FOO88b], [FOO88a], [WIL93], [WIL94], [WIL95], [YAH11a], [ZHO91], [VAN91], [LO93], [SAT94] , [AKY05]	Réseau de Hopfield avec ses extensions	▪ Somme des dates des dernières opérations. ▪ Minimiser le Makespan (C_{max}). ▪ Minimiser la totalité des dates de début de toutes les opérations.
[SAB02], [CHE01]	Réseaux de neurones compétitifs (Competitive networks)	▪ Minimiser le Makespan (C_{max}). ▪ Résolution du problème multiprocesseur sous les contraintes de ressources et le séquencement des opérations.
[PHI94] [GEN97] [JAI98], [SAB02], [FON02]	Réseaux de neurones à rétropropagation avec leurs modifications	▪ Minimiser le Makespan (C_{max}).

Tableau I. 7 : Synthèse bibliographique sur l'évolution de l'application de réseau de neurones pour le problème job shop

Après cette synthèse bibliographique sur les méthodes utilisées pour la résolution du problème d'ordonnancement sans tenir compte des contraintes d'indisponibilité des ressources entre autre la maintenance des systèmes de production ainsi que les politiques adoptées.

Partie C : Présentation de la maintenance des systèmes de production

1. Introduction

Le problème de l'ordonnancement se complique davantage si l'on considère les temps de maintenance des différentes machines.

La complexité croissante des machines et des équipements des ateliers de production est considérée parmi les principales conséquences du développement industriel. Cependant et afin de demeurer compétitif dans ce domaine, les entreprises de fabrication doivent produire des produits toujours de bonne qualité au coût le plus bas possible. Il existe plusieurs manières d'agir à la minimisation du coût, entre autre la fabrication plus vite et évidement sans arrêt des produits sans défauts dans le but d'atteindre la production maximale par unité de temps. Et suite à ces exigences de compétitivité à travers la réduction du coût de la production nécessitant à la fois la rapidité et la continuité de cette dernière, l'automatisation et l'informatique paraissent solutions permettant d'assurer la progression considérable de la rapidité de la production. Cependant et malgré l'augmentation des cadences, cette rapidité admet une asymptote c'est-à-dire qu'il est constamment impossible d'accélérer les rythmes de fabrications. Face à ces exigences, (éviter le ralentissement, les arrêts de la production sans défaut de produits) le système productif doit subir le minimum de temps de non-production en exceptant les arrêts inévitables dus à la production elle-même (changement d'outil, montée en pression, en température…) [AUB04]. Cependant et dans ces conditions les machines doivent produire avec un régime de rendement maximal (pas de pannes).

L'objectif illustré présente l'un des buts à atteindre de la fonction maintenance dans une entreprise. Au cours de ce mémoire nous allons entamer le problème de l'ordonnancement de la production tout en tenant compte des contraintes d'indisponibilité des machines dues aux tâches de maintenances préventives systématiques qui présentent des *trous* pour la production. Dans la sous-section suivante nous allons présenter quelques définitions pour le problème de la maintenance.

2. Généralités sur la maintenance et définitions normatives

Afin de mieux introduire cette bièvre généralité sur la maintenance que nous allons traiter au cours de cette sous section, il parait indispensable de définir quelques termes qui seront utilisés pour évoquer l'historique des différentes étapes d'évolution de la fonction

maintenance. D'après l'ouvrage de [MAS97] ou aussi les définitions normatives tirées des Normes AFNOR X 60-010-1994, nous pouvons évoquer les définitions suivantes :

- **Panne :** un système est en panne s'il est inapte d'accomplir une fonction requise dans ces conditions la production est stoppée.

Remarque : une panne est généralement la conséquence d'une défaillance, toutefois elle peut exister sans une défaillance préalable.

- **Défaillance:** c'est l'altération ou la cessation de l'aptitude d'un bien à accomplir une fonction requise. Une fois une entité est devenue défaillante, celle-ci est en état de panne. Les défaillances peuvent être qualifiées et classées de différentes manières, en fonction de la rapidité de manifestation, du degré d'importance, des causes, des conséquences.
- **Dépannage :** c'est l'action sur un bien en panne en vue de le remettre en état de fonctionnement avant réparation. Le système est remis provisoirement en service. [AFNOR]
- **Réparation :** *« Action définitive et limitée de la maintenance à la suite d'une défaillance »* [AFNOR]
- **Entretien :** c'est un paramètre de conception prévu pour réduire le temps de réparation. Ce terme est très lié à la maintenance et dans des dictionnaires l'entretien peut être confondu à la maintenance. Nous allons éclaircir ce terme par la suite.

En partant de cette dernière définition, nous pouvons illustrer la première différence entre l'entretien et la maintenance selon l'ouvrage [SMI34], [DHI02] par *« L'entretien est un paramètre de conception destiné à réduire le temps de réparation, par opposition à la maintenance, qui est l'acte de réparer ou entretenir un élément ou un équipement »*

Après ce bref survol de quelques définitions de la maintenance, nous allons passer pour présenter quelques généralités sur la fonction maintenance.

Afin d'avoir une idée plus générale sur les différents types de maintenances adoptées dans les systèmes de production, nous l'illustrons par la Figure I. 6 qui constitue une annexe A de la norme NF EN 13306 X 60-319 de juin 2001.

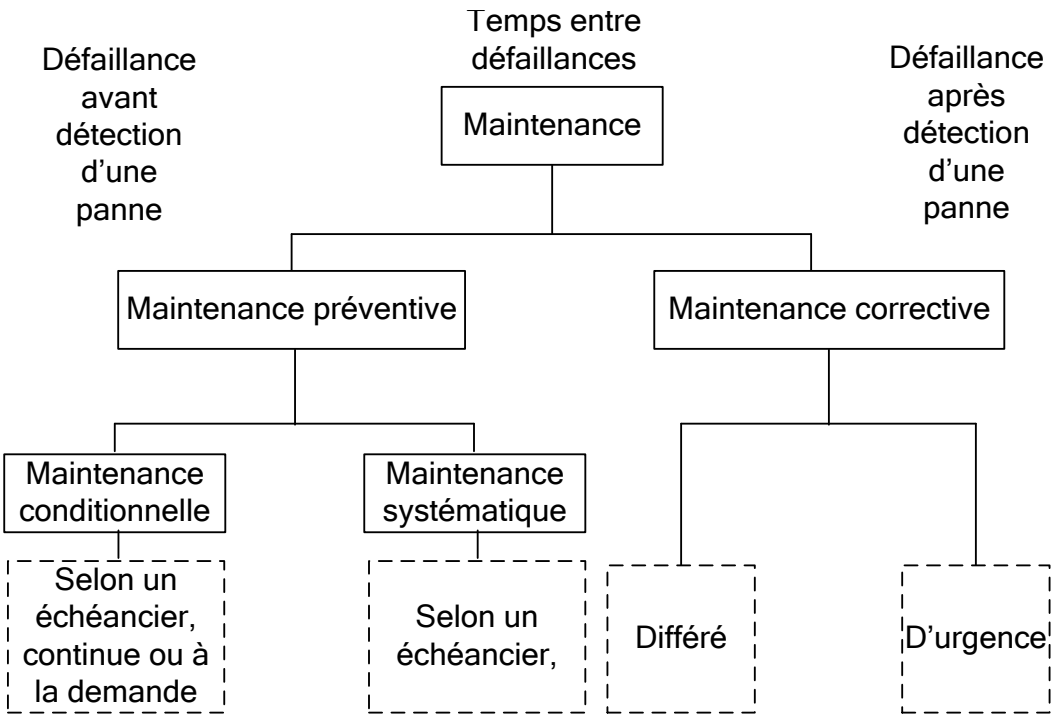


Figure I. 6 : Les différents types de maintenance

3. Les politiques de la maintenance

A travers la présentation de l'historique de la fonction maintenance d'une part et la Figure 1.6 d'autre part, cette activité industrielle parait en plein évolution pour s'adapter continuellement et parfaitement aux contraintes de production et de services. En effet, face aux concurrences industrielles qui se manifestent à plusieurs niveaux, à savoir le développement technologique se traduisant par la variété du matériel utilisé, de même une diversification entre les entreprises et même au sein de chacune d'elle. Dans tout ce contexte, la maintenance en bénéficie à travers la variété des matériels induisant des politiques de maintenance variées et adaptées en fonction des besoins [AUB04]. De même et à travers la définition normative de la maintenance nous pouvons noter que la maintenance regroupe deux actions principalement : *maintenir* et *rétablir*. Ainsi le premier terme désigne une action préventive effectuée en vue de prévenir la panne. Le deuxième terme fait référence à un aspect correctif c'est-à-dire une intervention après la panne. Toutes ces actions permettent de cibler les opérations suivantes dépannage, graissage, visite, réparation, amélioration, vérification, etc. Elles permettent de conserver le potentiel du matériel afin d'assurer la continuité et la qualité de la production ainsi que la sécurité d'opération [MAS97]. Il en découle ainsi deux types de maintenances purement complémentaires : la Maintenance Préventive (MP) et la Maintenance Corrective (MC).

A. La maintenance corrective

La Maintenance Corrective (MC) consiste à intervenir sur un équipement une fois que celui-ci est défaillant et la production est stoppée pour reprendre son activité au moins provisoirement. Dans la littérature la maintenance corrective était qualifiée de maintenance réactive. Cette qualification peut illustrer une intervention réactive suite à un événement tel que aléa, dégradation de qualité de la production, baisse du rendement, défaillance partielle ou complète [AUB04]. Habituellement, la maintenance corrective est une action de maintenance non planifiée [DHI02].

Il en découle qu'au niveau de la maintenance corrective, on distingue deux principaux types de maintenances : la maintenance palliative et la maintenance curative

1. *Maintenance palliative* ***:***

Elle regroupe les opérations de maintenance correctives appelées aussi dépannages, destinées essentiellement à supprimer les effets de défaillances dont le but de reprendre provisoirement une fonction requise. Ces opérations à caractère provisoires de dépannages ne sont pas immédiates pour assurer le bon fonctionnement du système de production, c'est pourquoi elles doivent être rapidement suivies d'opérations de maintenance curatives.

2. *Maintenance curative :*

Elle regroupe les opérations de maintenance correctives appelées aussi réparation dont l'objectif est soit des modifications, soit des améliorations pour une réparation durable et une remise en état d'origine d'une partie concernée d'un système suite à son défaillance. Par conséquent ces opérations font la deuxième phase de dépannage [AUB04].

Il existe aussi un troisième type de la maintenance corrective, c'est la maintenance améliorative qui consiste à se débarrasser définitivement des défaillances à travers l'apport des modifications à la conception d'origine et non pas une simple réparation et ceci dans le but d'étendre la durée de vie des pièces, de réduire la consommation d'énergie, de standardiser des composantes, d'améliorer la maintenabilité [MAS97] etc.

B. La maintenance préventive

La Maintenance Préventive (MP) désigne les opérations de remplacement, de révision dans le but de prévoir la dégradation de l'équipement afin d'éviter ou de réduire les défaillances des systèmes en exploitation.

La maintenance préventive est subdivisée en trois principaux types, elle peut être systématique, conditionnelle ou prévisionnelle.

1. *Maintenance préventive systématique*

Les principales opérations de cette maintenance se présentent sous forme d'inspections périodiques et des interventions systématiques planifiées de deux types. Soit selon un calendrier (à périodicité fixe) ou en fonction du nombre d'heures de fonctionnement, nombre de d'unités produites ou aussi du kilométrage parcourus.

La maintenance préventive systématique concerne quatre différents types d'équipements [NIC02]. Le premier type d'équipements représente ceux dont leur défaillance peut toucher directement la sécurité des biens et des personnes. Le second type d'équipements concerne les équipements dont le coût de leurs défaillances est très élevé et ceci peut être envisagé au niveau d'une production dont les tâches sont pas préemptives par exemple (production laitière, production de la colle, de la peinture…) c'est-à-dire qu'une défaillance entraine directement une grande perte matérielle. Un troisième type d'équipement qui nécessite des opérations de maintenance préventive systématique est celui dont l'arrêt ou le redémarrage est long et ceci est très fréquent dans le cas d'une production nécessitant des chaudières qui fonctionnent sans arrêt (cimenterie, carrelage, céramique…). Enfin un dernier type d'équipement qui sont soumis à des règlementaires (service d'aviation, militaire…).

Dans le troisième chapitre de cette thèse nous avons opté pour l'étude de la maintenance préventive systématique dans le contexte de l'ordonnancement conjoint de la production et de la maintenance dans un atelier de type job shop.

En conclusion il est à noter que ce type de maintenance est facile à gérer vue la périodicité (fixe) de ses interventions car elle permet d'éviter les détériorations importantes et de diminuer les risques graves imprévue. Par contre un inconvénient réside au niveau du phénomène d'usure et de l'âge qui n'est pas pris en considération durant ces interventions systématiques [NIC02].

2. *Maintenance préventive conditionnelle*

La maintenance préventive conditionnelle est considérée la plus intéressante pour suivre l'évolution de l'état d'une machine [MAS97]. De même il est à signaler que l'efficacité d'une telle maintenance dépend directement de la fiabilité des opérations liées aux relevés, mesures et contrôles révélateurs du niveau de dégradation du système. Interventions déclenchées par le franchissement d'un seuil prédéterminé [AUB04]. C'est pourquoi, il faut réserver des moyens

de mesure fiables et spécialisés (capteurs de seuils). Parmi les objectifs attendus de ce type de maintenance [NIC02] :

- Eviter les démontages inutiles dans le but de réaliser les tâches de maintenance systématiques qui peuvent créer des défaillances.
- Augmenter la sécurité du système (biens et personnes)
- Eviter les tâches de maintenance corrective (non programmées) et ceci en suivant les dates de début d'anomalie pour intervenir dans un délai échéancier.

3. *Maintenance préventive prévisionnelle*

La même norme européenne [NF EN 13306 X 60-319] définit ce type de maintenance par : *« Maintenance conditionnelle exécutée en suivant les prévisions extrapolées de l'analyse et de l'évaluation de paramètres significatifs de la dégradation du bien ».*

La maintenance préventive prévisionnelle est appelée aussi maintenance prédictive comme son nom l'indique elle prédit l'occurrence d'une telle défaillance. Son principe est d'analyser d'une manière prédictive ou prévisionnelle l'évolution de l'état de chaque équipement dans le but d'estimer une tendance évolutive d'un disfonctionnement qui sera détecté par un instrument de mesure et donnera une échéance de bon fonctionnement avant que l'équipement tombe en panne. A titre d'exemple l'analyse des vibrations, l'analyse des huiles [AUB04].

4. Conclusion du chapitre I

Ce chapitre nous a permis d'énoncer quelques définitions relatives au problème d'ordonnancement en général dans les systèmes de production. Nous avons énuméré les méthodes existantes de résolution de ce problème que ce soit les méthodes exactes, méthodes heuristiques et les méthodes approchées. La maintenance des ressources dans les systèmes de production influe sur la fiabilité et le coût de la production.

Pour cela, il faut tenir compte de cette composante dans la résolution d'un problème d'ordonnancement.

La plupart des travaux sur ce problème repose sur les méthodes métaheuristiques (réseaux de neurones, Les algorithmes génétiques,…).

Dans le chapitre II, nous allons utiliser une méthode métaheuristique basée sur les réseaux de Hopfield déjà utilisée par Willems *et al.* [WIL94] et [WIL95] que nous avons amélioré dans le but de minimiser le Makespan C_{max} et le nombre de cycles requis pour que le réseau converge vers une solution optimale ou proche de l'optimale.

Pour tenir compte des contraintes de la maintenance, tel que la maintenance préventive systématique, nous avons proposé dans le chapitre III, une autre méthode heuristique pour

résoudre le problème d'ordonnancement conjoint de la production et de la maintenance dans un atelier de type job shop.

Chapitre II : Nouvelles procédures d'initialisation en vue de l'optimisation d'un réseau de neurone Hopfield appliqué aux problèmes d'ordonnancement type job shop

Résumé *: Le présent chapitre vise à présenter notre contribution principale qui est une nouvelle procédure d'initialisation en vue de l'optimisation d'un réseau de neurone Hopfield appliqué aux problèmes d'ordonnancement d'ateliers à cheminement multiples : job shop. Cette nouvelle procédure ou aussi souvent appelée heuristique constitue en une amélioration des algorithmes proposés par Willems et al. [WIL94] et [WIL95]. La démarche consiste en une heuristique de choix optimal d'initialisation des dates de début des opérations afin d'accélérer le réseau de neurones Hopfield (HNN).*

L'avantage de cette nouvelle heuristique est de fournir une amélioration de la convergence du réseau de Hopfield vers une solution optimale ou proche de l'optimale pour le critère à minimiser qui est le Makespan du problème d'ordonnancement des ateliers de type Job Shop.

Les résultats de simulation utilisant cette méthode heuristique améliorée sur plusieurs benchmarks, sont performants. Une étude comparative avec d'autres algorithmes existants, tels que les algorithmes génétiques, les réseaux de neurones adaptatifs de satisfaction de contrainte et les approches combinées par heuristique pour le problème d'ordonnancement des ateliers de type Job Shop généralisé, le recuit simulé, les règles de priorité tel que (SPT) la plus courte durée opératoire, etc., a été élaborée.

A travers cette étude en simulation, il a été démontré que l'approche de Willems et al. [WIL94] et [WIL95] combinée à notre méthode heuristique est efficace en terme de résolution, de vitesse, de convergence de qualité de la solution escomptée ainsi que de la réduction du temps de calcul.

1. Introduction

L'intérêt des réseaux de neurones est davantage marqué par leurs succès d'application pour résoudre les problèmes d'optimisation et en particulier le problème d'ordonnancement des ateliers de type Job Shop qui est connu comme un problème d'optimisation $\mathcal{NP}$-difficile.
Dans le chapitre I, nous avons fait un état de l'art de l'application des réseaux de neurones pour la résolution du problème d'ordonnancement du type job shop. Le tableau (I.7) donne un aperçu de quelques références relatives aux différents réseaux de neurones utilisés pour optimiser la recherche du Makespan C_{max}. Dans ce chapitre, nous allons nous limiter au réseau de neurone du type Hopfield appliqué à ce problème.
Ce type de réseau a été utilisé par Willems *et al*. [WIL94] et [WIL95].
Nous l'avons repris en apportant à celui-ci une nette amélioration dans l'étape de son initialisation. Ce qui a engendré une réduction du C_{max} pour plusieurs exemples (benchmarks).
Dans la suite de la section 2 de ce chapitre, nous allons rappeler succinctement les réseaux de Hopfield et les outils mathématiques utilisés et nous mettons l'accent sur l'étape de généralisation de celui-ci.
Dans la section 3, nous rappelons l'architecture générale du réseau de Hopfield utilisé par Willems *et al*. [WIL94] et [WIL95]. Les améliorations en termes d'initialisation de ce réseau de neurone seront étudiées dans cette section et un organigramme de l'heuristique modifiée sera présenté. Une étude de la simulation sur plusieurs benchmarks sera donnée.

2. Présentation du réseau de neurone de Hopfield [HAY99]

L'origine du réseau de neurone Hopfield a été introduite par le physicien J. Hopfield en 1982 [HOP82]. Il s'agit d'un modèle à réseau récursif qui est composé de neurones entièrement connectés, capables d'exécuter des tâches de calcul. L'idée de l'utilisation de ce réseau pour résoudre des problèmes d'optimisation $\mathcal{NP}$-difficiles a été lancée pour la première fois par Hopfield et Tank [HOP85]. L'application du réseau de neurone de type Hopfield pour la résolution du problème du voyageur de commerce par ces derniers avec succès, a motivé plusieurs chercheurs du domaine de l'ordonnancement à son utilisation.
Nous pouvons décrire le réseau de Hopfield comme étant un système physique dont l'espace de phase contient un nombre fixe de points stables représentant la mémoire fondamentale du système.

A. Caractéristiques fondamentales du réseau de Hopfield

Le réseau de Hopfield, utilise le neurone formel de McCuloch Pitts [MCC43] (voir [HAY99]) comme base de ces éléments.

Chaque neurone a deux états déterminés par le potentiel d'activation

$$S_i = +1 \Rightarrow \textit{état actif} \tag{II.1}$$

$$S_i = +1 \Rightarrow \textit{état non actif} \tag{II.2}$$

Pour un réseau de N neurones, les états du réseau sont définis par le vecteur :

$$S = \left[S_1, S_1, ..., S_N\right] \tag{II.3}$$

$S_i = \pm 1$ est l'état d'un neurone, i représente un bit d'information et le vecteur S est de $N \times 1$ représente un mot binaire de N informations.

Notons que S_i est une forme limitée de la variable d'état x_i sous les conditions suivantes :

- Le temps t tend vers l'infini permet au réseau récursif la relaxation vers un état stable.
- La pente de la non linéarité $\varphi(.)$ à l'origine est infinie.
- C'est-à-dire que la non linéarité est du type fonction signe.

La Figure II. I illustre la structure d'un neurone artificiel.

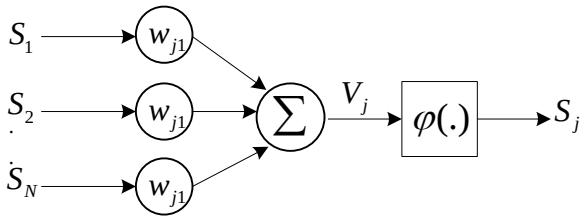


Figure II. 1 : Structure d'un neurone artificiel.

$$S_j = signe\left[V_j\right] \tag{II.4}$$

$$S_j = \begin{cases} +1 \ \ si \ \ V_j > 0 \\ -1 \ \ si \ \ V_j < 0 \\ si \ V_j = 0 \Rightarrow S_j \ reste \, dans \, l'état \ précédent \end{cases}$$

La fonction S_j est illustrée par la Figure (II .2).

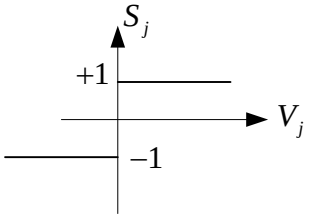


Figure II. 2 : Expression graphique

Le réseau de Hopfield a deux phases de fonctionnement, la phase de stockage ou mémorisation et la phase de généralisation. Ces deux phases déjà citées seront exposées ci-dessous comme suit :

1. *La phase de mémorisation (apprentissage)*

Supposons qu'on veut stocker un vecteur de dimension *N* (mots binaires) noté par $\{\xi_\mu / \mu = 1, 2, ..., p\}$.

Nous appelons les *p* vecteurs, mémoire fondamentale représentant les patterns (i.e. exemples) à mémoriser par le réseau.

Soit $\xi_{\mu,i}$ le $i^{ème}$ élément de la mémoire fondamentale $\mu = 1, 2, ..., p$. Suivant la loi d'apprentissage de Hebb [HAY99], le poids synaptique d'un neurone *i* à un neurone *j* est défini par :

$$w_{ji} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_{\mu,j}.\xi_{\mu,i} \qquad \text{(II.5)}$$

$$w_{ii} = 0 \text{ pour tous } i \qquad \text{(II.6)}$$

L'équation (II.6) indique que chaque neurone n'a pas de contre réaction sur lui-même. Soit W la matrice de dimension $(N \times N)$ correspondant aux poids du réseau et w_{ji} représente son $j^{ième}$ élément. En combinant les deux dernières équations (II. 5) et (II. 6) en une seule équation sous forme matricielle :

$$W = \frac{1}{N}\sum_{\mu=1}^{p} \xi_\mu.\xi_\mu{}^T - \frac{p}{N} I \qquad \text{(II.7)}$$

$\xi_\mu.\xi_\mu{}^T$ est le produit scalaire de ξ_μ par lui-même.

Nous pouvons noter :

- La sortie de chaque neurone dans le réseau est rebouclée à tous les autres neurones.
- Il n'existe pas de bouclage de neurones sur eux-mêmes.
- La matrice de poids est symétrique c'est-à-dire : $w_{ij} = w_{ji} \Rightarrow W^T = W$

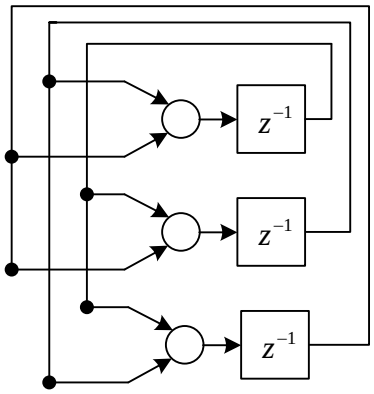


Figure II. 3 : Architecture générale d'un réseau de neurone
Hopfield constitué de 3 neurones

2. *Phase de généralisation (ou relaxation)*

Durant la phase de généralisation, un vecteur X de dimension *N*, appelé « probe » est imposé comme étant un vecteur d'état. Il a des valeurs ± 1. Il représente une information incomplète ou tronquée ou bruitée de la mémoire fondamentale du réseau.

Une procédure de façon aléatoire se lance pour examiner le potentiel d'activation du réseau de neurone.

L'état de sortie du neurone *j* dépend de la valeur de V_j. En effet, nous pouvons envisager trois situations possibles :

- Si $V_j > 0$, le neurone *j* prend l'état+1.
- Si $V_j < 0$, le neurone *j* prend l'état-1.
- Si $V_j = 0$, le neurone *j* reste dans l'état ou il était.

La procédure continue d'un neurone à un autre jusqu'à ce que aucun changement est à considérer. Finalement le réseau produit un état invariant sous forme "*y*" dont les éléments satisfont la condition de stabilité exprimée par l'équation (II. 8).

$$y_j = Sgn\left(\sum_{i=1}^{N} w_{ji} y_i - \theta_j\right) \qquad j = 1, 2, \ldots N \tag{II.8}$$

Ou bien sous forme matricielle par l'équation (II.9).

$$Y = \mathrm{sgn}\left(Wy - \theta\right) \tag{II.9}$$

W : Matrice des poids synaptiques.

θ : Vecteur de tous les biais.

La condition de stabilité est exprimée par la condition d'alignement. Le réseau de Hopfield converge vers un état stable.

B. Exemple [HAY99]

Soit le réseau à 3 neurones de la Figure (II.4).

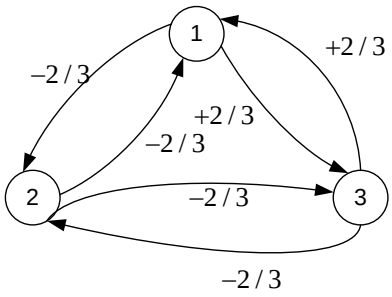


Figure II. 4 : Architecture générale d'un réseau de neurone Hopfield (exemple)

La matrice de poids du réseau est :

$$W = 1/3\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix} \qquad \text{(II.10)}$$

Le biais appliqué pour chaque neurone est nul.

Avec 3 neurones dans le réseau, il existe 2^3=8 possibilités d'états à considérer. A partir de ces 8 états, il ya 2 états stables (1, -1, 1) et (-1, 1,-1) par ce que ces deux états satisfont la condition de stabilité :

En effet,

$$Wy = 1/3\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}\begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = 1/3\begin{bmatrix} +4 \\ -4 \\ +4 \end{bmatrix}$$

En utilisant

$$Sgn[Wy] = \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = y$$

Similairement

$$Wy = 1/3\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} = 1/3\begin{bmatrix} -4 \\ +4 \\ -4 \end{bmatrix} \Rightarrow Sgn[Wy] = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix}$$

Donc ces deux états satisfont les conditions d'alignement. En outre, suivant la procédure asynchrone décrite, nous pouvons avoir le graphe suivant de la Figure (II. 5) :

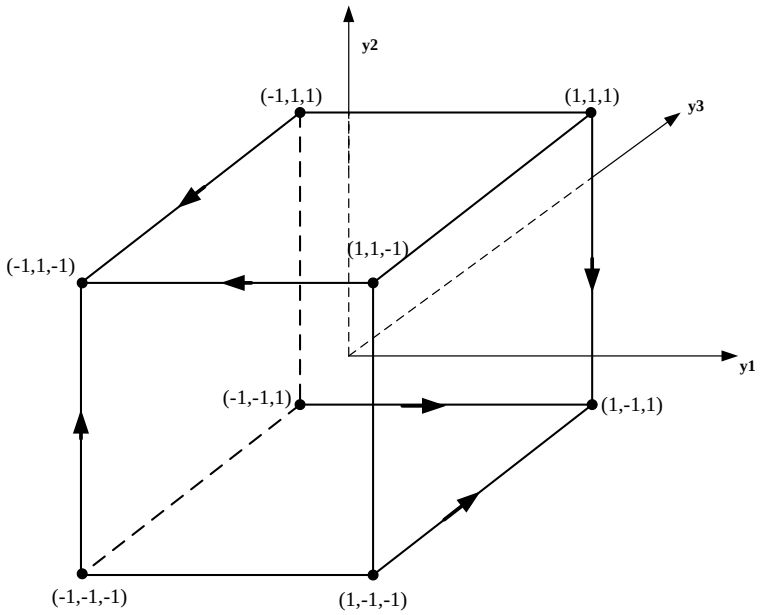


Figure II. 5 : Architecture graphique du réseau de neurone de Hopfield de 3 neurones

Ce graphe montre une symétrie par rapport aux deux états stables. Cette symétrie est due au fait qu'un neurone reste dans son état passé si le potentiel d'activation est nul.

Le réseau a deux mémoires fondamentales (1, -1, 1) et (-1, 1,-1).

Si le réseau est à l'état (1, 1, 1), (-1, -1, 1) et (1, -1, -1) alors il converge après une seule itération vers l'état stable (1, -1, 1).

Si le réseau est à l'état (-1, 1, 1), (1, 1, -1) et (-1, -1, -1) alors il converge vers (-1, 1, -1) en une seule itération.

L'application de l'équation (II.7) donne :

$$W = 1/3\begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix}\begin{bmatrix} +1 & -1 & +1 \end{bmatrix} + 1/3\begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix}\begin{bmatrix} -1 & +1 & -1 \end{bmatrix} - 2/3\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$W = 1/3 \begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}$ On retrouve la matrice des poids synaptiques.

La correction d'erreur se voit aussi sur le graphe de fluence : si le vecteur X appliqué au réseau égal (-1, -1, 1), (1, 1, 1) ou (1, -1, -1) la sortie résultante est alors la mémoire fondamentale (1, -1, 1). Chacune de ces valeurs représente une erreur comparée au modèle stocké de même si X est (1, 1, -1), (-1, -1, -1), ou bien (-1, 1, 1) alors le résultat de la sortie est (-1, 1, -1).

C. Résumé du modèle de Hopfield

1. *Stockage (apprentissage)*

Soit $\xi_1, \xi_2, ... \xi_p$ N-dimensionnel mémoire.

Construire le réseau en utilisant la loi du produit scalaire (c'est-à-dire la loi de Hebb).

$$w_{ji} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^{p} \xi_{\mu,i} . \xi_{\mu,j} & j \neq i \\ 0 & j = i \end{cases} \qquad \text{(II.11)}$$

2. *Initialisation*

Soit X un vecteur d'entré présenté au réseau.
L'algorithme est initialisé par :

$$S_j(0) = x_j \qquad j = 1, ..., N \qquad \text{(II.12)}$$

Où $S_j(0)$ est l'état du neurone j au temps $n = 0$ et x_j est le j^{ieme} élément du vecteur d'entrée X .

3. *Itération jusqu'à la convergence*

Ajuster les éléments du vecteur $S(n)$ d'une façon asynchrone suivant :

$$S_j(n+1) = Sgn\left[\sum_{i=1}^{N} W_{ji} S_i(n) \right] \qquad \text{(II.13)}$$

Répéter la procédure jusqu'à ce que l'état S reste inchangé.

4. Sortie

Soit S_{fixed} dénote l'état stable calculé à la fin de l'étape 3. La sortie du réseau est donnée donc par l'équation (II.14) suivante.

$$Y = S_{fixed} \quad \text{(II.14)}$$

3. Réseau de neurone Hopfield appliqué au problème d'ordonnancement job shop

A. Description du problème et notations utilisées

Après cette brève introduction du réseau de Hopfield dans la section précédente, nous allons appliquer ce réseau pour résoudre le problème job shop.

Dans ce chapitre, Le problème traité est un problème d'ordonnancement job shop est du type déterministe et statique. Ce problème est considéré NP-difficile [GAR79] malgré la simplicité de sa modélisation en définissant à la fois, les données (tâches et machines), les contraintes et les objectifs. C'est la raison pour laquelle, ce problème est devenu un terrain d'essai privilégié pour tester plusieurs méthodes exactes, heuristiques et notamment les nouvelles approches de résolution qui constituent par exemple les métaheuristiques que nous avons exposé au chapitre I.

Dans le présent chapitre [YAH11a], nous utilisons une métaheuristique utilisant un réseau de neurone Hopfield pour modéliser le problème d'ordonnancement job shop.

On rappelle que la résolution du problème d'ordonnancement consiste à ordonnancer c'est-à-dire programmer ou planifier dans le temps l'allocation d'un ensemble de *n* tâches (ou jobs) sur un ensemble de *m* ressources (ou machines) selon une gamme opératoire donnée tout en respectant un certain nombre de contraintes dans le but de minimiser un ou plusieurs critères.

Ce travail constitue une extension du travail de Willems et al [WIL94] et [WIL95] qui ont utilisé des règles de priorités permettant de trier à un instant *t* les opérations candidates à une machine quelconque [YAH11a]. C'est-à-dire une fois une tâche est arrivée sur une machine qui ne peut pas l'exécuter (la traiter immédiatement) vue son indisponibilité, la tâche sera placée dans une file d'attente. Lorsque la machine sera disponible à exécuter une tâche, et la file d'attente comprend plusieurs opérations, nous utilisons une règle de priorité pour résoudre le conflit de décision. Une règle de priorité représente une formule associant une valeur à chaque opération d'une file d'attente, qui est calculée en général sur les paramètres de l'opération.

En effet de nombreuses règles de priorités conçues pour l'ordonnancement de la production

ont suscité beaucoup de recherches pendant plusieurs décennies. Malgré les études qui ont été faites pour déterminer les effets de ces règles sur l'ordonnancement et leurs performances globales par rapport à quelques critères, il s'est avéré qu'aucune règle ne surpasse les autres sur tous les objets et dans toutes les configurations d'ateliers.

Ces règles sont utiles lorsque l'on tente de trouver un ordonnancement relativement bon par rapport à un objectif unique tel que le Makespan. La classification de ces règles est faite selon plusieurs critères. Selon un état de l'art fait par Haupt en 1989 [HAU89], parmi les critères de classification des règles de priorité nous pouvons rappeler par exemple selon que les règles statiques ou dynamiques. Une deuxième façon de les classifier selon qu'elles sont locales ou globales.

Une règle de priorité est considérée statique si la valeur de la priorité est indépendante du temps, mais elle est juste une fonction des tâches ou des données des machines. A titre d'exemple nous pouvons donner l'exemple de la règle de priorité **(SPT)** qui s'exprime par (short processing time first) c'est-à-dire la décision au niveau de la file d'attente est affectée à l'opération ayant le plus court temps opératoire. D'après [BOU91], la règle **SPT** a donné à priori de bons résultats bien qu'ils ne sont pas les meilleurs. Par contre une règle est dynamique, si la valeur de la priorité dépend du temps. A titre d'exemple une règle de priorité peut accorder la priorité à une tâche *i* à un instant *t*, alors qu'à un autre instant cette même tâche peut être de même priorité avec une autre tâche *p*.

Le deuxième type de classification concerne la distinction entre une règle locale et une règle globale. Ceci se spécifie par la nature des informations dont ces règles sont basées à ce moment. En effet une règle locale requiert deux types d'informations locales [PIN08]. Elle constitue les informations relatives à la file d'attente où les tâches sont en attente ou bien à la machine sur laquelle la tâche est en en file d'attente. Alors qu'une règle est globale si elle prend en considération les informations relatives à d'autres machines.

Ainsi nous commençons tout d'abord par rappeler les règles de base qui ont été appliquées dans leurs travaux [WIL94] et [WIL95] afin de limiter l'espace de recherche.

La liste ci-dessous des ces règles de priorité, constitue des règles statiques, locales, et liées aux tâches.

1. (Earliest possible starting time first): la date de début au plutôt possible : les opérations des tâches qui peuvent être ordonnancées les premières sont ordonnancées les premières.

2. (Earliest possible ending time first): la date de fin au plutôt possible : les opérations des tâches qui peuvent s'achever au plutôt : c'est-à-dire les opérations ayant le plutôt (date de début+temps opératoire) sont ordonnancées les premières.
3. (Shortest processing time first): il s'agit d'ordonnancer les tâches dans l'ordre croissant des temps opératoires.
4. (Longest processing time first): il s'agit d'ordonnancer les tâches dans l'ordre décroissant des temps opératoires.
5. (Shortest remaining processing time first): il s'agit d'ordonnancer les tâches dans l'ordre décroissant des temps de traitement restant.
6. (Shortest remaining processing time first): il s'agit d'ordonnancer les tâches dans l'ordre croissant des temps de traitement restant.

Le présent problème sera formulé en utilisant l'un des principaux outils de modélisation en recherche opérationnelle qui n'est autre que la programmation linéaire et plus précisément la programmation linéaire en nombre entiers dont nous rappelons quelques données sur cet outil.

La programmation linéaire en nombre entiers (PLNE) est d'habitude conçue pour résoudre des problèmes NP-Complets. De même, elle fait partie aussi des principales méthodes avancées de la programmation mathématique. Elle consiste à optimiser une fonction linéaire à plusieurs variables pour trouver un extrémum (maximum ou minimum). Cette fonction linéaire doit exprimer un ensemble de contraintes sous forme d'égalités ou d'inégalités.

Le Tableau (II. 1) présente et définit les notations standards utilisées par Willems *et al* [WIL93], [WIL94] et [WIL95] pour formuler le problème d'ordonnancement job shop. Nous utilisons les mêmes notations dans le reste du chapitre.

Symbole	Désignation
n	Nombre de jobs ou de tâches
i, p	Indices de jobs ou de tâches
J	Job ou tâche
m	Nombre de ressources ou de machines
k, l, f, s, h	Indices de ressources ou de machines
j, k, r	Indices des opérations
O_{ijk}	Opération j du job i exécutée par la machine où la ressource k
t_{ijk}	Durée opératoire de l'opération O_{ijk}
S_{ijk}	Date de début de l'opération O_{ijk}
H	Une constante
Y_{ipk}	Une variable qui indique quel job qui s'exécute sur la machine k
W_f	Un poids positif de la rétroaction mettant à jour les raccordements (par exemple +0.1)
$n / m / J / C_{max}$	Représente le problème d'ordonnancement selon le système de notation de Conway *et al* [CON67].

Tableau II. 1 : Notations utilisées du problème d'ordonnancement de type job shop

Le problème d'ordonnancement job shop considéré sera défini comme suit :

Il s'agit d'ordonnancer n jobs ou tâches $\{J_1, J_2, ..J_n\}$ dans un atelier de type job shop à m machines. L'objectif se ramène à trouver l'ensemble des dates de début optimales S_{ijk} des opérations qui respectent certaines contraintes tout en tenant compte d'un seul critère d'optimisation. Ce critère n'est autre que la minimisation du Makespan C_{max}. Ceci revient à la recherche du chemin le plus court entre l'état initial du système, décrit par la liste des jobs à exécuter, et l'état final, dans lequel les jobs sont tous achevés. D'où il est crucial de définir ces contraintes pour pouvoir formuler mathématiquement le problème d'ordonnancement considéré.

En fait, la réalisation de ces tâches se fait en passant par différents types de machines qui ne peuvent réaliser qu'une seule tâche à la fois.

Chaque tâche est définie par une séquence d'opérations appelée aussi gamme opératoire et ceci tout en respectant les contraintes temporelles et les contraintes liées à la disponibilité des machines requises par cette tâche.

Toutes les tâches sont disponibles à l'instant $t=0$. En outre, on suppose qu'il n'ya pas de

chevauchement entre les opérations et les tâches c'est-à-dire qu'une opération ne peut commencer que lorsque les opérations qui la précèdent ont été accomplies. Les durées opératoires des différentes opérations et des différentes tâches sont connues d'avance. De même les machines sont supposées toujours disponibles, leurs pannes ainsi que leurs périodes de maintenance ne sont pas pris en considération dans cette partie. La contrainte d'indisponibilité des machines pour des périodes de maintenance sera traitée dans le chapitre suivant. De même, on considère qu'il n'ya pas de préemption d'opérations.

Au cours du problème considéré deux types de contraintes seront considérées, les contraintes de précédence (temporelles) et les contraintes de partage de ressources. Ces deux types de contraintes sont modélisées sur le graphe disjonctif d'un job shop introduit par Roy et Sussman en 1964 [ROY64] illustré par la figure 1.3 du premier chapitre sur un graphe disjonctif d'un job shop de 4 tâches et de 3 machines. Sur ce graphe qui modélise les contraintes citées, nous rappelons que les arcs conjonctifs indiquent les contraintes de séquencement (technologiques) et les arcs disjonctifs relient les paires d'opérations exécutées sur la même machine.

B. Modélisation du problème job shop par la programmation linéaire en nombre entier

Comme la programmation linéaire en nombre entier exige la mise en évidence des contraintes linéaires qui peuvent être exprimées par des égalités on inégalités, nous commençons en première étape par rappeler les contraintes de séquencement ou de précédence représentées par les arcs conjonctifs [ROY64] (voir Figure (1.4) du chapitre I) qui signifient que les jobs doivent être traités sur des machines dans l'ordre fixé et définis par les concepteurs de processus de planification. Concrètement, l'opération *j* du job *i* supposée à la machine (où la ressource) *l* doit être traitée après l'opération (*j-1)* du job *i* supposée à la machine *k.* Ainsi pour que l'ordonnancement soit réalisable, il est nécessaire de respecter cette inégalité $S_{ijl} \geq S_{i(j-1)k} + t_{i(j-1)k}$.

Ce qui peut être reformulé par l'équation (II. 15) qui exprime la contrainte de précédence (ou de séquencement) ci-dessous :

$$S_{ijl} - S_{i(j-1)k} + t_{i(j-1)k} \geq 0 \qquad \text{(II.15)}$$

En effet, il existe $n(m-1)$ équations de contraintes de séquencement pour un problème d'ordonnancement d'un atelier de type Job Shop de *n* jobs et de *m* machines (*n / m/ J/ B).*

Généralement la contrainte de partage de ressource est d'un type disjonctif c'est-à-dire que chaque machine ne peut réaliser qu'une opération à la fois et chaque opération O_{ijk} nécessite

une seule machine à la fois, durant sa durée opératoire t_{ijk}. Cette contrainte est présentée par les deux équations (II.16a) et (II.16b). Pour deux opérations O_{plk} et O_{ijk} partageant la même machine k, les contraintes suivantes doivent être respectées afin d'éviter le chevauchement (à temps) entre ces opérations :

$$S_{ijk} - S_{plk} - t_{plk} \geq 0 \qquad \text{Si l'opération } O_{plk} \text{ est exécutée en premier lieu} \qquad \text{(II.16a)}$$

$$S_{plk} - S_{ijk} - t_{ijk} \geq 0 \qquad \text{Si l'opération } O_{ijk} \text{ est exécutée en premier lieu} \qquad \text{(II.16b)}$$

En utilisant une variable de décision booléenne Y_{ipk} qui indique quel est le job qui est exécuté en premier lieu par la machine k, définie par le système d'équation (II.17):

$$Y_{ipk} = \begin{cases} 1 & si \quad S_{ijk} \leq S_{plk} \\ 0 & si \quad S_{ijk} > S_{plk} \end{cases} \qquad \text{(II.17)}$$

Les contraintes de partage des ressources deviennent :

$$S_{plk} - S_{ijk} + H\left(1 - Y_{ipk}\right) - t_{ijk} \geq 0 \qquad \text{(II.18)}$$

$$S_{ijk} - S_{plk} + HY_{ipk} - t_{plk} \geq 0 \qquad \text{(II.19)}$$

Où H représente un nombre positif arbitraire plus grand que la valeur maximale de toutes les durées opératoires t_{ijk} des opérations des jobs et ceci en respectant les contraintes données par les équations (II.18) et (II.19). La constante H peut être calculée par l'équation (II.20) en ajoutant toutes les durées opératoires (dans le cas du plus mauvais ordonnancement) avec :

$$H > \sum_{i=1}^{i=n} \sum_{j=1}^{j=m} t_{ijk} \qquad \text{(II.20)}$$

Il existe $nm\left(n-1\right)$ équations de contraintes de partage de ressources pour un problème d'ordonnancement d'un atelier de type Job Shop de n jobs et de m machines ($n / m/ J/ C_{max}$) pour lequel la fonction coût (critère à minimiser) est le temps d'achèvement de la dernière opération, ce qui implique la minimisation du Makespan C_{max}. La solution résultante des deux types de contraintes (contraintes de précédence et contraintes de partage de ressources) est un ensemble de $n\left(m-1\right)$ équations pour un problème d'ordonnancement $n / m/ J/ C_{max}$.

Les équations (II.18) et (II.19) représentent les contraintes de ressources qui génèrent un modèle de programmation linéaire en nombre entier qui sera résolue par un réseau de neurone du type Hopfield.

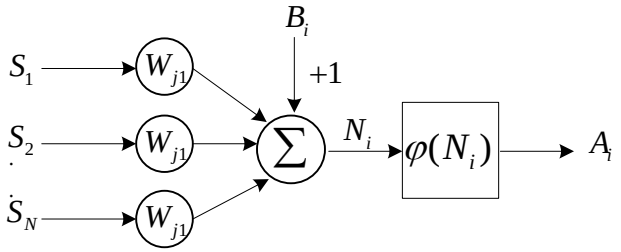
4. Résolution de la programmation linéaire en nombre entier du problème d'ordonnancement job shop par un réseau de neurone Hopfield

Le problème d'ordonnancement job shop représenté par le modèle de la programmation linéaire en nombre entier se prête bien pour l'application de réseaux de neurones [YAH11a]. En effet, en utilisant une structure neuronale de type Hopfield utilisant des neurones de types Macculoch et Pitts bouclés, nous pouvons modéliser le problème de job shop sachant que les sorties du réseau représentent les dates de début S_{ijk}.

Dans la suite de ce chapitre, nous allons présenter les neurones utilisés dans le réseau. Il est à noter que le réseau de Hopfield adopté comporte des sous réseaux de neurones appelés unités (*S, CS, CRI* et *CRII*) qui représentent respectivement les dates de début, les contraintes de séquencement et les deux contraintes de ressources.

A. Modèle d'un neurone formel adopté

Le neurone ou unité ou cellule est définie par la sommation linéaire pondérée des signaux d'entrée connectés en série avec la fonction d'activation non linéaire $\varphi(N_i)$ comme le montre la Figure (II. 6). L'état de cette unité représentée par (l'unité-S) représente la date de début S_{ijk}.



$$\begin{cases} N_i = \sum_{j=1}^{n} \left(W_{ij} A_j \right) + B_i \\ A_i = \varphi\left(N_i \right) \end{cases}$$

Figure II. 6 : Modèle général d'un neurone

B. Modèle de contrainte de séquencement

L'équation (II.15) qui modélise les contraintes de séquencement est illustrée par l'unité-*CS* représentée par la Figure (II.7).

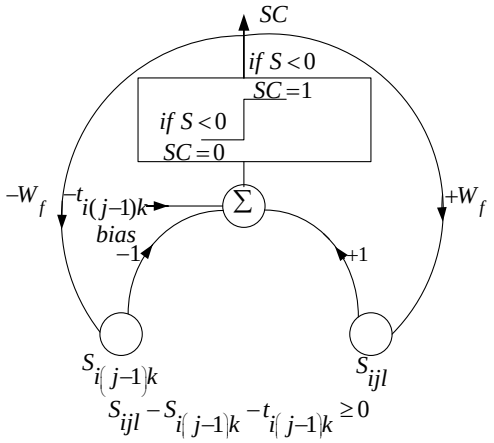


Figure II. 7 : Réseau de Hopfield illustrant les contraintes de séquencement unité-*CS*

C. Modèle de contrainte de ressource

Pour les contraintes de ressources, elles peuvent être représentées par deux unités de neurones, unité-*CRI* et unité-*CRII* (Figure (II. 8) et Figure (II. 9)) qui découlent respectivement des équations (II.18) et (II.19).

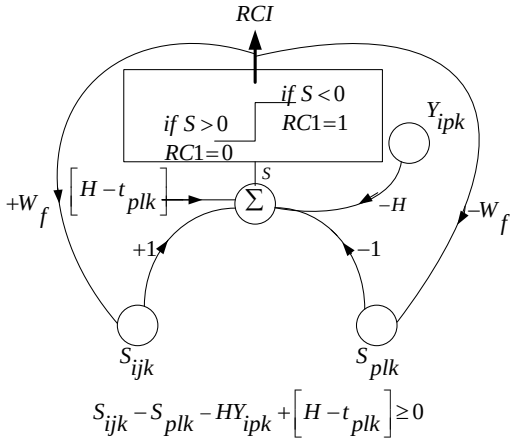


Figure II. 8 : Réseau de Hopfield illustrant les contraintes de ressources *CRI*

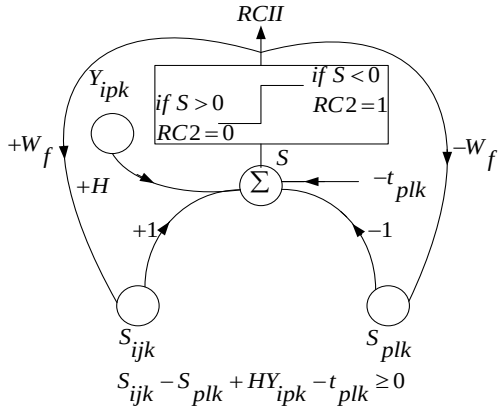


Figure II. 9 : Réseau de Hopfield illustrant les contraintes de ressources *CRII*

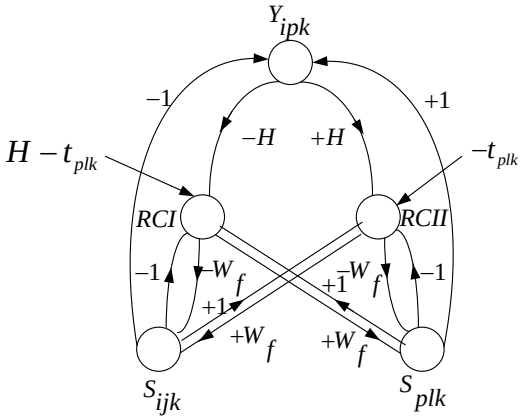


Figure II. 10 : Architecture générale des contraintes de ressources *CRI* et *CRII*

La Figure (II.10) représente l'architecture générale pour les contraintes de ressources *RCI* et *RCII*.

D. Modèle général des contraintes de ressources

Toutes les unités du modèle représentant l'architecture générale de la programmation linéaire du problème d'ordonnancement de type job shop, qui inclut les différentes unités : *S-unité, CS-unité, RCI-unité, RCII-unité* et *Y-unité* sont interconnectées et représentées par deux schémas bloc SC-bloc pour les contraintes de séquencement et RC-bloc pour les contraintes de ressources.

La simulation du modèle de l'architecture générale des contraintes de ressources obtenue peut nous fournir plusieurs ordonnancements (optimaux, faisables, ou non) et ceci dépend du choix de l'initialisation des dates de début S_{ijk}.

Donc la solution correspondante à ce problème réside à ce niveau au choix spécifique de l'initialisation des dates de début afin d'assurer l'obtention de la solution optimale ou sous optimale ou bien des solutions réalisables (Makespan optimal) en réduisant le maximum le nombre de cycles ou plutôt le nombre d'itérations requises pour trouver une solution valide correspondante à chaque benchmark testé.

De même, le paramètre W_f est utilisé comme un poids positif de la rétroaction mettant à jour les raccordements entre les unités, son rôle est l'ajustement des dates de début S_{ijk}.

Toutes les procédures de ce calcul seront répétitives jusqu'à ce que nous obtenions une liste de dates de début qui satisfait les contraintes prescrites et par conséquent déterminer l'ordre de passage des tâches sur chaque machine.

Afin d'atteindre l'objectif de ce travail, qui n'est autre que l'étude des performances de l'architecture du réseau de neurone (Figure (II.10)) proposée par Willems et Brandts [WIL95] suivant le choix d'initialisation des dates de début S_{ijk} , nous présentons trois types d'initialisation dont l'une d'elle constitue notre nouvelle heuristique d'initialisation.

a. *Première méthode "procédure aléatoire d'initialisation des différentes valeurs des dates de début S_{ijk}"*

La première méthode d'initialisation affecte à chaque activation des dates de début (*S-unité*) une valeur aléatoire. Les différentes valeurs aléatoires affectées doivent être différentes d'une unité à l'autre. Cette initialisation fait partie de la littérature [WIL95].

b. *Deuxième méthode "procédure aléatoire d'initialisation avec la même valeur fixe pour l'ensemble des dates de début S_{ijk}"*
c. *Troisième méthode " nouvelle proposition heuristique"*

La troisième méthode d'initialisation constitue une heuristique utilisée au niveau du calcul au sein de l'architecture générale de l'architecture générale des contraintes de ressources représentée par la Figure (II.10). Cette heuristique est basée sur une procédure de prétraitement du problème à résoudre dont le but est d'accélérer l'algorithme de Willems et Brandts [WIL95] afin de trouver une solution optimale ou proche de l'optimale. En effet, l'heuristique proposée est basée sur la base de données du problème classique d'ordonnancement job shop.

Pour pouvoir expliquer cette heuristique, nous avons besoin tout d'abord de définir le problème d'ordonnancement job shop dans un cas général. Ceci est illustré par le Tableau II.2 par l'intermédiaire d'un modèle général d'un benchmark du problème job shop (machines et temps opératoires correspondant.

	Machine M_k (Temps opératoire : t_{ijk})					
Tâches	Opérations					
	1	2	.	k	.	r
1	5(3)	3(5)	.	$h(t_{1kh})$		$f(t_{1rf})$
2	2(6)	1(4)	.	.	.	.
.	.	.	.	.	.	.
i	$h(t_{i1h})$	$l(t_{i2l})$.	.	$f(t_{ikf})$	.	$s(t_{irs})$
.	.	.	.	.	.	.
n	$f(t_{n1f})$	.	.	$s(t_{nks})$		$h(t_{nrh})$

Tableau II. 2 : Gamme opératoire d'un job shop généralisé

L'initialisation adoptée pour les dates de début des opérations considère que les machines sont continuellement disponibles et que chaque tâche est traitée toute seule et indépendamment des autres. C'est-à-dire, au cours de cette initialisation, nous prenons compte toujours des contraintes de précédence entre les opérations sans tenir compte des contraintes de partage de ressources entre les tâches. Et c'est pour cette raison que nous avions considéré l'initialisation des opérations de chaque tâche sans tenir compte des autres tâches de l'ordonnancement de l'atelier qui peuvent partager l'utilisation de la même machine à un instant donné.

Par exemple, la procédure d'initialisation pour la tâche i ($i^{ème}$ ligne du Tableau (II.2)) se déroule comme suit : nous commençons à la date de début S_{i1h} =0 par attribuer la machine h à la première opération de la tâche i (O_{i1h}) durant son temps de traitement t_{i1h}. En deuxième étape et à la date S_{i2l}= t_{i1h} nous attribuons la machine l à la deuxième opération de la même tâche i (O_{i2l}) durant un temps de traitement t_{i2l} et ainsi de suite pour le reste des opérations de la même tâche i. Et c'est ainsi que nous organisons les différentes opérations du problème général d'ordonnancement job shop modélisé suivant le graphe disjonctif tel qu'il est présenté par le Tableau (II. 2). Le modèle général résultant de l'initialisation que nous avons opté est illustré par la Figure (II.11).

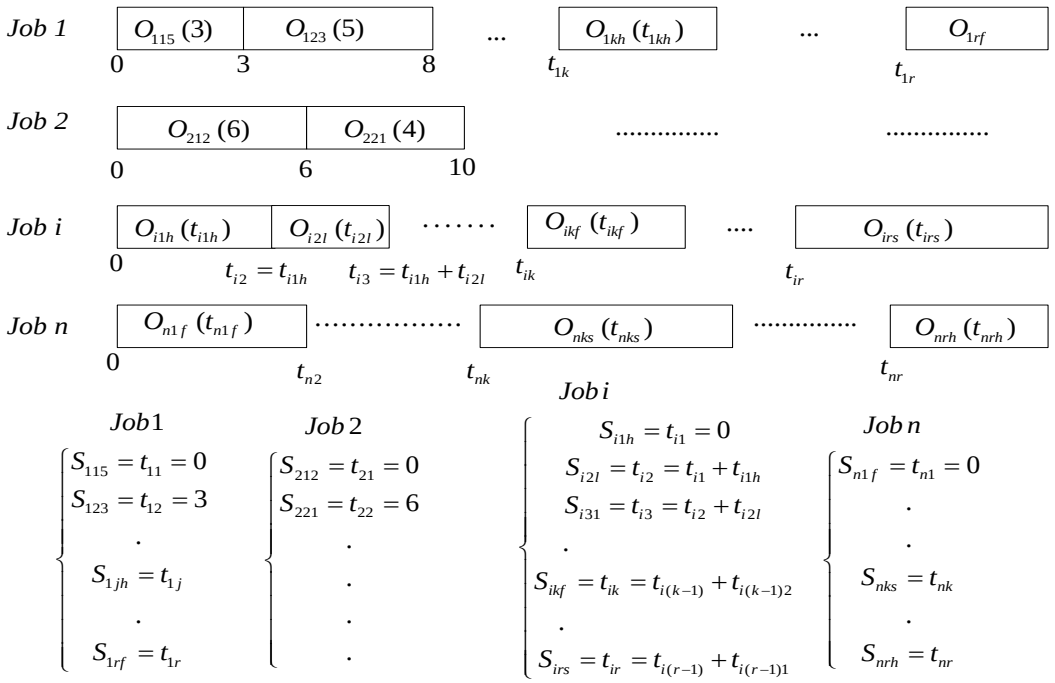


Figure II. 11 : Initialisation des différentes dates de début d'un problème de job shop utilisant la nouvelle heuristique proposée (troisième méthode)

L'implémentation du modèle du réseau de Hopfield proposé par Willems et Brandts [WIL95] ainsi que l'amélioration proposée est articulée autour de cinq procédures d'initialisation:

La procédure de la présentation de données : par la saisie du nombre de jobs, du nombre de machines, des durées opératoires des opérations sur les machines, de la gamme opératoire de chaque job ainsi que les valeurs des dates de début initiales *CI* des S_{ijk} proposées suivant la méthode 3.

CI: Valeur initiale de la date de début.

La procédure de la mise à jour des dates de début (prétraitement): par l'ajustement des dates de début initiales S_{ijk} pour satisfaire les contraintes de séquencement en utilisant le paramètre ΔV.

ΔV étant les changements faits par l'algorithme sur toutes les dates de début comparé à *CI* afin de converger vers une solution faisable pour un problème d'ordonnancement de type job shop.

La procédure de résolution neuronale : Calculer les différents états des neurones *CS, CRI, CRII* et Y_{ipk}

La procédure d'ajustement : Si au moins l'une des deux contraintes (contrainte de séquencement ou contrainte de ressource liée aux variables binaires) persiste violée, le programme effectue une nouvelle résolution, en considérant les composantes de la solution obtenue comme étant les entrées des neurones.

La procédure de réalisabilité : Cette procédure est activée si l'ajustement donne une solution qui vérifie les deux contraintes.

Ainsi s'achève l'organisation générale de cette nouvelle procédure heuristique proposée dans [YAH11a] qui respecte conjointement les contraintes de séquencement et les contraintes de ressources par l'organigramme de la Figure (II.12). Cet organigramme, présente les procédures proposées pour améliorer l'initialisation des dates de débuts S_{ijk} utilisés par le modèle de la programmation linéaire en nombre entier qui est transformé en réseaux de neurones Hopfield illustrés successivement par les Figures (II.7-II.10).

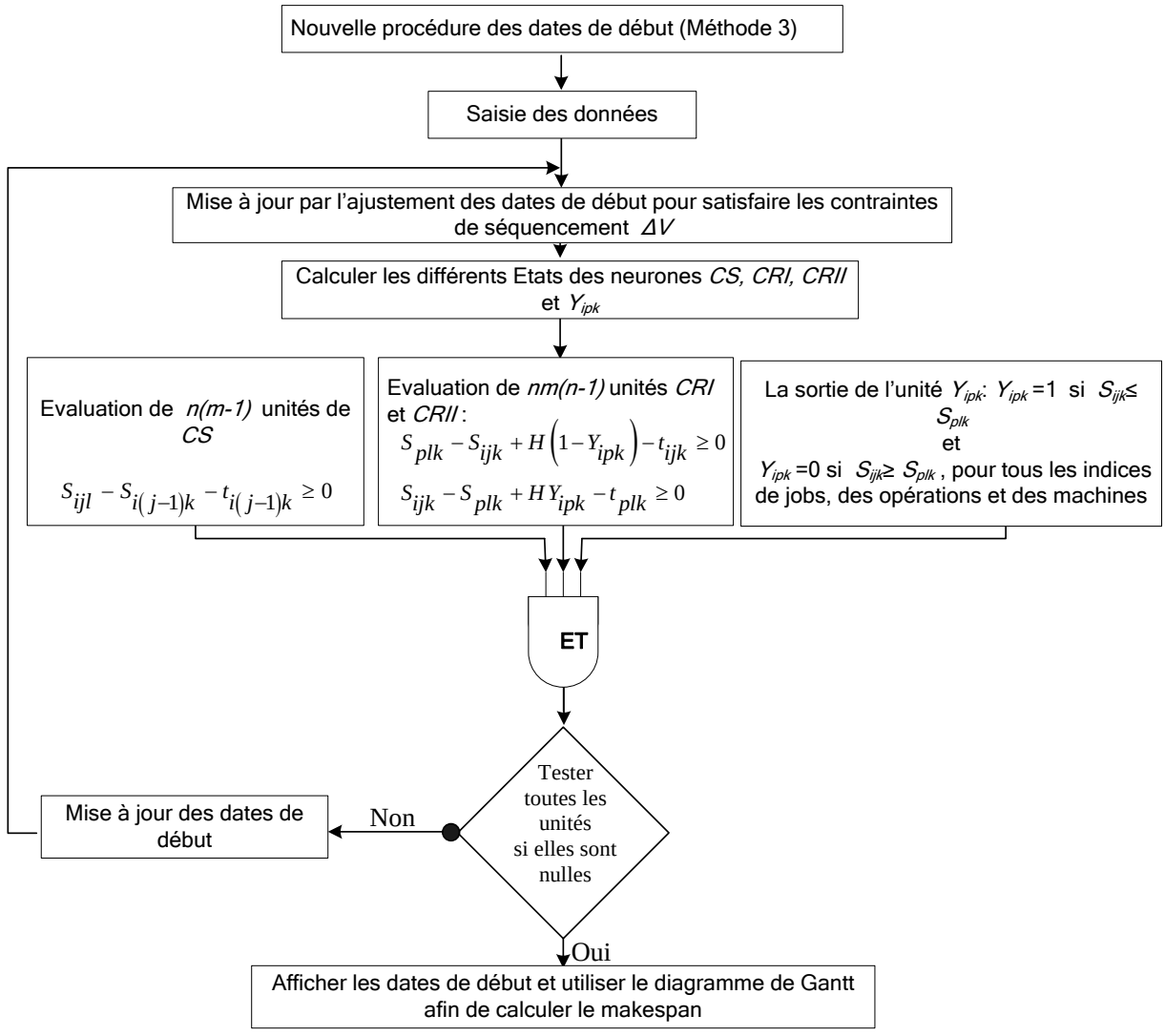


Figure II. 12 : Organigramme de l'amélioration apportée (Méthode 3)

5. Résultats de simulation

Les expériences numériques dans la dynamique des systèmes complexes permettent d'augmenter la compréhension des expériences de laboratoire [ZIM04]. C'est la raison pour laquelle nous utiliserons cet outil pour valider l'heuristique proposée. Nous adopterons une démarche qui sera présentée en quatre étapes pour examiner les résultats de simulation sur le modèle de réseau de neurone de type Hopfield proposé par Willems et Brandts [WIL95] par le calcul des unités (*S, CS, CRI* et *CRII*).

La première étape sera réservée pour présenter et commenter les résultats de simulation utilisant les deux premières méthodes. Dans la deuxième étape nous exposons les résultats de simulation utilisant la nouvelle heuristique d'initialisation. En troisième étape une étude comparative des résultats de simulation pour les trois méthodes d'initialisation sera présentée. Enfin et en dernière étape une étude comparative en profondeur de l'heuristique proposée à l'égard d'autres approches de résolution de la littérature tels que les algorithmes génétiques, les contraintes de satisfaction des réseaux de neurones adaptatifs (CSANN) (GA), la méthode tabou, le recuit simulé, l'algorithme d'acceptation à seuil…

L'outil utilisé pour visualiser les solutions envisagées étant le diagramme de Gantt.

Pour valider l'heuristique proposée et pouvoir mettre en évidence son impact par rapport aux autres méthodes sur le modèle de Willems et Brandts [WIL95], notre choix portera sur des benchmarks dont la valeur optimale du Makespan est connue d'avance pour les méthodes de résolution citées. En effet, nous avons présenté dans le premier chapitre un ensemble de benchmarks de différentes tailles et très connu dans la littérature et en particulier dans les travaux de Willems *et al.* [WIL94], [WIL95].

Il s'agit de problèmes où des instances de petite et de moyenne taille ne dépassant pas 10jobs/10 machines. Ils sont détaillés comme suit par leurs gammes opératoires et durées opératoires des tâches:

- *Exemple 1 : Un* problème de job shop à 2 jobs et 3 machines. Ce problème est noté 2*/3/J/Cmax* et illustré par le Tableau I. 4 du chapitre I.
- *Exemple 2 : Un* problème de job shop à 4 jobs et 3 machines. Ce problème est noté *4/3/J/Cmax* et illustré par le Tableau I. 3 du chapitre I.
- *Exemple 3 : Un* problème de job shop mt06 à 6 jobs et 6 machines. Ce problème est noté *6/6/J/Cmax* et illustré par le Tableau I. 5 du chapitre I.
- *Exemple 4 : Un* problème de job shop mt10 à 10 jobs et 10 machines. Ce problème est noté *10/10/J/Cmax* et illustré par le Tableau I. 6 du chapitre I.

Pour évaluer les résultats de simulation, nous rappelons quelques paramètres utilisés :

W_f : une petite valeur positive du poids de la rétroaction mettant à jour les raccordements (par exemple +0.1)

DD_M: Dates de début modifiés utilisées pour tracer le diagramme de Gantt (après le changement d'origine à t = 0).

C_{max}: Makespan du problème considéré.

NC: Le nombre de cycles nécessaires pour que le réseau de neurone de Hopfield converge.

CPU : le temps de calcul correspondant au temps total nécessaire pour trouver la solution après un nombre prédéterminé de cycles en secondes (de l'anglais *Central Processing Unit).*

A. Résultats de simulation de la méthode 1 et de la méthode 2

1. *Première méthode "procédure aléatoire d'initialisation des différentes valeurs des dates de début S_{ijk}"*

Pour connaitre les performances de cette procédure aléatoire d'initialisation, nous l'avons testé sur deux problèmes de petite taille qui constituent les deux premiers exemples déjà présentés par l'exemple 1 (*2/3/J/Cmax*) et l'exemple 2 (*4/3/J/Cmax*) et ce en partant de dates de début initiales différentes.

- A propos de l'exemple 1, la valeur optimale du Makespan est $C_{max}=22.2$

Le Tableau (II. 3) suivant représente respectivement l'indice de l'expérience, la valeur du Makespan ainsi que la remarque concernant la solution correspondante à chacune 30 générations des dates de début choisies de façon aléatoire. Sur le même tableau, les lignes sélectionnées en gris correspondent aux solutions dont les C_{max} sont acceptables.

CS : contrainte de séquencement.

CR : Contrainte de ressource.

Indice de l'expérience	Valeur du Makespan C_{max}	Remarques
1	32.2	
2	9	Chevauchement entre les opérations (violation de deux contraintes, CS et CR).
3	22.2	Solution optimale.
4	34.4	
5	0	Violation de la contrainte de CS.
6	29.2	

7	34.4	
8	9.1	Chevauchement entre les opérations (violation de deux contraintes, CS et CR).
9	19.5	Chevauchement entre les opérations (violation de deux contraintes, CS et CR).
10	0	CS n'est pas respectée c'est-à-dire l'ordre d'exécution des opérations n'est pas respecté.
11	33.5	
12	14	Chevauchement entre les opérations.
13	28	
14	0	CS n'est pas respectée.
15	30.9	
16	42	
17	14	Chevauchement entre les opérations.
18	29.3	
19	33.3	
20	34.3	
21	21	Chevauchement entre les opérations.
22	9	Chevauchement entre les opérations.
23	0	CS n'est pas respectée.
24	17.5	Chevauchement entre les opérations.
25	0	CS n'est pas respectée.
26	0	CS n'est pas respectée.
27	17.5	Chevauchement entre les opérations.
28	31.2	
29	29.4	
30	39.4	

Tableau II. 3 : Solutions correspondantes à l'initialisation aléatoire de l'exemple 1

La Figure (II. 8) montre la courbe obtenue qui représente les valeurs du Makespan (solution) *Cmax* pour une liste aléatoire des dates de début *CI* tracée pour 30 expériences pour l'exemple 1.

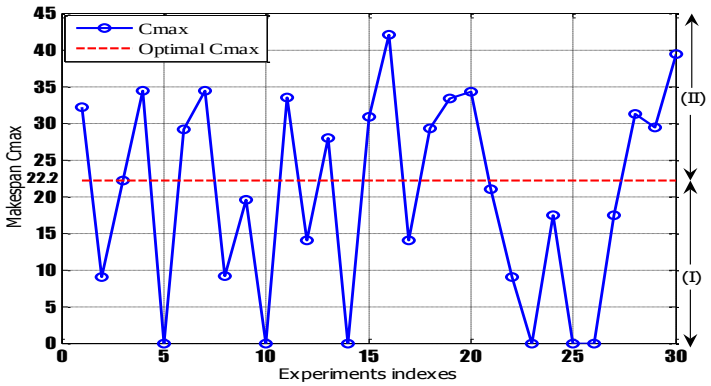


Figure II. 13 : Résultats de la simulation utilisant la méthode d'initialisation aléatoire pour l'exemple 1: *2/3/J/Cmax*

Nous notons à travers la courbe ci-dessus de la Figure (II. 13), que 50% des solutions des expériences ne convergent pas vers une solution réalisable. Ces solutions ayant un Makespan C_{max}<22.2 ne respectent pas les équations (II.15), (II.18), et (II.19) c'est-à-dire les contraintes de séquencement *CS* et les contraintes de ressources *CRI* et *CRII*. Cette divergence des solutions est due à deux raisons, soit :

- Par ce qu'il ya un chevauchement (recouvrement) entre les opérations exécutées sur la même machine (Figure II.14) avec un $C_{max} \neq 0$.
- La violation de la contrainte de séquencement c'est-à-dire l'ordre d'exécution des opérations n'est pas respecté, $C_{max} = 0$.

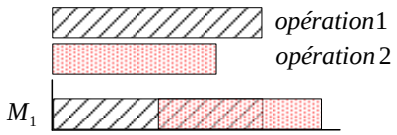


Figure II. 14 : Chevauchement entre les opérations

D'autres parts, pour les expériences qui ont mené à des solutions acceptables $C_{max} > 22.2$, ils respectent à la fois les deux contraintes du problème de job shop traité, mais il ya dans des cas de temps libre (temps mort) entre les opérations utilisant la même machine. C'est-à-dire même si la machine est libre, elle n'exécute pas immédiatement l'opération en attente et ceci est illustré par un exemple par la Figure (II. 15).

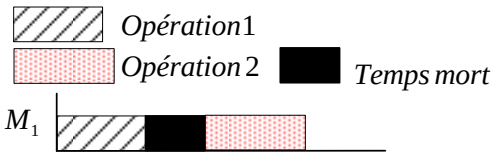


Figure II. 15 : Temps mort entre les opérations

- ❖ A propos de l'exemple 2, la valeur optimale du Makespan est $C_{max} = 14.3$

Le Tableau (II.4) représente les résultats de la même méthode d'initialisation pour l'exemple 2. Ce tableau donne les mêmes informations que l'exemple 1.

Indice de l'expérience	Valeur du Makespan C_{max}	Remarques
1	0	CS n'est pas respectée.

2	0	CS n'est pas respectée.
3	0	CS n'est pas respectée.
4	12,5	Chevauchement entre les opérations.
5	0	CS n'est pas respectée.
6	0	CS n'est pas respectée.
7	27.6	
8	0	CS n'est pas respectée.
9	26.8	
10	0	CS n'est pas respectée.
11	0	CS n'est pas respectée.
12	33.4	
13	0	CS n'est pas respectée.
14	0	CS n'est pas respectée.
15	0	CS n'est pas respectée.
16	29.5	
17	0	CS n'est pas respectée.
18	11	Chevauchement entre les opérations.
19	0	CS n'est pas respectée.
20	14.6	
21	0	CS n'est pas respectée.
22	0	CS n'est pas respectée.
23	0	CS n'est pas respectée.
24	30.6	
25	0	CS n'est pas respectée.
26	0	CS n'est pas respectée.
27	13	Chevauchement entre les opérations.
28	0	CS n'est pas respectée.
29	0	CS n'est pas respectée.
30	0	CS n'est pas respectée.

Tableau II. 4 : Solutions correspondant à l'initialisation aléatoire de l'exemple 2

La Figure (II.16) montre la courbe obtenue représentant les valeurs du Makespan (solution) C_{max} pour une liste aléatoire des dates de début CI tracée pour 30 expériences correspondant à l'exemple 2.

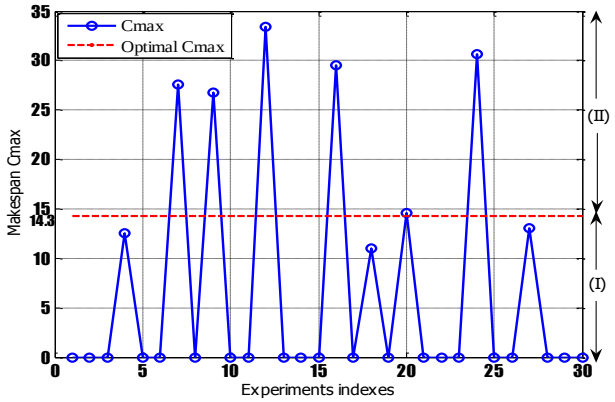


Figure II. 16 : Résultats de la simulation utilisant la méthode d'initialisation aléatoire pour l'exemple 2: *4/3/J/Cmax*

La courbe de la Figure (II.16), montre que 70% des solutions des 30 expériences ne convergent pas vers une solution acceptable parce que la valeur du Makespan $C_{max} < 14.3$ et ce ci ne respecte pas les équations (II.15), (II.18), et (II.19) c'est-à-dire les contraintes de séquencement *CS* et les contraintes de ressources *CRI* et *CRII* sont violées. Cette divergence des solutions est due à deux raisons, soit :

- Que l'ordre des opérations n'est pas respecté (environ 60% des solutions non convergentes).
- Il y a chevauchement entre les opérations (40% des solutions non convergentes).

D'autres parts, pour les expériences qui ont mené à des solutions acceptables $C_{max} > 14.3$ et qui représentent 30%, ils respectent à la fois les deux contraintes du problème de job shop traité, mais il ya dans des cas de temps libre (temps mort) entre les opérations utilisant la même machine.

Enfin, nous pouvons conclure que la probabilité d'obtenir une solution optimale $(C_{max} = 14.3)$ avec cette méthode d'initialisation aléatoire est faible.

2. *Deuxième méthode "procédure aléatoire d'initialisation en conservant la même valeur fixe pour l'ensemble des dates de début S_{ijk}"*

Les deux Tableaux II. 5-II. 6 montrent respectivement les résultats de simulation testant les deux exemples benchmarks *2/3/J/Cmax* et *4/3/J/Cmax* utilisant la deuxième méthode d'initialisation.

S_{ijk}	IC	ΔV	DD_M
S_{111}	30	-18	12
S_{122}	30	-11	19
S_{133}	30	-03	27
S_{213}	30	-30	0
S_{221}	30	-27	7
S_{232}	30	-20	10
$C_{\max} = 29$		$NC = 147$	

Tableau II. 5 : Résultats de simulations pour la deuxième méthode d'initialisation en testant l'exemple 1, *2/3/J/*C_{max}

S_{ijk}	S_{111}	S_{122}	S_{133}	S_{212}	S_{221}	S_{233}	S_{313}	S_{322}	S_{331}	S_{412}	S_{423}	S_{431}
IC	27	27	27	27	27	27	27	27	27	27	27	27
ΔV	-27	-23	-17	-20	-19	-15	-11	-8	-5.9	-6	-3	0.1
DD_M	0	4	10	7	8	12	16	19	21.1	21	24	27.1
	$C_{\max} = 28.1$						$NC = 325$					

Tableau II. 6: Résultats de simulations pour la deuxième méthode d'initialisation en testant l'exemple 2, *4/3/J/*C_{max}

Commentaires et discussions concernant la deuxième méthode d'initialisation

Les diverses simulations, qui sont de l'ordre d'une centaine de générations de valeurs aléatoire égales pour *IC* ont abouties à des solutions convergentes toujours vers la même valeur d'une solution C_{max} non-optimale.

B. Résultats de simulation de la nouvelle méthode heuristique d'initialisation (Méthode 3)

Après l'analyse des résultats obtenus par les méthodes classiques d'initialisation (méthode 1 et méthode 2), d'autres investigations ont été effectuées afin d'améliorer le Makespan et atteindre une solution optimale dans un délai raisonnable et ceci en introduisant la troisième méthode d'initialisation qui constitue l'amélioration proposée sur le réseau de Willems et Brandts [WIL95]. Cette heuristique, sera testée dans la suite afin de mettre en évidence son efficacité.

Les résultats obtenus en appliquant cette heuristique, sur l'exemple 1 et l'exemple 2, sont présentés par le Tableau II. 7 et le Tableau II. 8. Les Figures II. 12-II. 13 représentent respectivement les diagrammes de Gantt représentant les solutions correspondantes.

S_{ijk}	IC	ΔV	DD_M
S_{111}	0	0	0
S_{122}	5	0.1	5.1
S_{133}	13	1.1	14.1
S_{213}	0	0	0
S_{221}	7	1.1	8.1
S_{232}	10	3.2	13.2
$C_{\max} = 22.2$		$NC = 24$	

Tableau II. 7 : Résultats de simulations pour la troisième méthode d'initialisation testant l'exemple 1, *2/3/J/*C_{max}

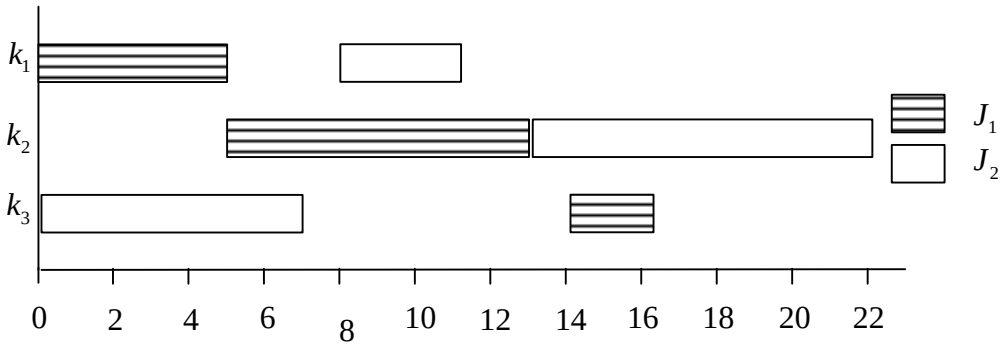


Figure II. 17 : Diagramme de Gantt correspondant à la solution du test de l'exemple 1 par la troisième méthode d'initialisation

S_{ijk}	S_{111}	S_{122}	S_{133}	S_{212}	S_{221}	S_{233}	S_{313}	S_{322}	S_{331}	S_{412}	S_{423}	S_{431}
IC	0	4	7	0	1	5	0	3	5	0	3	6
ΔV	0	3.8	5.3	0	3	3.1	0	2.7	3.1	2	2.1	5.2
DD_M	0	7.8	12.3	0	4	8.1	0	5.7	8.1	2	5.1	11.2
	$C_{\max} = 14.3$						$NC = 32$					

Tableau II. 8 : Résultats de simulations pour la troisième méthode d'initialisation testant l'exemple 2, *4/3/J/C_{max}*

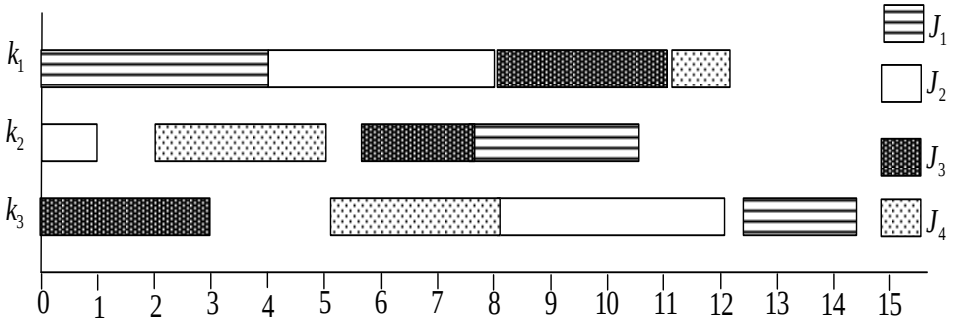


Figure II. 18 : Diagramme de Gantt correspondent à la solution du test de l'exemple 2 par la troisième méthode d'initialisation

Afin de mettre en évidence l'efficacité de cette heuristique sur le réseau de Hopfield proposé par Willems et Brandts [WIL95], nous proposons de la comparer vis-à-vis des résultats de simulation donnés par les deux premières méthodes d'initialisation.

Le Tableau II. 9, présente un résumé sur les résultats obtenus pour les trois méthodes d'initialisation testant les cinq différents benchmarks. A travers cette recapitalisation (Tableau II. 9 et les autres résultats de simulation présentés), il est clair que la troisième méthode "nouvelle heuristique proposée" est plus avantageuse que les deux premières.

Le premier avantage et le plus important, c'est que cette méthode garantit la convergence immédiate (une seule solution en une seule fois) du réseau de Hopfield proposé par Willems et Brandts [WIL95] vers une solution optimale ou proche de l'optimale pour le critère à minimiser qui est le Makespan du problème d'ordonnancement des ateliers de type Job Shop. Et ceci peut être remarqué sur les trois critères qui ont été mises en évidence sur le Tableau II. 9 pour tous les benchmarks testés.

	Première méthode d'initialisation			Deuxième méthode d'initialisation			Nouvelle méthode heuristique proposée		
	Cmax	*NC*	*CPU*	*Cmax*	*NC*	*CPU*	*Cmax*	*NC*	*CPU*
2/3	29.3	46	2.4	29	147	1.73	22.2	24	0.32
4/3	20.4	83	4.5	28.1	325	10.75	14.3	32	1.07
4/5	73.6	254	14.2	44.1	318	15.04	68.2	192	9.54
ft06	85.2	472	156.4	129.1	2725	578.45	67.4	393	81.89
10/10	142	1596	34562	256	4562	45869	107	1358	27994

Tableau II. 9 : Comparaison entre les trois méthodes d'initialisation

Une meilleure amélioration qui s'interprète par une réduction du temps de calcul et même du nombre d'itérations requises pour trouver une solution.

Ce que nous pouvons constater d'après les diverses simulations, (les trois méthodes d'initialisation pour tous les benchmarks testés) une forte augmentation du temps d'exécution (en *CPU*) en fonction si on procède à une variation du nombre de jobs (2/3 et 4/3) tel qu'il est représenté par la Figure II. 19. De même aussi, une forte augmentation du temps d'exécution est illustrée par la Figure (II. 20) si on procède à une variation du nombre de machines (4/3 et 4/5). Ceci peut être justifié par l'explosion du nombre de variables : (*n.m)* variables continues et $n(m-1)$ contraintes de séquencement ainsi que les $nm(n-1)$ contraintes de disjonctions (contraintes de ressources).

Il est à noter que le temps (*CPU*) est exprimé en seconde, il correspond au temps total requis pour trouver une solution après un nombre prédéfini de cycles.

De même, ce temps de calcul (*CPU*) comprend également le temps de prétraitement nécessaire pour la procédure de mise à jour des dates de débuts S_{ijk} de l'organigramme de la Figure II. 12.

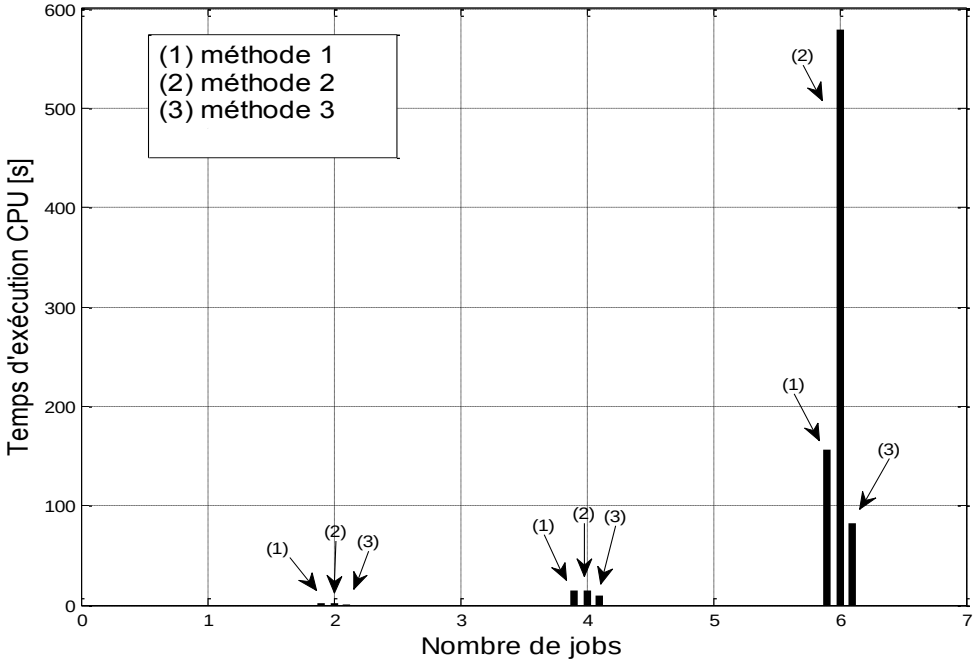


Figure II. 19 : Variation du temps d'exécution en fonction du nombre de jobs

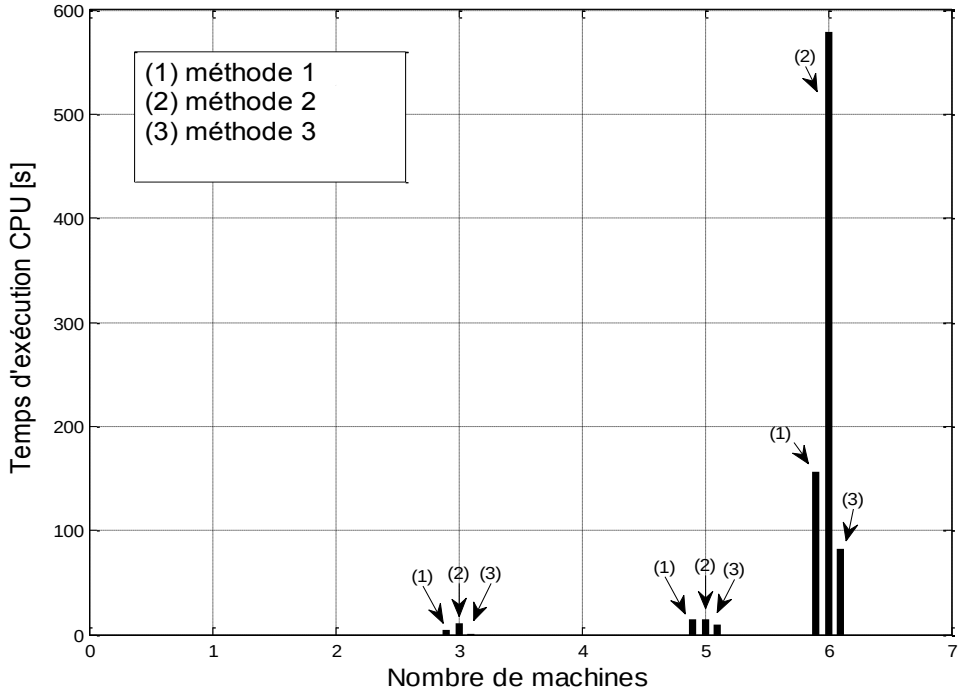


Figure II. 20 : Variation du temps d'exécution en fonction du nombre de machines

En inspectant les résultats présentés par les Figures (II.19)-(II.20), on constate une nette réduction du temps de calcul (*CPU*) des résultats de la méthode 3 par rapport aux deux autres

méthodes d'initialisation et ceci que ce soit en augmentant le nombre de jobs ou le nombre de machines.

Face à ces améliorations prouvées par notre heuristique d'initialisation sur le réseau de neurone de Hopfield proposé par Willems et Brandts [WIL95] pour résoudre le problème d'ordonnancement job shop, nous pouvons noter que la qualité de la solution dépend sensiblement de l'état initial des neurones (dates de débuts initiales). Ainsi, une question intéressante peut se poser à ce niveau : comment la nouvelle heuristique d'initialisation peut-elle réduire le temps de calcul CPU tout en garantissant la convergence immédiate du réseau vers une solution optimale ou proche de l'optimale ? Ceci peut être expliqué comme suit, le problème d'ordonnancement job shop considéré à la section 3.*B*, est modélisé par trois équations : l'équation (II.15) qui modélise les contraintes de séquencement, les équations (II. 18)-(II.19) qui modélisent les contraintes de ressources d'équations. Pour la troisième méthode d'initialisation, la contrainte de séquencement est respectée dès le départ et ce à partir de l'heuristique de l'initialisation adoptée au réseau de neurone Hopfield. Concrètement cette heuristique nous a permis de minimiser le nombre d'équation à chaque itération (uniquement les deux équations (II.18) - (II.19). Ce ci nous mène automatiquement à un gain de temps de calcul pour trouver une solution optimale ou proche de l'optimale. Alors que pour les deux premières méthodes d'initialisation, le réseau de Hopfield doit tenir compte des deux contraintes (trois équations) à chaque itération.

C. Etude comparative avec d'autres algorithmes existants

Au cours de ce travail, nous avons développé une heuristique d'initialisation du réseau de Hopfield proposé par Willems et Brandts [WIL95] afin de résoudre le problème d'ordonnancement job shop statique.

A travers les résultats de simulation utilisant cette heuristique, nous avons pu noter plusieurs avantages en termes de :

- Convergence du réseau
- Réduction du nombre de cycles
- Réduction du temps de calcul CPU

L'étude comparative va être faite en deux parties :

La première partie va présenter une étude comparative des résultats donnés en simulant le réseau de neurone de Hopfield proposé par Willems et Brandts [WIL95] sans initialisation et les résultats utilisant l'heuristique d'initialisation développé.

Le Tableau II. 10 présente la première partie de l'étude comparative en termes de nombre de cycle *NC* et de la valeur du critère à minimiser qui est la Makespan C_{max}.

	Willems et Brandts [WIL95]		Nouvelle méthode heuristique proposée	
	C_{max}	*NC*	C_{max}	*NC*
2/3	22	42	22.2	24
4/3	14	142	14.3	32

Tableau II. 10 : Comparaison entre Willems et Brandts [WIL95] et la nouvelle heuristique développée

A travers le Tableau II. 10, nous pouvons remarquer une nette amélioration sur le nombre de cycle nécessaire pour aboutir à une solution très proche pour les deux benchmarks testés par Willems et Brandts [WIL95].

La deuxième partie portera sur une étude comparative de l'heuristique proposée par rapport à d'autres algorithmes les contraintes de satisfaction des réseaux de neurones adaptatifs (CSANN) (GA), les algorithmes génétiques, la méthode tabou, le recuit simulé, l'algorithme d'acceptation à seuil et la règle de priorité SPT. Cette étude comparative est illustrée par le Tableau II. 11 dont la majorité de ses valeurs sont tirées de la littérature de Käschel et al [KÄS99].

Examples	Proposed heuristic	CSANN	Genetic algorithm	Simulated annealing	Threshold accepting	Flood-method	SPT
ft06	67.4	55	55	55	55	55	83
ft10	-	-	951	938	954	995	1399
10x10	107	95	-	-	-	-	-

Tableau II. 11 : Comparaison avec d'autres algorithmes existants

De nombreux autres paramètres ne sont pas fournis par les utilisateurs de ces algorithmes tels que le temps de calcul *CPU* et le nombre de cycles *NC* qui ont participé à ces résultats. Toutefois, l'heuristique proposée donne des résultats optimaux ou proche de l'optimum lorsqu'elle est testée sur des benchmarks d'atelier job shop de petite taille et ce en un seul essai. D'où cette heuristique parait plus simple en terme d'implémentation et de temps de calcul pour les problèmes d'ordonnancement de type job shop.

Cependant, et pour les autres méthodes, même si elles ont donné de bons résultats, comme le cas des algorithmes génétiques par exemple, il nécessite un temps de calcul assez élevé pour

générer une première solution et il fournit une solution optimale ou proche de l'optimale après plusieurs essais et plusieurs générations de populations initiales. Egalement, le recuit simulé est une méthode puissante mais requiert beaucoup de temps afin de générer de nouvelles configurations pour converger.

Les approches actuelles basées sur le réseau de neurones de Hopfield ont essentiellement deux faiblesses qui limitent particulièrement leurs utilités dans des applications réelles. D'abord ils sont sensibles aux minimums locaux et aussi, ils requièrent beaucoup de temps pour converger.

6. Conclusion

Comme ce chapitre présente une amélioration du réseau de neurone de Hopfield initialement proposé par Willems et Brandts [WIL95] pour la résolution du problème d'ordonnancement job shop qui a été modélisé par la programmation linéaire en nombre entier.

Au cours de ce travail, nous avons présenté notre contribution principale qui est une nouvelle procédure d'initialisation en vue de l'optimisation d'un réseau de neurone Hopfield appliqué aux problèmes d'ordonnancement d'ateliers à cheminement multiples : job shop. Cette démarche consiste en une heuristique de choix optimal d'initialisation des dates de début des opérations afin d'accélérer le réseau de neurones Hopfield (HNN).

L'avantage essentiel de cette nouvelle heuristique est l'amélioration de la convergence du réseau de Hopfield vers une solution optimale ou proche de l'optimale pour le critère à minimiser qui est le Makespan du problème d'ordonnancement des ateliers de type Job Shop.

Pour mettre en évidence l'efficacité de notre heuristique d'initialisation, nous l'avons comparé à deux autres méthodes d'initialisation en les testant sur les même benchmarks.

Une étude comparative avec d'autres algorithmes existants, tels que les algorithmes génétiques, les réseaux de neurones adaptatifs de satisfaction de contrainte et les approches combinées par heuristique pour le problème d'ordonnancement des ateliers de type Job Shop généralisé, le recuit simulé, la règle de priorité telle que (SPT) la plus courte durée opératoire, etc., a été élaborée.

A travers cette étude en simulation, il a été démontré que l'approche de Willems *et al.* [WIL94] et [WIL95] combinée à notre méthode heuristique est efficace en terme de résolution, de vitesse de convergence, de la qualité de la solution escomptée ainsi que de la réduction du temps de calcul.

Tout le long de ce chapitre, nous avons considéré que dans le problème job shop, les machines sont supposées toujours disponibles. Dans le chapitre qui va suivre, nous allons

tenir compte des contraintes d'indisponibilité des machines dans le problème job shop. Par conséquent, dans le chapitre III, les contraintes d'indisponibilité des machines dues aux périodes de maintenance préventive systématique seront prises en compte pour résoudre le problème d'ordonnancement conjoint de la production et de la maintenance.

Chapitre III : Nouvelle méthode de décalage pour la résolution du problème d'ordonnancement conjoint de la production et de la maintenance dans un atelier job shop

Résumé *: Dans la littérature, la plupart des modèles d'ordonnancement des machines étudiés, supposent que les ressources (machines) sont continuellement disponibles. Cependant, dans beaucoup de situations réalistes, les machines doivent être maintenues et par conséquent peuvent être indisponibles pendant une certaine période.*

Le présent chapitre traite une nouvelle heuristique basée sur une méthode de décalage employée pour résoudre le problème d'ordonnancement conjoint de la production et de la maintenance préventive systématique pour un problème d'ordonnancement de type job shop. L'heuristique proposée a pour objectifs la réduction du Makespan C_{max} *ainsi que le coût total de la maintenance par cycle de production CM qui est due soit à l'avance ou soit au retard des tâches de la maintenance préventive des machines. Généralement si les tâches de la maintenance sont exécutées en retard (tâches retardées), ceci peut avoir un coût de maintenance plus élevé et cela est due au fait qu'elles augmentent le risque de pannes et produisent un arrêt de production. La méthode proposée est basée sur un ensemble de règles tenant compte des contraintes de production ainsi que les contraintes de partage des machines et les contraintes de l'exécution des tâches de maintenance du problème d'ordonnancement job shop. Pour souligner l'efficacité de la méthode proposée, nous avons appliqué la nouvelle heuristique sur quelques benchmarks d'illustration. Les exécutions du nouvel algorithme concernent le Makespan* C_{max}*, le temps libre de chaque machine et quelques autres statistiques liées au coût de la production et de la maintenance CM.*

1. Introduction

L'ordonnancement de la production et la planification de la maintenance préventive (MP) sont deux domaines qui ont reçu une attention considérable à la fois dans l'industrie manufacturière, et dans la littérature de la recherche opérationnelle [CAS05]. Jusqu'à présent rares sont les travaux qui l'étudient conjointement et que nous appelons souvent les problèmes d'ordonnancement conjoint de la production et de la maintenance. Ils ont été étudiés séparément. En effet, dans les problèmes d'ordonnancement classiques [FOO88], [ROY64], [YAM95], [WIL93], [WIL94], [GAR00], [WIL95], la majorité des travaux supposent que les ressources (ou machines) sont toujours disponibles durant l'horizon d'ordonnancement planifié. Toutefois, une machine peut être indisponible pour plusieurs raisons. Une première raison est de nature stochastique telle que les pannes imprévues et dans ce contexte nous pouvons citer les travaux [CHE06], [CHI10], [AYT05]. Une deuxième raison est de nature déterministe qui constitue par exemple en les périodes d'indisponibilité prévues et planifiées d'avance pour les tâches de maintenance préventive.

C'est dans ce contexte et afin de rester compétitif et performant dans le domaine de la production industrielle et manufacturière, de nombreux paramètres doivent être pris en compte pour s'adapter aux fluctuations du marché caractérisées essentiellement par les demandes irrégulières (aléatoires) de produits en plus des perturbations au sein des ateliers dues aux pannes accidentelles des machines.

Une présentation succincte de la maintenance des systèmes de production a été traitée au premier chapitre. Nous avons rappelé quelques généralités et définitions normatives sur la maintenance suivie par une présentation des différentes politiques de la maintenance. Dans ce chapitre, nous avons essayé de proposer une nouvelle heuristique pour résoudre le problème d'ordonnancement conjoint de la production et de la maintenance pour l'atelier de type job shop. La politique de la maintenance utilisée au cours de ce travail est la maintenance préventive systématique. Pour cela, et dans la deuxième partie de ce chapitre, nous allons commencer par présenter un aperçu succinct sur le problème d'ordonnancement de la production et de la maintenance qui sera suivi en première étape par une modélisation du problème posé et en deuxième étape par le rappel des différentes stratégies d'ordonnancement conjoint. La troisième partie de ce chapitre sera réservée pour rappeler les méthodes de résolution du problème de l'ordonnancement conjoint de la production et de la maintenance ainsi qu'un état de l'art des travaux abordés dans la littérature au niveau de l'atelier de type job shop. Dans la quatrième partie de ce chapitre, notre méthode proposée pour résoudre le

problème d'ordonnancement conjoint de la production et de la maintenance pour l'atelier de type job shop utilisant l'intégration des périodes de maintenance préventive systématique sera détaillée et présentée. La méthode proposée est à base d'une méthode heuristique qui est la simulation à évènement discret exposée au premier chapitre. Afin de détailler d'avantage cet algorithme, nous avons présenté une application directe portant sur les étapes de la simulation sur un benchmark de type job shop (4 tâches et 3 machines) donné par le Tableau I. 3 du premier chapitre. Enfin des commentaires relatifs à cette méthode seront établis à travers les résultats de simulations sur quelques instances générées de benchmark.

2. Ordonnancement de la production en présence de la maintenance

L'ordonnancement de la production en présence des contraintes de la maintenance se ramène à l'ordonnancement conjoint de la production et de la maintenance. Ce problème est une tâche complexe, car il consiste d'une part à ordonnancer la production sous les contraintes de respect des délais, coûts et qualité des produits et d'autre part, à planifier la maintenance sans violer les contraintes de sûreté de fonctionnement des équipements qui assurent la fiabilité des systèmes de production.

L'ordonnancement conjoint de la production et de la maintenance vise l'amélioration des performances de l'atelier par ce qu'il influe directement sur le tryptique coût/qualité/délais d'où sa nécessité.

A. Position du problème

La minimisation du coût de la production est l'un des paramètres à prendre en considération et qui dépend nécessairement de l'optimisation du Makespan C_{max} et au respect d'un compromis entre les contraintes de la production et les contraintes de la maintenance. Ce but est atteint à travers une coordination de tâches entre l'équipe de la production et l'équipe de la maintenance au sein de l'atelier.

Face aux contraintes de la production, la question posée est comment respecter les dates d'intervention de la maintenance préventive (MP) ou bien comment les retarder pour assurer la continuité de la production ? Le fait de reporter les interventions de (MP) ou même les réparations, ces deux actions provoquent à la fois la réduction du temps conçu pour la production et augmentent la probabilité de défaillance de la machine ou de l'équipement en général et par conséquent un arrêt de la production. De même il arrive même à se trouver dans des situations de conflits entre les deux plannings (production et maintenance) ce qui n'amène pas nécessairement à un fonctionnement optimal de l'atelier.

En effet, en considérant les contraintes et les conflits multiples imposés par l'ordonnancement des tâches de la production (Figure III.1) et de la planification des tâches de la maintenance préventive (Figure III. 2), il faut trouver un compromis entre les deux pour garantir une production sans arrêt. Notre travail rentre dans ce cadre. Il se traduit par l'amélioration de la productivité du système de production (réduire le Makespan) et ceci en intégrant des règles de décision pour pouvoir ordonnancer conjointement les tâches de la production et les tâches de la maintenance. C'est-à-dire trouver la décision adéquate lors des conflits de l'ordonnancement séparé comme indiquées par la Figure III. 3.

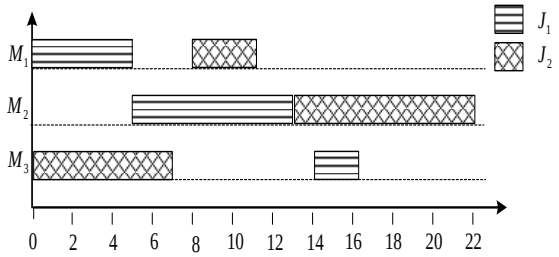


Figure III. 1 : Ordonnancement de la production

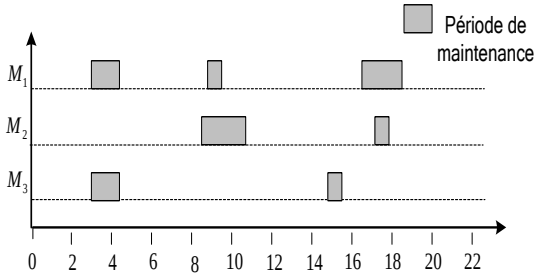


Figure III. 2 : Ordonnancement de la maintenance

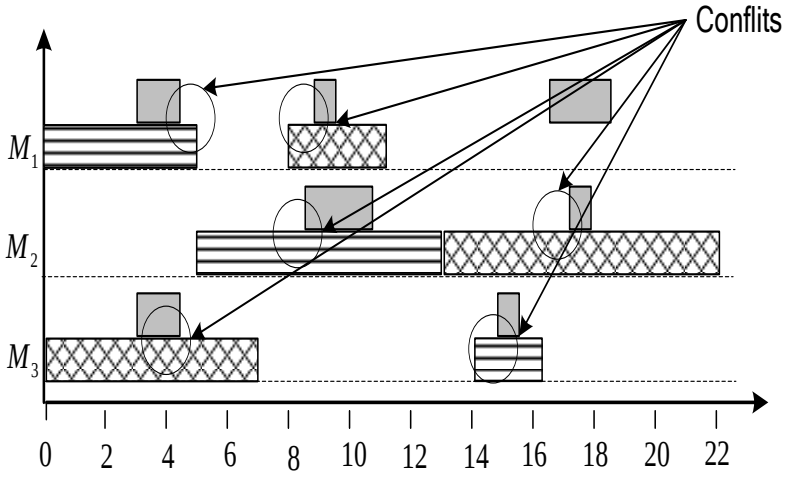


Figure III. 3 : Conflits entre tâches de maintenance et tâches de production dans un ordonnancement séparé

B. Les stratégies d'ordonnancement conjoint

Un état de l'art sur la relation entre la production et la maintenance a été présenté par Budai *et al* [BUD06], ils ont expliqué cette relation par : «*Certaines opérations de maintenance peuvent être réalisées pendant la production et certaines peuvent être faite pendant les arrêts de production régulière en soirée, week-ends et les jours fériés. Cependant, dans de*

nombreux cas, des unités de production doivent être impérativement fermées pour maintenance. Cela peut conduire à des tensions entre le département de la production et la maintenance d'une entreprise».

Egalement, la littérature dédiée au problème d'ordonnancement conjoint a classé les stratégies de résolution des conflits d'exécution des tâches de la production et de la maintenance en trois :

La stratégie séparée, la stratégie séquentielle et la stratégie intégrée. La distinction entre ces trois stratégies intervient uniquement par la méthode de résolution du problème et non pas par la qualité des résultats obtenus.

3. *Stratégie séparée*

Cette stratégie consiste à ordonnancer séparément les tâches de maintenances et les tâches de la production. C'est-à-dire dans ce cas de stratégie, il y aura nécessairement des retards soit au niveau de la production ou au niveau de la maintenance. Ceci constitue un effet néfaste conjointement sur le coût de la production ou bien sur la fiabilité du matériel qui peut engendrer des pannes et par conséquent arrêter la production. Afin de remédier aux conflits entre les ressources humaines et matérielles, il fallait assurer une collaboration adéquate entre les services de maintenance et les services de la production. C'est pour cette raison qu'il faut penser à d'autres stratégies qui peuvent résoudre ces conflits efficacement.

4. *Stratégie séquentielle*

La stratégie séquentielle du problème d'ordonnancement conjoint consiste à ordonnancer l'une des deux tâches, production ou maintenance. Elle optimise un seul aspect : production maintenance. Par exemple s'il s'agit d'ordonnancer les tâches de production, les tâches de maintenance seront considérées comme des périodes d'indisponibilité des ressources et par conséquent il interdit de violer ces contraintes. C'est-à-dire utiliser les indisponibilités des machines comme contraintes supplémentaires à la production et vice-versa.

5. *Stratégie intégrée*

Cette stratégie consiste à respecter à tout moment les conflits existants entre l'ordonnancement de la production et l'ordonnancement de la maintenance. Ceci permet de limiter les risques d'interférence entre les tâches de la production et les tâches de la maintenance, d'où l'amélioration de l'ordonnancement.

3. Les méthodes de résolution du problème de l'ordonnancement conjoint de la production et de la maintenance et état de l'art pour l'atelier de type job shop

Le problème d'ordonnancement conjoint de la production et de la maintenance est un problème crucial et récent. Plusieurs méthodes ont été utilisées pour le résoudre.

Récemment ce problème a été traité pour quelques types d'ateliers très limités. A titre d'exemple nous citons pour le cas de l'atelier d'une machine unique [LOW10], [CAS05], pour l'atelier à machines parallèles [BER10]. Pour l'atelier de type flow shop nous pouvons citer [BEN06], [BEN10]. Néanmoins les travaux qui traitent l'ordonnancement conjoint de la production et de la maintenance pour le cas de l'atelier job shop sont de complexité importante et de plus en plus sont rarement traités [HAR03a], [HAR03b] c'est la raison pour laquelle nous sommes motivés à travailler sur ce problème.

Le Tableau III. 1, présente un petit aperçu de quelques travaux du problème d'ordonnancement conjoint de la production et de la maintenance en donnant conjointement le type d'atelier, la stratégie d'ordonnancement ainsi que la méthode adoptée.

Références (auteurs)	**Type d'atelier**	**Stratégie d'ordonnancement**	**Méthode de résolution**
[CAS05]	Machine unique	intégrée	Programmation
[SON09]	Machine unique	Séparée	Politique à Seuil
[HAR03a], [HAR03b]	Job shop	Intégrée	Algorithme génétique
[KAR08]	Job shop	séparée	Méthodes heuristiques
[SUN10]	Machine parallèle	Intégrée	Méthodes heuristiques
[BEN06],	Flow shop	intégrée	Colonie de fourmis
[BEN010]	Flow shop	intégrée	Algorithmes génétiques
[BER10]	Flow shop	intégrée	Colonie de fourmis

Tableau III. 1: Etat de l'art sur les méthodes de résolution du problème d'ordonnancement conjoint

4. Ordonnancement conjoint de la production et de la maintenance dans le cas d'une maintenance préventive systématique

A. Présentation du problème et symboles utilisés

Il s'agit d'étudier un problème combinant l'ordonnancement conjoint de la production et de la maintenance. Le problème consiste à ordonnancer dynamiquement n tâches indépendantes sur *m* machines différentes assujetties à des tâches de maintenance préventives systématiques dont les dates sont fixes où permettent le décalage dans un atelier de type job shop de taille $n \times m$. L'objectif à atteindre est la minimisation de deux critères conjointement, un critère lié à la production qui est le Makespan C_{max} et un deuxième critère lié à la maintenance qui est la minimisation du coût total des tâches de maintenance préventive systématique. Ces critères seront détaillés ci-dessous.

1. *Critère de la production*

Dans cette sous section, nous considérons l'objectif classique de minimisation du Makespan C_{max} qui est la date d'achèvement de la dernière tâche de l'ordonnancement aussi le temps requis pour accomplir toutes les tâches.

Les paramètres suivants sont adoptés pour formuler tous les critères de production qui peuvent être présentés comme suit :

$J = \{J_1, J_2 ... J_n\}$	: L'ensemble de *n* tâches qui sont toutes disponibles à $t = 0$.
$M = \{M_1, M_2 ... M_m\}$	: L'ensemble de *m* machines (ou ressources) présentant des périodes d'indisponibilités durant l'horizon d'ordonnancement.
i, p, q, d	: Les indices des tâches, $1 \le i \le n$, il en est de même pour *p, q* et *d*.
k, h ,s, f	: Les indices des machines, $1 \le k \le m$, il en est de même pour *h, f* et *s*.
j, r, y, z	: Les indices des opérations, $1 \le j \le m$, il en est de même pour *r, y* et *z*.
O_{ijk}	: Représente la $j^{ème}$ opération de la tâche J_i exécutée sur la machine M_k débutant à l'instant S_{ijk} et lui correspondant une durée opératoire t_{ijk} .
t_{ijk}	: La durée opératoire de l'opération O_{ijk}.
S_{ijk}	: La date de début de l'opération O_{ijk}.
$C_{\max}$	: Le temps requis pour accomplir toutes les tâche de

l'ordonnancement appelé Makespan ou encore la somme des dates de fin des tâches.

Chaque machine M_k ne peut traiter à la fois qu'une seule opération O_{ijk} qui ne peut être exécutée que sur une seule machine durant une durée opératoire t_{ijk} .

2. *Les critères de la maintenance*

Les dates de début et les dates de fin des périodes d'indisponibilités sont supposées flexibles pour l'exécution des tâches de maintenances préventives systématiques des machines. Le nombre de périodes d'indisponibilités, appelées souvent trous [KUB02] diffèrent d'une machine à une autre et ne sont pas forcément de la même taille ou uniformément réparties. Et par conséquent les machines considérées sont différentes au niveau de leurs besoins en maintenance durant l'horizon d'ordonnancement.

Le décalage des périodes d'indisponibilité des machines est effectué en fixant un délai maximal toléré pour l'avancer ou le retarder et ceci selon les circonstances imposées par l'affectation des tâches de production par rapport à l'attribution initiale des tâches de maintenance. La procédure de décalage a un effet direct simultané sur le Makespan C_{max} et sur le coût total de la maintenance préventive systématique. Ce coût est égal au coût supposé fixe de la maintenance sans décalage durant le cycle de la production auquel nous ajoutons un coût supplémentaire dû au décalage des indisponibilités.

Ci-dessous on définit les paramètres et les symboles utilisés pour étudier le critère de la maintenance.

Dans ce qui suit, notons que le terme «date initiale de début » désigne la date spécifique fixée d'avance par les praticiens pour exécuter les tâches de maintenance préventives systématiques des machines dans les conditions normales (sans intervenir les contraintes de la production) et une indisponibilité est représentée par une tâche de maintenance.

$s^l_{k\,In}$: La date initiale de début de la $l^{ème}$ indisponibilité sur la machine M_k

$e^l_{k\,In}$: La date initiale de fin de la $l^{ème}$ indisponibilité sur la machine M_k

d^l_k : La durée de la $l^{ème}$ indisponibilité sur la machine M_k $d^l_k = e^l_{k\,In} - s^l_{k\,In}$

l_k : Le nombre d'indisponibilité ou de trous sur la machine, M_k

L_k : La période de la maintenance préventive systématique (indisponibilité) de la machine M_k

$s^l_{k\,Nw}$: La nouvelle date de début de la $l^{ème}$ indisponibilité sur la machine M_k après

une intervention (en avance ou en retard).

$e^{l}_{k\,Nw}$: La nouvelle date de fin de la $l^{ème}$ indisponibilité sur la machine M_k après une intervention (en avance ou en retard).

$\Delta t^{a}_{k\max}$: L'avance maximale tolérée de l'indisponibilité sur la machine M_k

$\Delta t^{d}_{k\max}$: Le retard maximal toléré de l'indisponibilité sur la machine M_k

MC^{a}_{k} : Le coût d'avancement par unité de temps d'une indisponibilité pour la machine M_k (imposé par les praticiens ou les ingénieurs)

MC^{d}_{k} : Le coût de retard par unité de temps d'une indisponibilité pour la machine M_k (imposé par les praticiens ou les ingénieurs)

P^{a}_{k} : Le coût dû à un avancement d'une indisponibilité de la machine M_k

P^{d}_{k} : Le coût dû à un retard d'une indisponibilité de la machine M_k

P_k : Prix unitaire d'une indisponibilité

MC : Le coût supplémentaire total pour un cycle de maintenance où toutes les tâches sont exécutées.

La Figure III. 4 présente le modèle générique du problème du job shop à traiter.

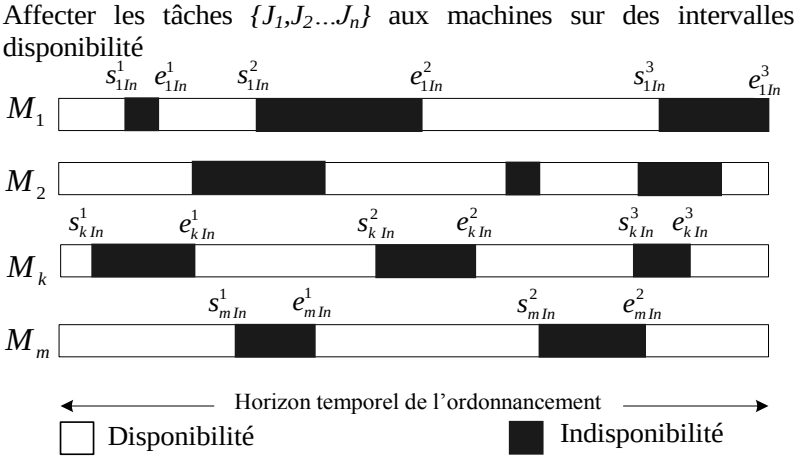


Figure III. 4 : Présentation du modèle générique du problème

Dans cette étude, le critère de maintenance prend en compte uniquement les coûts supplémentaires de maintenance correspondants aux décalages (en avance ou en retard) des indisponibilités des machines. Le coût de la main-d'œuvre n'est pas inclus parce qu'il est supposé connu à l'avance pour chaque tâche de maintenance d'une machine donnée.

De même les tâches de maintenance correctives ainsi que les pannes ne seront pas considérées au cours de cette modélisation de la ligne de production considérée. En effet notre objectif principal est de trouver un ensemble de dates de débuts S_{ijk} qui respectent les contraintes qui tiennent compte des deux critères d'optimisation, la minimisation du Makespan C_{max} et du coût total de la maintenance préventive systématique MC. Ceci sera traité par la formulation mathématique du problème d'ordonnancement job shop avec contraintes d'indisponibilité. Ce problème est désigné par la notation de [GRA79] qui a été étendue par [LAW79], reprise par, [LAW82] et [BLA96] sur trois champs $J/nr/C_{\max}, MC$, où J désigne le système des machines qui est l'ordonnancement dans l'atelier job shop avec des tâches non résumables (nr). Une tâche ou une opération est appelée non resumable (nr), si une fois elle a commencé, elle ne peut pas être interrompue. Le terme MC désigne le second critère à optimiser.

3. *Les contraintes du problème*

Un ordonnancement est dit faisable, s'il respecte toutes les contraintes de l'atelier. En effet l'atelier qui fait l'objet de cette étude doit respecter conjointement trois types de contraintes :

a. Les contraintes de séquencement

La contrainte de séquencement a été expliquée et modélisée au chapitre II par l'équation (II.15) que nous rappelons ci-après :

$$S_{ijl} - S_{i(j-1)k} + t_{i(j-1)k} \geq 0 \qquad \text{(III.1)}$$

b. Les contraintes de ressources ou de machines

En outre cette contrainte a été expliquée et modélisée au chapitre II par les deux équations (II.18-II.19). Mais on rappelle que chaque machine M_k ne peut traiter qu'une seule tâche à la fois. De même, si l'on suppose que la machine M_k est entrain d'exécuter l'opération O_{prk}, cette condition implique que les dates de début des autres opérations qui partagent la même machine M_k doivent respecter la condition suivante décrite par l'équation (III.1).

$$S_{plk} - S_{ijk} + H\left(1 - Y_{ipk}\right) - t_{ijk} \geq 0 \qquad \text{(III.2)}$$

$$S_{ijk} - S_{plk} + HY_{ipk} - t_{plk} \geq 0 \qquad \text{(III.3)}$$

$$S_{ijk} \notin \left[S_{prk}, S_{prk} + t_{prk}\right] \qquad \text{(III.4)}$$

c. Les contraintes d'indisponibilité des ressources

Aucune date de début S_{ijk} d'une opération de production O_{ijk} ne peut intercepter une indisponibilité d'une machine M_k c'est-à-dire pas de chevauchement entre les tâches de production et les indisponibilités. En d'autres termes, n'importe quelle tâche de production doit commencer et finir (en dehors) avant de rencontrer un intervalle d'indisponibilité afin de ne pas perturber l'ordonnancement des tâches de maintenance. De même la répartition des indisponibilités exclut la possibilité de chevauchement entre elles.

Nous pouvons rappeler encore que les intervalles d'indisponibilités ont la flexibilité de décalage dans les deux sens (en avance ou en retard) avec une marge limite préconisée par le constructeur de la machine : Δt_k^a et Δt_k^d, et ce tout en respectant les besoins de la maintenance. Une fois il y a eu un décalage des tâches de maintenance, nous aurons une modification de leurs dates de début et par conséquent, les nouvelles dates $s_{k\,Nw}^l$ et $e_{k\,Nw}^l$ peuvent influer l'affectation des tâches de production aux machines. Ainsi la contrainte d'indisponibilité des ressources sera modélisée par l'équation (III.5).

$$\left[S_{ijk}, S_{ijk} + t_{ijk}\right] \cap \left[s_{k\,Nw}^l, e_{k\,Nw}^l\right] = \varnothing \qquad \text{(III.5)}$$

Donc la résolution du problème d'ordonnancement conjoint de la production et de la maintenance d'un atelier de type job shop consiste à trouver les différentes dates de début des opérations de productions s_{ijk} convenables tout en satisfaisant les différentes contraintes citées au-dessus et données par les équations (III.2-III.3) et (III.4-III.5) dont l'objectif est l'obtention conjointement d'une valeur minimale du Makespan C_{max} avec un coût de maintenance MC minimal. Ceci constitue notre nouvel algorithme de décalage que nous présentons ci-dessous.

B. Proposition d'un nouvel algorithme de décalage pour la résolution du problème d'ordonnancement conjoint de la production et de la maintenance dans un atelier job shop

Dans l'environnement d'atelier de type job shop, les problèmes d'ordonnancement sont de plus en plus complexes notamment avec les contraintes d'indisponibilité de ressource. C'est pourquoi dans la littérature, on trouve différentes méthodes (mathématiques, heuristiques…) et différents algorithmes appliqués pour résoudre ce genre de problème. Nous proposons un algorithme de décalage qui est de type heuristique pour résoudre le problème considéré au cours de ce travail. Nous définissons, dans ce qui suit, quelques notations qui seront utilisées

tout au long de notre étude. Le Tableau III. 2 illustre un cas général d'une gamme opératoire d'un problème d'ordonnancement job shop composé de *n* tâches et de *m* machines donnant les machines et les temps opératoires correspondants à chaque opération de chaque tâche. De même le Tableau III. 3. présente la distribution des tâches de maintenances appropriées pour les différentes machines de l'atelier durant un horizon d'un cycle de production.

	Machines (durée opératoire)					
Tâches	Opérations					
	1	2	.	*k*	.	*r*
1	5(3)	3(5)	.	$h(t_{1kh})$		$f(t_{1rf})$
2	2(6)	1(4)	.	.	.	.
.	.	.	.	.	.	.
i	$h(t_{i1h})$	.	.	$f(t_{ikf})$	.	$s(t_{irs})$
.	.	.	.	.	.	.
n	$f(t_{n1f})$	.	.	$s(t_{nks})$		$h(t_{nrh})$

Tableau III. 2 : Gamme opératoire d'un job shop de taille $n \times m$ (même que le Tableau II. 2 du chapitre II)

Machines	Réparation des indisponibilités sur les machines
M_1	$\left\{\left[s_{1In}^{1}, e_{1In}^{1}\right], \ldots \left[s_{1In}^{i}, e_{1In}^{i}\right], \ldots \left[s_{1In}^{m1}, e_{1In}^{m1}\right]\right\}$
M_2	$\left\{\left[s_{2In}^{1}, e_{2In}^{1}\right], \ldots \left[s_{2In}^{i}, e_{2In}^{i}\right], \ldots \left[s_{2In}^{m2}, e_{2In}^{m2}\right]\right\}$
M_h	$\left\{\left[s_{hIn}^{1}, e_{hIn}^{1}\right], \ldots \left[s_{hIn}^{i}, e_{hIn}^{i}\right], \ldots \left[s_{hIn}^{mh}, e_{hIn}^{mh}\right]\right\}$
M_r	$\left\{\left[s_{rIn}^{1}, e_{rIn}^{1}\right], \ldots \left[s_{rIn}^{i}, e_{rIn}^{i}\right], \ldots \left[s_{rIn}^{mr}, e_{rIn}^{mr}\right]\right\}$

Tableau III. 3: Distribution des tâches de maintenance pour les différentes machines

L'idée principale de l'algorithme proposé consiste à utiliser l'approche simulatoire pour la résolution du problème d'ordonnancement [LOP01]. Cette technique a été présentée au chapitre I en mettant l'accent sur son utilisation pour résoudre le problème d'ordonnancement job shop. Et d'après Lopez [LOP01] :

« La simulation est donc une méthode d'évaluation de performances (temporelles ou non) compte tenu de la façon dont est réalisée la conduite du flux de travaux à l'intérieur du système ».

Donc la procédure de décalage est effectuée en balayant le temps itérativement et en sélectionnant la règle adéquate à appliquer et ceci parmi les règles qui seront présentées et détaillées ci-dessous, [YAH09], [YAH11b].

1. *Règle 1 : Initialisation des dates de début de chaque opération*

L'application de la 1ère règle, consiste à supposer que les ressources sont toutes disponibles pour exécuter les opérations de chaque tâche, et ce indépendamment des autres. C'est pour cette raison que nous allons prendre en considération que les contraintes de précédence qui seront appliquées uniquement pour initialiser les dates de début de toutes les opérations de toutes les tâches.

Nous rappelons que la première règle a fait l'objet d'une méthode d'initialisation pour le réseau de neurone de type Hopfield pour l'ordonnancement job shop que nous avons proposé au chapitre II pour accélérer l'algorithme de Willems et Brandts [WIL95] afin de trouver une solution optimale ou proche de l'optimale.

Par exemple, la procédure d'initialisation pour la tâche i ($i^{ème}$ ligne du Tableau (II.2)) se déroule comme suit : nous commençons à la date de début S_{i1h} =0 par attribuer la machine h à la première opération de la tâche i (O_{i1h}) durant son temps de traitement t_{i1h}. En deuxième étape et à la date $S_{i2l}= t_{i1h}$, nous attribuons la machine l à la deuxième opération de la même tâche i (O_{i2l}) durant un temps de traitement t_{i2l} et ainsi de suite pour le reste des opérations de la même tâche i. Et c'est ainsi que nous organisons les différentes opérations du problème général d'ordonnancement job shop modélisé suivant le graphe disjonctif tel qu'il est présenté par le Tableau (III. 2). Le modèle général résultant de l'initialisation que nous avons opté est illustré par la Figure III.5, [YAH09], [YAH11b].

Les valeurs des dates d'initialisations résultantes seront illustrées par la Figure III. 5.

Si à un certain moment non spécifié t, plusieurs opérations des diverses tâches sont dans une file d'attente, prêtes à être exécutées par une machine M_k, nous choisissons une parmi elles en utilisant une règle de priorité entre les tâches. Généralement, cette règle de priorité adoptée est une décision prise par les ingénieurs de production ou les praticiens. Par exemple, une tâche i est plus prioritaire à l'exécution que la tâche $i+1$.

Si on choisit l'exécution de l'opération O_{ijk} sur la machine M_k, ceci implique nécessairement le retardement des autres opérations de la file d'attente par un temps $\Delta t_2 = t_{ijk}$. Cette règle est illustrée par un exemple représenté à travers la Figure III. 6.

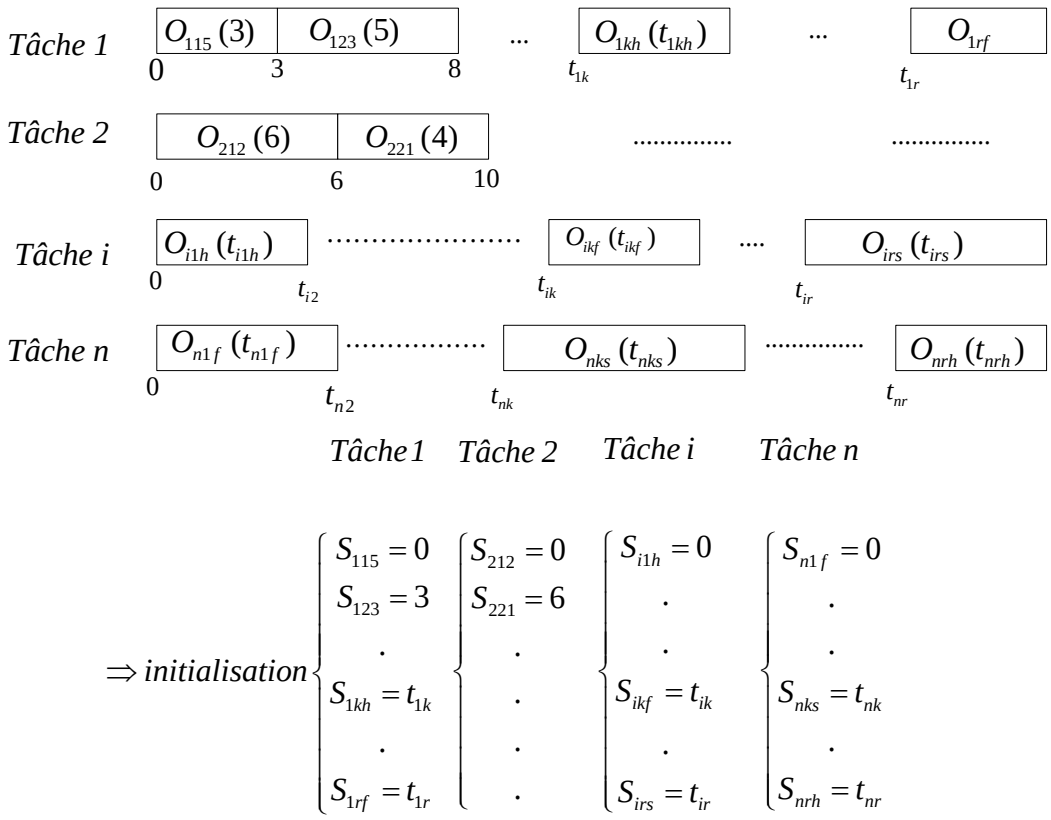


Figure III. 5 : Initialisation d'un problème d'ordonnancement job shop en appliquant la règle 1

2. *Règle 2 : La décision à prendre dans le cas d'une file d'attente*

Si à un certains moment non spécifié t, plusieurs opérations des diverses tâches sont prêtes à être exécutées par une machine M_k, nous choisissons une parmi elles en utilisant une règle de priorité entre les tâches. Généralement, cette règle de priorité adoptée est une décision prise par les ingénieurs de production ou les praticiens. Par exemple, une tâche i est prioritaire à l'exécution que la tâche $i+1$.

Si on choisit de sélectionner l'exécution de l'opération O_{ijk} sur la machine M_k, ceci implique nécessairement le retardement des autres opérations en attente par un temps $\Delta t_2 = t_{ijk}$. Cette règle est illustrée par un exemple dans la Figure 3.3.

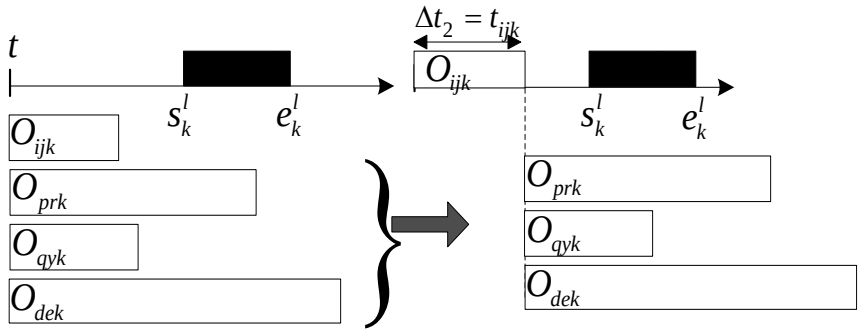


Figure III. 6 : Affectation des opérations suivant la règle 2

3. *Règle 3 : Décalage des opérations*

Le décalage de n'importe qu'elle opération O_{ij} par un temps Δt_3 implique automatiquement le décalage de toutes les opérations de la tâche i $(O_{i(j+1)}, O_{i(j+2)})...(O_{i(j+n)})$ qui suivent cette opération O_{ij} par le même retard Δt_3. La présente règle est modélisée par la Figure III. 7.

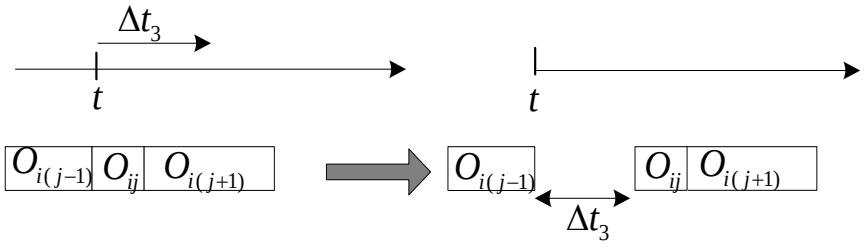


Figure III. 7 : Décalage des opérations suivant la règle 3

4. *Règle 4 : Retardement ou avancement des périodes d'indisponibilité*

Cette règle sera applicable dans les conditions suivantes : si nous vérifions qu'aucune opération de production parmi la liste des opérations en file d'attente à exécuter sur la machine M_k , ne puisse être achevée avant que la machine devienne disponible.

Face à cette situation, nous pouvons envisager deux cas possibles de la règle 4 (règle 4a et règle 4b) :

a. Premier cas de la règle 4a:

S'il est possible de sélectionner une opération pour l'exécuter sur la machine M_k en retardant la période d'indisponibilité par un temps $\Delta t_k^d \le \Delta t_{k\max}^d$ et à condition qu'elle soit achevée

avant qu'on atteigne un intervalle d'indisponibilité (avec ce retard). Ainsi les deux nouvelles dates de début et de fin de l'indisponibilité deviennent respectivement (III.6-III.7):

$$s^l_{k\,Nw} = s^l_{k\,In} + \Delta t^d_k \quad \text{(III.6)}$$

et

$$e^l_{k\,Nw} = e^l_{k\,In} + \Delta t^d_k \quad \text{(III.7)}$$

Concernant les autres opérations (production), elles devraient être décalées par un temps Δt_{4a} décrit par l'équation (III.8).

$$\Delta t_{4a} = t_{ijk} + (e^l_{k\,Nw} - s^l_{k\,In}) \quad \text{(III.8)}$$

Sachant que le temps opératoire de l'opération choisie est t_{ijk}. Ce retard de l'intervalle d'indisponibilité entraine un coût supplémentaire de maintenance P^d_k exprimé par l'équation (III.9).

$$P^d_k = \Delta t^d_k \times MC^d_k \quad \text{(III.9)}$$

Cette situation peut traduire la règle 4a qui peut être illustrée par la Figure III.8.

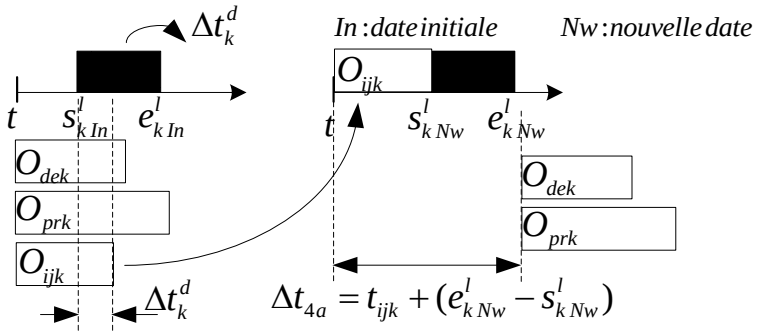


Figure III. 8 : Affectation des opérations suivant la règle 4a

b. Deuxième cas la règle 4b, (Figure III.9):

Puisqu'aucune opération de production ne peut être exécutée et ne puisse finir avant l'arrivée de l'intervalle d'indisponibilité, il est nécessaire de l'avancer par un temps Δt^a_k et par conséquent avancer respectivement les dates de début et de fin de l'indisponibilité de la machine M_k, [YAH11b]. Ainsi les deux nouvelles dates de début et de fin de l'indisponibilité s'expriment respectivement suivant les équations (III.10-III.11) :

$$s^l_{k\,Nw} = s^l_{k\,In} - \Delta t^a_k \quad \text{(III.10)}$$

et

$$e^l_{k\,Nw} = e^l_{k\,In} - \Delta t^a_k \qquad \text{(III.11)}$$

La valeur de Δt^a_k est choisie comme suit : Δt^a_k =min de$\left[\Delta t^a_{k\max}, (s^l_{k\,In} - t)\right]$, où t est l'instant d'exécution immédiate de cette règle.
Dans cette situation, toutes les opérations de la file d'attente devraient être décalées par Δt_{4b}.

$$\Delta t_{4b} = e^l_{k\,Nw} - t \qquad \text{(III.12)}$$

De la même manière, l'avancement de l'intervalle d'indisponibilité entraine un coût supplémentaire de maintenance égale à:

$$P^a_k = \Delta t^a_k * MC^a_k \qquad \text{III.13)}$$

Cette règle peut être illustrée par la Figure III. 9.

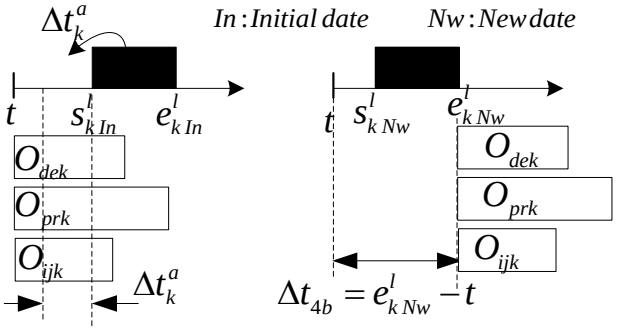


Figure III. 9 : Affectation des opérations suivant la règle 4b

5. *Règle 5 : Décalage de la liste d'attente une fois elle est située dans une indisponibilité*

Si l'instant t de l'ordonnancement d'une file d'attente des opérations de production se situe dans une indisponibilité; c'est-à-dire que $(s^l_k < t < e^l_k)$, la liste sera automatiquement décalée par Δt_5, comme l'illustre la Figure III.10.

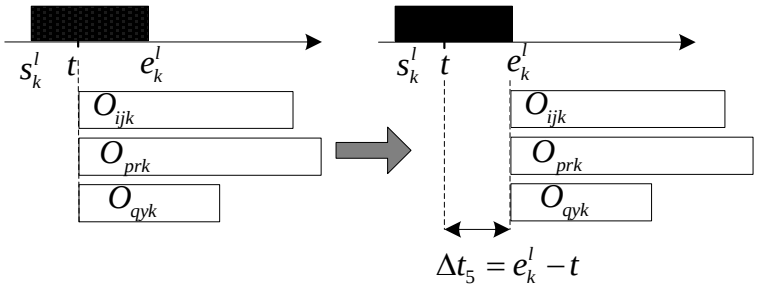


Figure III. 10 : Affectation des opérations suivant la règle 5

6. *Règle 6 : Décalage de la liste d'attente lorsque la machine traite une autre opération*

Cette règle s'applique, quand une liste d'opérations est attribuée à la même ressource k à un instant t, pendant qu'elle traite une autre opération O_{ijk}. Dans ces conditions, cette liste sera retardée par un temps Δt_6 comme le montre la Figure III. 11.

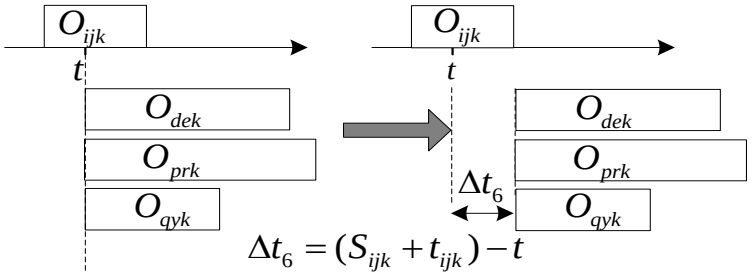


Figure III. 11 : Décalage de la liste d'attente suivant la règle 6

C. Exemple de simulation illustratif

A travers un exemple qui est généré et construit à partir d'un benchmark standard de l'atelier job shop, nous présentons les étapes de l'exécution de cette nouvelle proposition, en balayant le temps et en sélectionnant à chaque instant, la règle à appliquer.

En effet, l'exemple à tester, est un atelier de type job shop constitué de 4 tâches et de 3 machines. Le Tableau III. 4 présente les données de production de ce benchmark à savoir les machines et les temps opératoires correspondants aux opérations des différentes tâches. Pour les données de la maintenance, les machines M_1 et M_2 sont soumises à deux périodes de maintenance alors que la machine M_3 n'est soumise qu'à une seule. Les périodes de maintenance des machines sont comme suit : M_1[6 9], [12 15] signifie que la machine M_1 n'est pas disponible entre les dates de 6 et 9 et entre les dates 12 et 15, M_2[4 6], [8 10], M_3[6 9]. De même le Tableau III. 5 présente d'autres données de maintenance spécifiques à chaque machine.

	Machine M_k (Temps opératoire : t_{ijk})		
	Opérations		
Tâches	1	2	3
1	M_1 (4)	M_2 (3)	M_3 (2)
2	M_2 (1)	M_1 (4)	M_3 (4)
3	M_3 (3)	M_2 (2)	M_1 (3)
4	M_2 (3)	M_3 (3)	M_1 (1)

Tableau III. 4 : Gamme opératoire d'un job shop de taille 4×3 (4 tâches, 3 machines)

	Prix unitaire d'indisponibilité P_k	Marge d'avance $\Delta t^a_{k\max}$	Prix unitaire d'avance MC^a_k	Marge de retard $\Delta t^d_{k\max}$	Prix unitaire de retard MC^d_k
M_1	5	2	8	3	12
M_2	4	3	10	4	3
M_3	6	2	6	3	4

Tableau III. 5 : Données de maintenance le long d'un cycle de production

Ainsi l'algorithme ou l'heuristique proposée le long d'un cycle de production, commence toujours par appliquer les différentes règles expliquées précédemment.

La première étape consiste toujours par appliquer la règle 1 « Initialisation des dates de début des opérations» [YAH11a]. Le tableau III. 6 illustre l'initialisation des différentes dates de début des opérations et ce conformément aux données du Tableau III. 4.

Tâche	Date de début		Tâche	Date de début
Tâche 1	$S_{111}=0$		Tâche 2	$S_{212}=0$
	$S_{122}=4$			$S_{221}=1$
	$S_{133}=9$			$S_{233}=5$
Tâche 3	$S_{313}=0$		Tâche 4	$S_{412}=0$
	$S_{322}=3$			$S_{423}=3$
	$S_{331}=8$			$S_{431}=6$

Tableau III. 6 : Initialisation des dates de début selon la règle 1

La Figure III. 12 représente l'allocation des opérations aux machines à l'instant $t=0$ conformément à l'initialisation adoptée en appliquant la règle 1.

Donc relativement à la Figure III. 12, la machine M_1 peut exécuter sans aucun empêchement l'opération O_{111} ce qui entraine une valeur de sa date de début $S_{111}=0$ (valeur définitive). De même pour la machine M_3 , elle peut exécuter l'opération O_{313} ce qui entraine une valeur de sa date de début $S_{313}=0$ (valeur définitive). Tandis que la machine M_2 a le choix d'exécuter O_{212} ou O_{412}.

Selon la règle de priorité sélectionnée entre les tâches (la règle 2) d'une part, et en appliquant et la règle 3 d'autres part, la machine M_2 sélectionne O_{212} pour l'exécuter avant O_{412}, d'où : $S_{212}=0$ (valeur définitive) alors que S_{412} passe de $t=0$ à $t=1$ (Règle 2) et en appliquant la règle $3 \Rightarrow \begin{cases} S_{423} \text{ passe de 3 à 4} \\ S_{431} \text{ passe de 6 à 7} \end{cases}$.

Ainsi nous obtenons les nouvelles occupations de chaque machine à l'instant t=1 (Figure III.13).

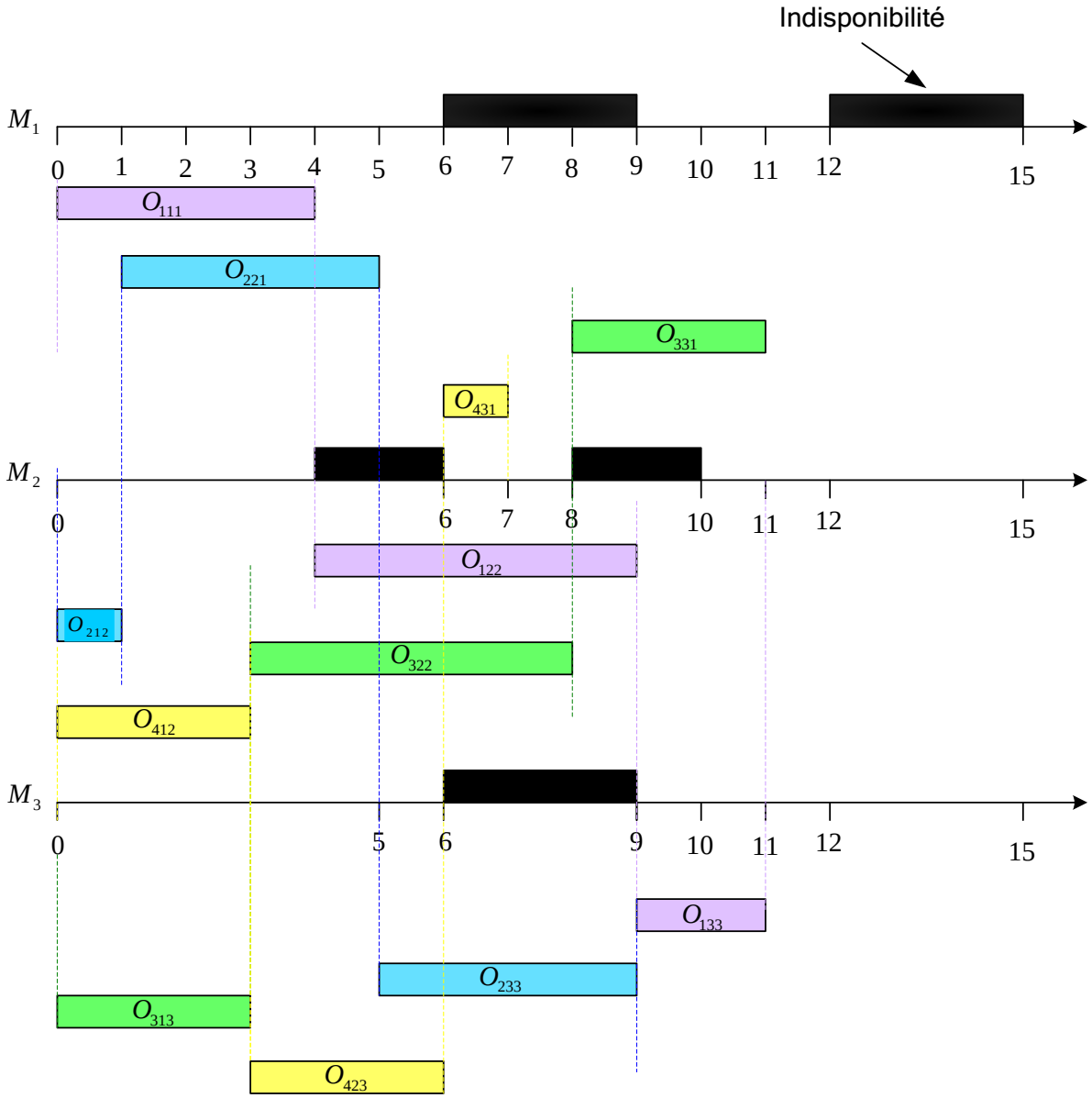


Figure III. 12 : Allocation des opérations aux machines suivant la règle 1 à l'instant t=0

En passant à l'instant $t=1$, nous remarquons que la machine M_1 est entrain d'exécuter O_{111} jusqu'à l'instant $t=4$ donc en appliquant la règle *6*, l'opération O_{221} doit avoir la nouvelle date de début $S_{221}=4$, qui reste provisoire. Selon la règle 3, le déplacement de O_{221} entraine d'une façon automatique le déplacement de O_{233} c'est-à-dire S_{233} passe de *5* à *8*.

La machine M_2 achève l'exécution de l'opération O_{212} et selon la règle 2, M_2 a la possibilité d'exécuter l'opération O_{412} donc $S_{412}=1$ (valeur définitive).

La machine M_3 est entrain d'exécuter l'opération O_{313} et il n'existe aucun empêchement (contraintes) avec les autres opérations, d'où l'allocation définitive des 3 machines à l'instant $t=2$ est développée par la Figure III.14 :

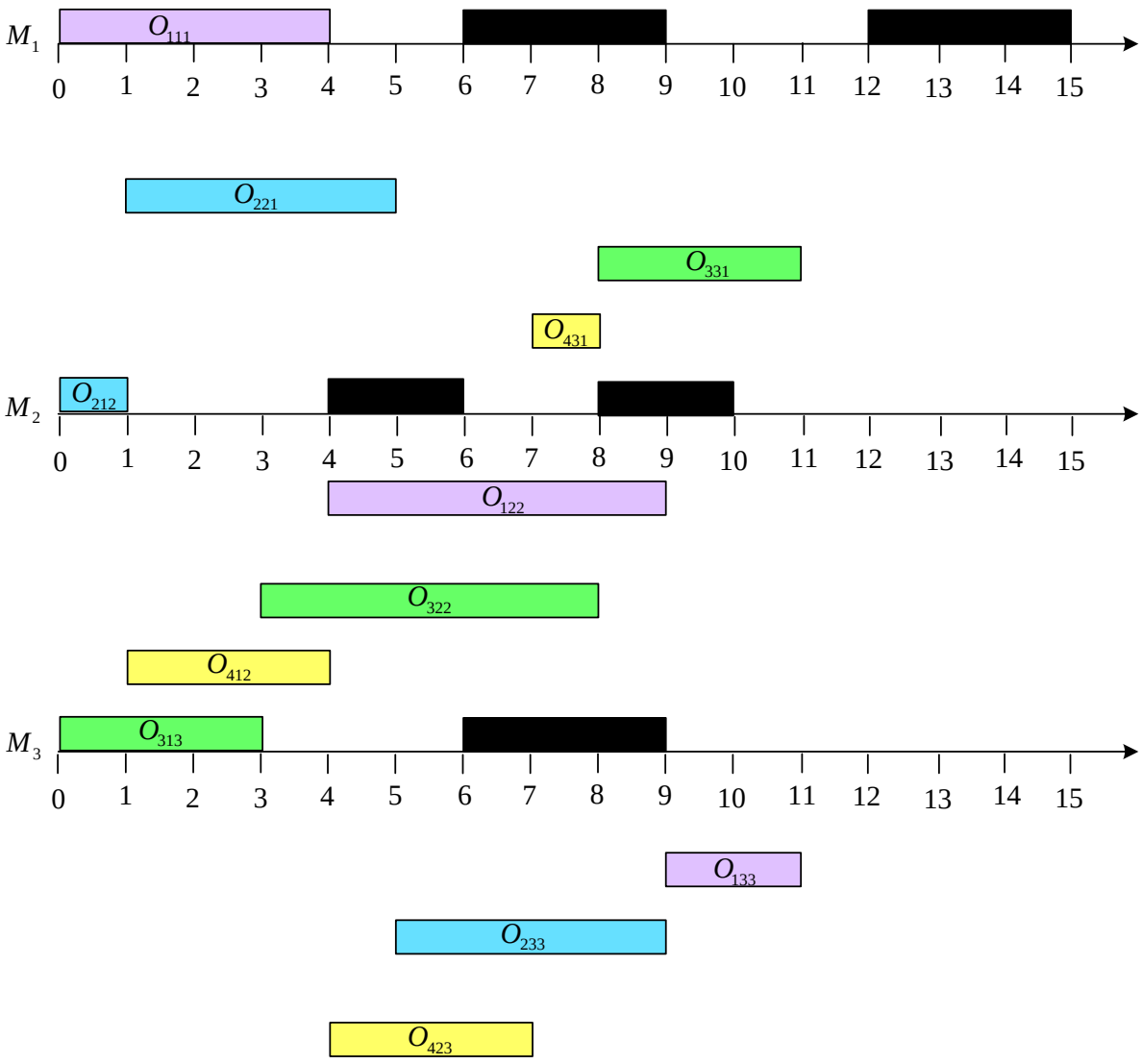


Figure III. 13: Allocation des opérations aux machines suivant la règle 1 à l'instant $t=1$

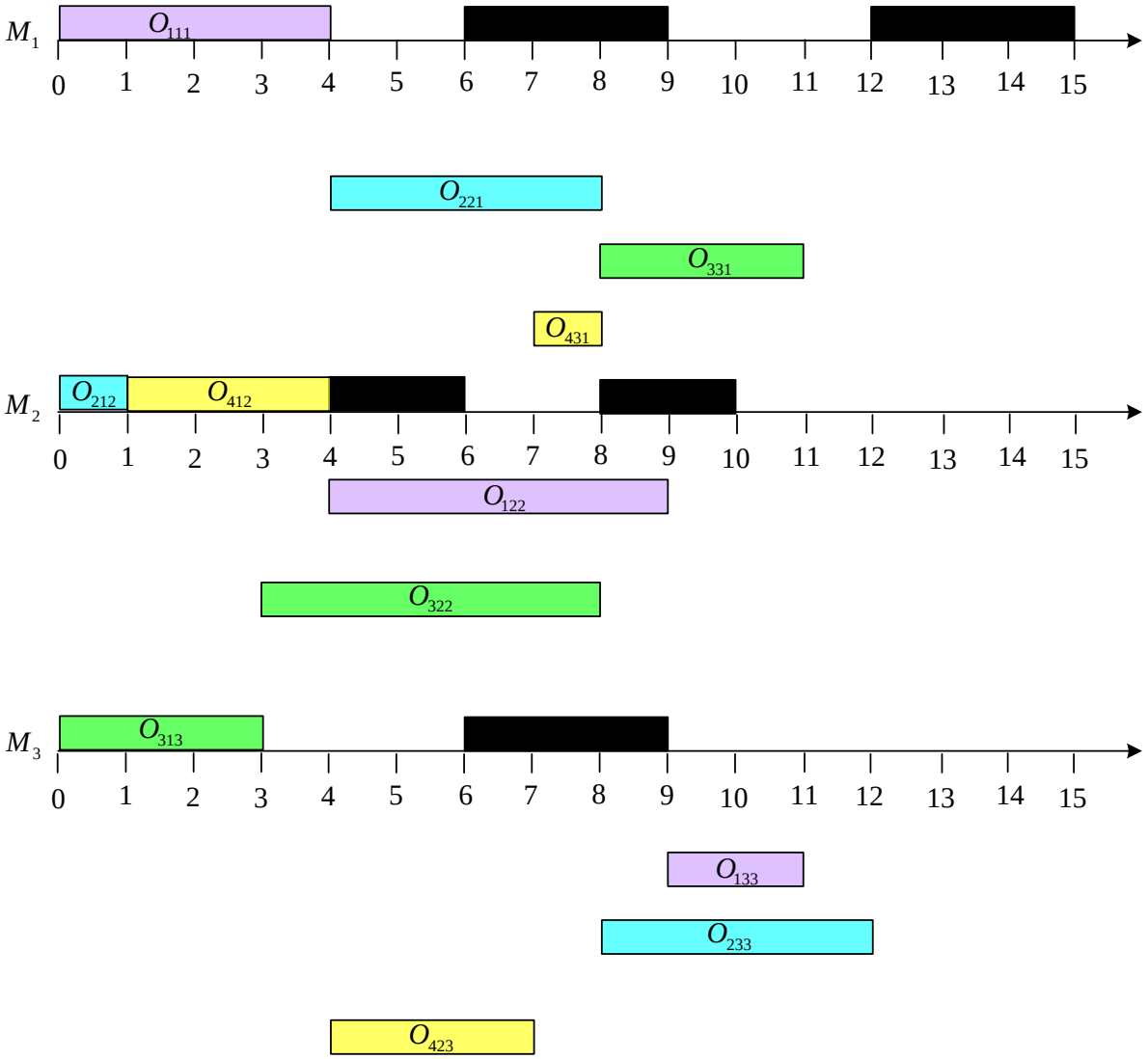


Figure III. 14 : Allocation des opérations aux machines suivant la règle 1 à l'instant $t=2$

A l'instant $t=2$ rien ne se passe, l'état des machines de l'atelier reste identique à celui à l'instant $t=1 \Rightarrow \begin{cases} M_1 \rightarrow O_{111} \\ M_2 \rightarrow O_{212} \\ M_3 \rightarrow O_{313} \end{cases}$

En passant à l'instant $t=3$: rien ne se passe pour les machines M_1 et M_3, tandis que pour la machine M_2 en appliquant la règle 6, l'opération O_{322} sera décalée par $\Delta t=1$. C'est à dire S_{322} passe de 3 à 4 et par conséquent, selon la règle 3 ; O_{331} se déplace et sa date de début S_{331} passe de 8 à 9.

L'allocation définitive des 3 machines à l'instant $t=3$ est développée par la Figure III.15 :

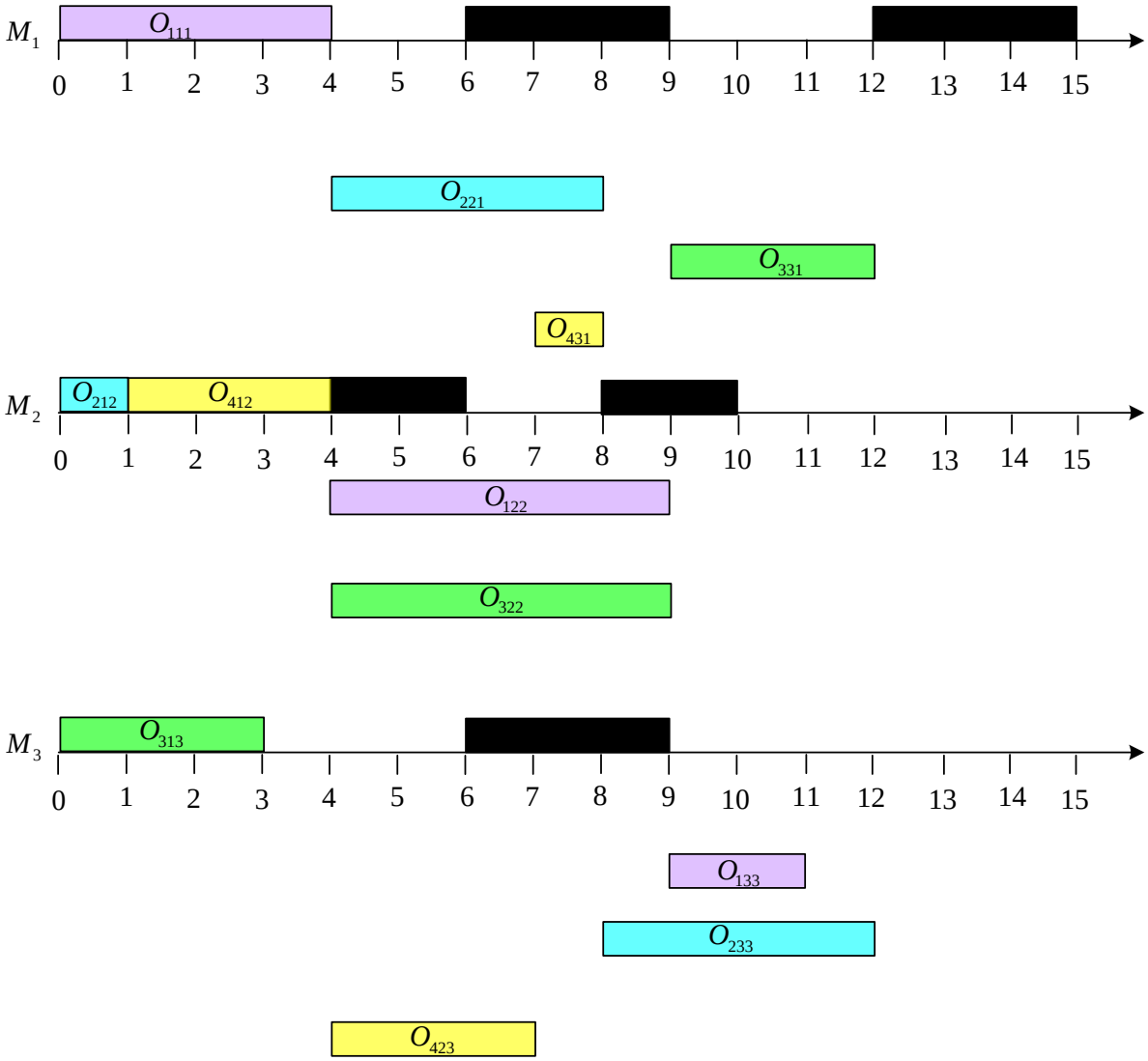


Figure III. 15 : Allocation des opérations aux machines suivant la règle 1 à l'instant $t = 3$

A l'instant $t = 4$:

Pour la machine M_1, la règle 4 à ce niveau est utile :

- Selon la règle 4a, en retardant l'intervalle d'indisponibilité par $\Delta t_r = 2 \Rightarrow$ le Coût du retard $= 2 \times 12 = 24$.
- Selon la règle 4b : en avançant l'intervalle d'indisponibilité par $\Delta t_a = 2 \Rightarrow$ le Coût d'avance $= 2 \times 8 = 16$.

Puisque le Coût d'avance < Coût du retard donc nous optons aux choix d'avancer l'intervalle d'indisponibilité par $\Delta t_a = 2$ et de retarder l'opération O_{221} par $\Delta t_{4b} = 3 \Rightarrow$ ainsi S_{221} passe de

4 à 7. Une fois l'opération O_{221} à été déplacée, et d'après la règle 3 l'opération O_{233} se déplacera aussi d'où S_{233} passe de 8 à 11.

Pour la machine M_2, il est évident d'appliquer la règle 5 c'est-à-dire : $Règle\,5 \Rightarrow \begin{cases} S_{122} \\ S_{322} \end{cases}$

passe de 9 à 6
passe de 4 à 6

Par conséquent et selon la règle 3 : $\begin{cases} S_{133} \text{ passe de 9 à 11} \\ S_{331} \text{ passe de 9 à 11} \end{cases}$

La machine M_3 est libre, d'où nous pouvons appliquer la règle 4.

- Selon la règle 4a : en retardant l'intervalle d'indisponibilité par $\Delta t_r = 1 \Rightarrow$ le Coût de retard est égale à $1 \times 4 = 4$.
- Selon la règle 4b : en avançant l'intervalle d'indisponibilité par $\Delta t_a = 2 \Rightarrow$ Coût d'avance est égale à $2 \times 6 = 12 \Rightarrow$ Coût du retard < Coût d'avance. Donc nous optons au choix de retarder l'intervalle d'indisponibilité par $\Delta t_r = 1$ et par conséquent la machine M_3 peut exécuter O_{423} donc $S_{423} = 4$ (valeur définitive) et la nouvelle situation à l'instant $t = 4$ est fournie par la Figure III. 16.

Remarque importante:

Tout avance ou retard d'un intervalle d'indisponibilité entraine nécessairement l'avance ou le retard des autres intervalles d'indisponibilité qui suivent.

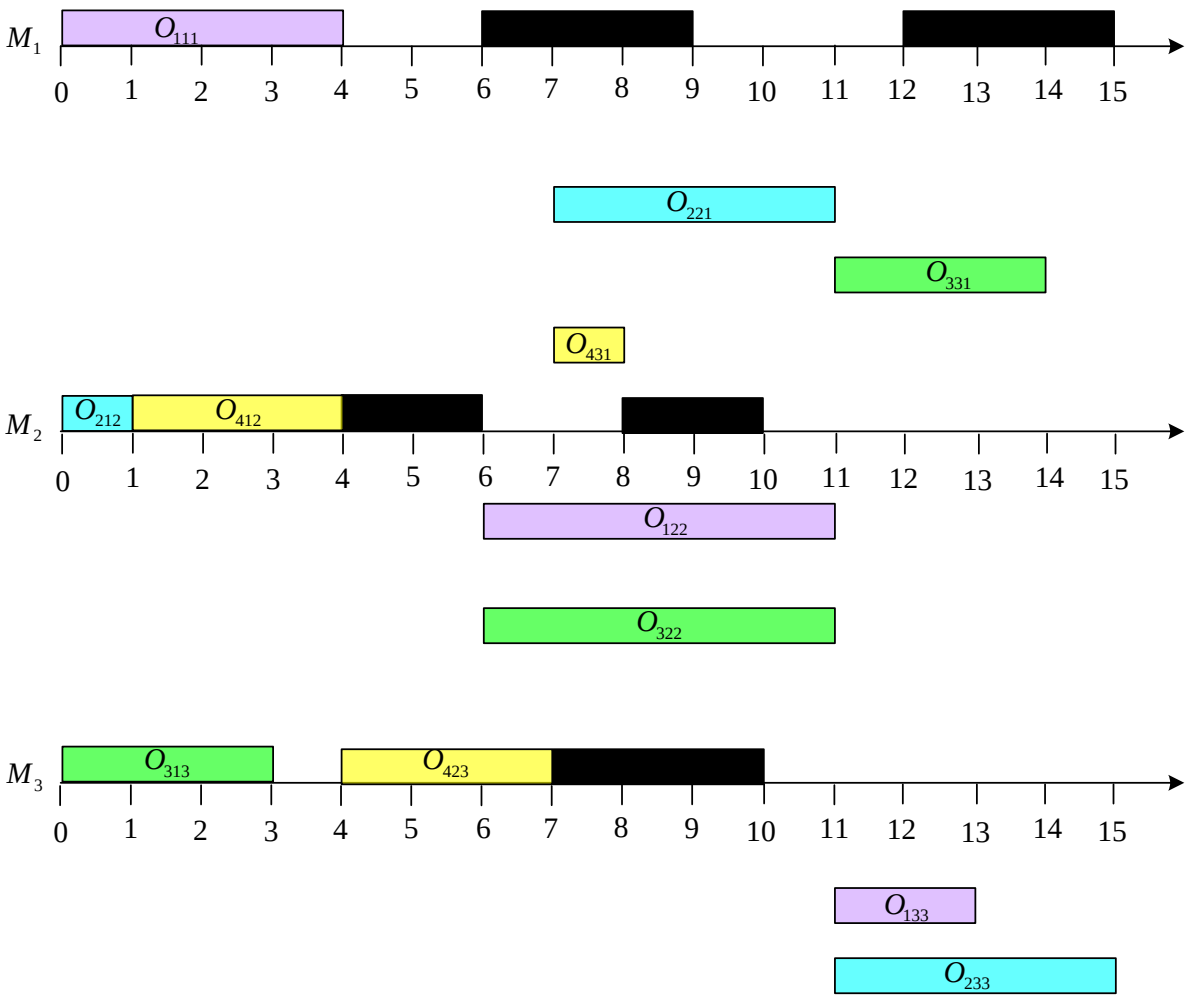


Figure III. 16 : Allocation des opérations aux machines suivant la règle 1 à l'instant $t = 4$

La situation de la Figure III.16 est obtenue avec un coût supplémentaire due aux déplacements des intervalles d'indisponibilité par machine valant :

Coût sup $\begin{cases} 16 \text{ pour } M_1 \\ 4 \text{ pour } M_3 \end{cases}$ et ainsi de suite, en appliquant l'une des règles proposées dans la sous section 4.B jusqu'à trouver les différentes valeurs définitives des dates de début S_{ijk} des différentes opérations. Ainsi s'achève cette résolution après avoir affecté la totalité des opérations sur leurs machines et trouver le coût supplémentaire total d'un cycle complet de production tout en garantissant un Makespan $C_{\max}$ optimal.

Le résultat final correspondant à la simulation du premier benchmark généré est présenté par le diagramme de Gantt de la Figure III.17.

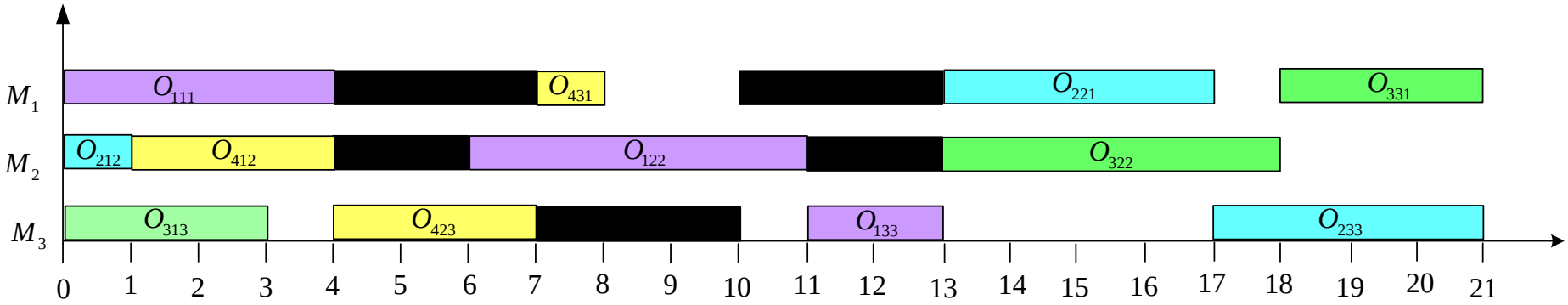


Figure III. 17: Allocation finale des opérations aux machines

Remarques:

Dans le but de mieux expliquer les coûts supplémentaires résultant aux déplacements des périodes d'indisponibilité, le Tableau III. 7 donne les détails définitifs de calcul de ces coûts.
Le Makespan C_{max}= 21(unités de temps)

	Mouvement → Coût	Machine
Coûts supplémentaires	$-2 \xrightarrow{\times 8} -16$	Pour M_1
	$+3 \xrightarrow{\times 3} 9$	Pour M_2
	$+1 \xrightarrow{\times 4} 4$	Pour M_3

Tableau III. 7 : Détails de calcul des coûts de déplacement des indisponibilités

Nota :

Le signe (-) est attribué aux déplacements dus à l'avance de l'indisponibilité et le signe (+) est attribué aux déplacements dus au retard.

Le Coût total de maintenance/cycle de production = Coût des intervalles d'indisponibilité pendant un cycle + les Coûts supplémentaires totaux dus aux déplacements des intervalles d'indisponibilités.

$$\textit{Le Coût total de maintenance par cycle} = \left[\left[(7-4)+(13-10)\right]\times 5+\left[(6-4)+(13-11)\right]\times 4+(10-7)\times 6\right]+16+9+4=93 \qquad \text{(III.10)}$$

D. Résultats de simulation

Afin d'évaluer l'heuristique proposée, cette nouvelle méthode de décalage a été testée pour résoudre le problème d'ordonnancement conjoint de la production et de la maintenance sur deux benchmarks générés et construits à partir de deux benchmarks standards de l'atelier job shop généralisé. Le premier benchmark est de petite taille il est formé de 2 tâches et 3 machines (Tableau III. 8), le second est aussi considéré de petite taille ; constitué de 6 tâches et de 6 machines noté dans la littérature par ft06 (Tableau III. 11).

Machine (durée opératoire)			
Tâche	opérations		
	1	2	3
1	1(5)	2(8)	3(2)
2	3(7)	1(3)	2(9)

Tableau III. 8 : Gamme opératoire (machines) et durée opératoire des tâches du problème *2/3/J/*C_{max}.

L'application de cet algorithme est faite en deux étapes :

a. En première étape, une simulation détaillée est exposée pour le premier benchmark en considérant les trois situations de l'atelier à savoir :
   - Ordonnancement sans contraintes d'indisponibilité
   - Ordonnancement avec contraintes d'indisponibilité (fixes)
   - Ordonnancement avec contraintes d'indisponibilité (avec décalage)
b. En deuxième étape, une présentation des résultats de simulation pour l'exemple 2 (6 tâches/6 machines) ft06 suivis par quelques statiques et commentaires.

1. *Première étape de simulation*

a. Ordonnancement sans contraintes d'indisponibilité

Les résultats de simulation sans contraintes d'indisponibilité pour le premier exemple seront visualisés par des diagrammes de Gantt (machines et tâches). La Figure III. 18 présente l'allocation des tâches alors que la Figure III. 19 présente l'allocation des machines.

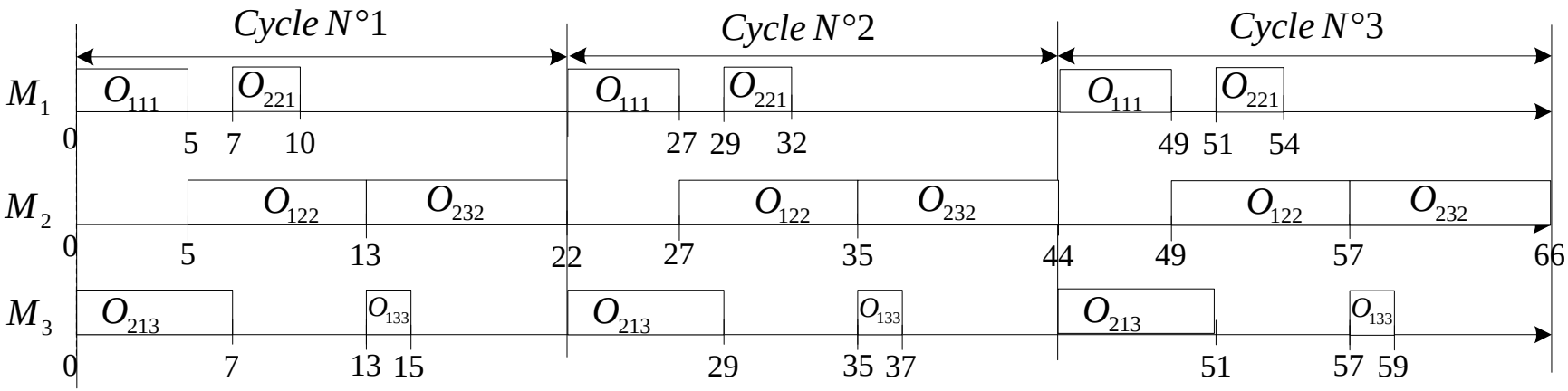


Figure III. 18: Allocation des machines sans contraintes de maintenance

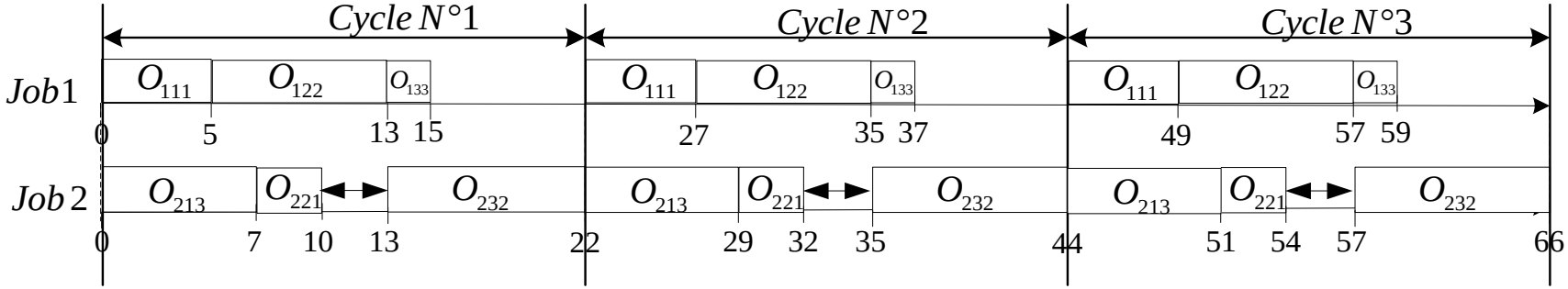


Figure III. 19 : Allocation des tâches sans contraintes de maintenance

b. Ordonnancement avec des contraintes d'indisponibilité (fixes)

Le Tableau III. 9 présente les donnés numériques de la maintenance respectivement aux machines pour le premier benchmark [YAH09].

Machine	M_1	M_2	M_3
Période (L_k)	14	18	10
Durée d_k^l	4	5	3
Prix unitaire d'indisponibilité P_k	10	12	8
Marge d'avance $\Delta t_{k\max}^a$	2	4	1
Marge de retard $\Delta t_{k\max}^d$	3	4	1
Prix unitaire d'avance MC_k^a	8	7	3
Prix unitaire de retard MC_k^d	12	10	5

Tableau III. 9 : Données de la maintenance pour le premier benchmark

Voici les définitions des deux paramètres de simulations suivant dont nous aurons besoin pour expliquer les détails de simulation :

- Le coût total : C'est le coût de maintenance dû à la somme de durée d'indisponibilité de période en plus de la pénalité additionnelle pour chaque unité de temps (de retard ou d'avance) de délai.

- Le taux de temps libre : c'est le pourcentage du temps de retard dans un cycle dans lequel la tâche ne peut être exécutée (arrêt) faute de contraintes de tâches de maintenance (l'objectif est d'avoir une valeur près de 0% c'est-à-dire sans décalage entre les opérations d'une même tâche).

Après avoir alloué les tâches de production du premier benchmark sans tenir compte des tâches de maintenance (Figures III. 18-III. 19), nous proposons en deuxième étape de répartir les périodes de maintenance préventives systématiques durant trois cycles de production afin d'ordonnancer de nouveau les tâches de production. Les Figure (III. 20-III. 21) illustrent les résultats de la simulation dans les conditions décrites, les rectangles noirs représentent les périodes de maintenance sur le diagramme de Gantt.

c. Ordonnancement avec contraintes d'indisponibilité (avec décalage)

Dans ce cas nous aurons un ordonnancement conjoint de la production et de la maintenance et ceci lorsque les périodes de maintenances peuvent tolérer un certain décalage en respectant les données du Tableau III. 10. Les critères à minimiser sont le Makespan C_{max} et le coût total de la maintenance. Les Figures (III. 22-III. 23) présentent les résultats de simulation correspondants à l'exemple 1. De même le Tableau III. 11, présente un résumé sur les résultats de simulation du même exemple (benchmark 1 du Tableau III. 8).

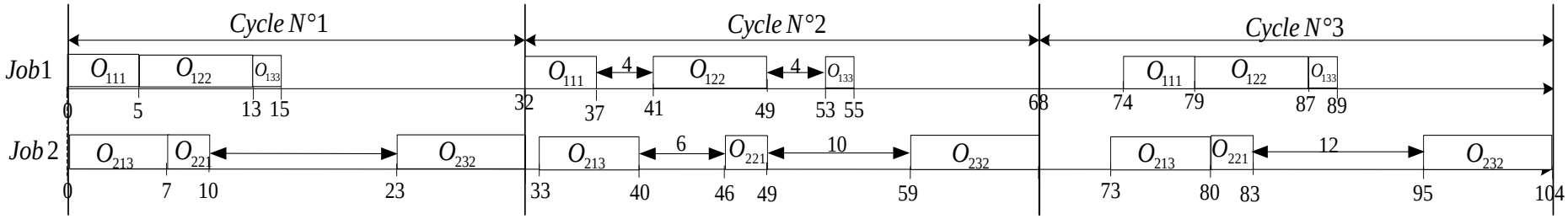


Figure III. 20 : Allocation des tâches avec contraintes de tâches de maintenance (fixe**)**

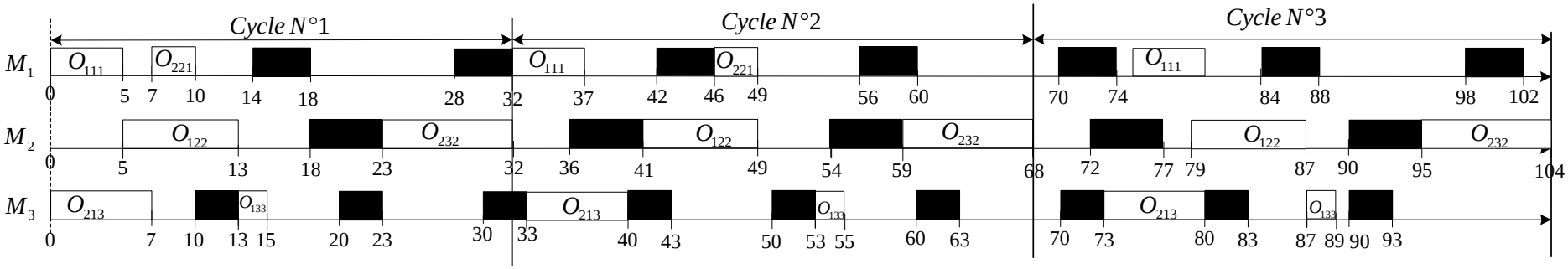


Figure III. 21 : Allocation des machines avec contraintes de tâches de maintenance (fixe)

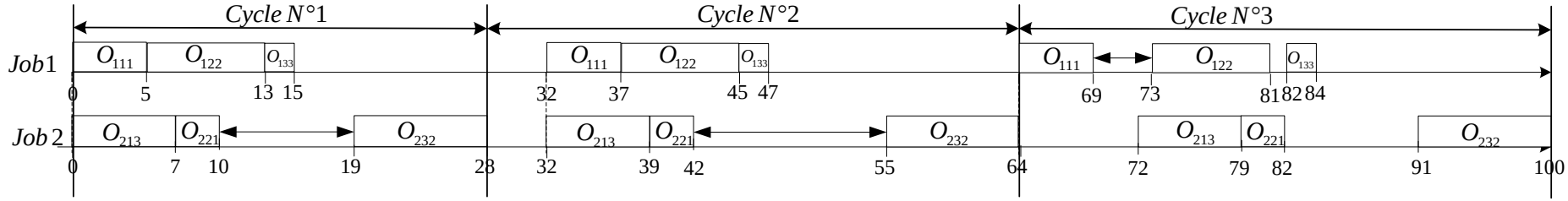


Figure III. 22 : Allocation des tâches en utilisant la nouvelle méthode de décalage

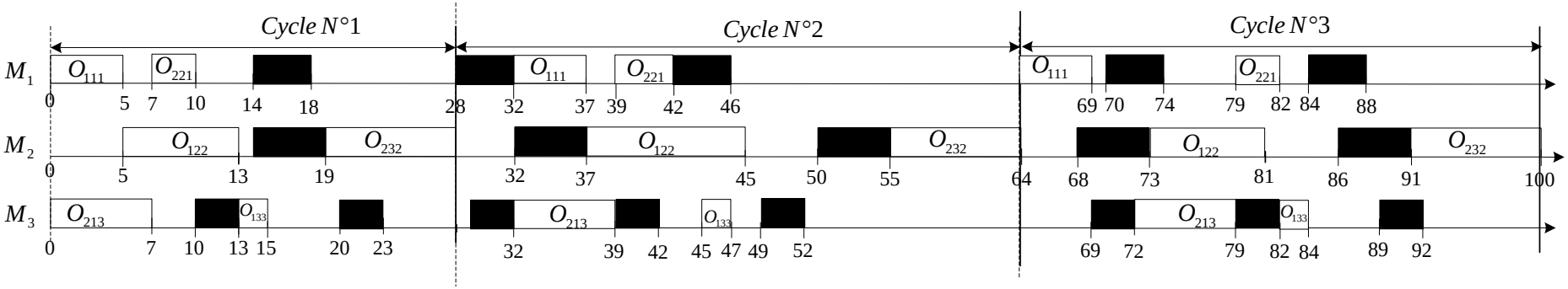


Figure III. 23 : Allocation des tâches en utilisant la nouvelle méthode de décalage

		1er cas : sans contraintes d'indisponibilité				2ème cas : avec contraintes d'indisponibilité (fixes)				3ème cas : avec contraintes d'indisponibilité			
		Cycle 1	Cycle 2	Cycle 3	Cycle moyen	Cycle 1	Cycle 2	Cycle 3	Cycle moyen	Cycle 1	Cycle 2	Cycle 3	Cycle moyen
C_{max}		22	44	66	-	32	68	104	-	28	64	100	-
Durée d'un cycle		22	22	22	34.66	32	36	36	34.66	28	36	36	33.33
Coût total		0	0	0	225.33	148	256	272	225.33	176	275	272	241
Facteur temps libre (%)	Job1	0	0	0	7.4	0	22.2	0	7.4	0	0	13.9	4.63
	Job2	13.63	13.63	13.63	39.45	40.62	44.4	33.34	39.45	32.14	36.11	25	31.08

Tableau III. 10 : Résultats de simulation obtenue pour le premier benchmark

2. Deuxième étape de simulation

Afin de tester davantage le nouvel algorithme de décalage pour la résolution du problème d'ordonnancement conjoint de la production et de la maintenance dans un atelier de type job shop job shop, un deuxième exemple de problème d'ordonnancement ft06 a été considéré. Le Tableau III. 11 présente les différentes données du benchmark ft06. De même, les données numériques de la maintenance des machines correspondantes au benchmark en question sont illustrées par le Tableau III. 12. Les données générées de maintenances pour le benchmark en question sont formées essentiellement par les périodes de maintenance systématique L_k, leurs durées correspondants d_k^l pour chaque machine M_k, le prix unitaire d'indisponibilité P_k, les marges d'avance $\Delta t_{k\max}^a$, les marges de retard $\Delta t_{k\max}^d$, le prix unitaire d'avance MC_k^a ainsi que le prix unitaire de retard MC_k^d.

Tâches	Machine (durée opératoire)					
	1	2	3	4	5	6
1	3 (1)	1(3)	2 (6)	4 (7)	6 (3)	5 (6)
2	2 (8)	3 (5)	5 (10)	1 (9)	1 (10)	4 (4)
3	3 (5)	4 (4)	6 (8)	4 (3)	2 (1)	5 (7)
4	2 (5)	1 (5)	3 (5)	6 (4)	5 (8)	6 (9)
5	3 (9)	2 (3)	5 (5)	6 (1)	1 (3)	4 (1)
6	2 (3)	4 (3)	6 (9)	1 (10)	5 (4)	3 (1)

Tableau III. 11 : Gamme opératoire (machines) et durée opératoire des tâches du problème (6×6)

Machine M_k	M_1	M_2	M_3	M_4	M_5	M_6
Période L_k	15	20	15	15	20	15
Durée d_k^l	4	5	3	4	5	3
Prix unitaire d'indisponibilité P_k	5	8	10	4	7	12
Marge d'avance $\Delta t_{k\max}^a$	2	4	1	2	4	1
Marge de retard $\Delta t_{k\max}^d$	3	4	1	3	4	1
Prix unitaire d'avance MC_k^a	9	12	8	6	7	9
Prix unitaire de retard MC_k^d	20	10	12	14	8	14

Tableau III. 12 : Données de la maintenance pour le deuxième benchmark

Le Tableau III. 13, présente un résumé sur les résultats de simulation du deuxième benchmark (Tableau III. 12).

		1ere expérience : avec des tâches de maintenances fixes						2ème expérience : avec des tâches de maintenance décalées					
		Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle5	Cycle moyen	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle moyen
C_{max}		98	203	338	443	578	-	113	205	306	401	501	-
Durée de cycle		98	105	135	105	135	115.6	113	92	101	95	100	100.2
Coût total		760	871	840	871	840	836.4	817	814	828	775	847	816.2
Temps libre (%)	Job 1	10.20	7.61	7.40	7.60	7.40	7.45	18.60	61.85	15.84	36.84	48	36.22
	Job 2	52.04	42.85	46.66	42.85	46.66	46.21	44.25	33.71	23.76	13.68	23	27.68
	Job 3	20.40	44.76	25.18	44.76	25.18	32.05	54.00	30.45	21.78	8.42	48	32.53
	Job 4	29.59	30.47	13.33	30.47	13.33	23.44	23	46.70	33.66	47.36	29	35.95
	Job 5	17.34	23.80	31.85	23.80	31.85	25.73	23	63.04	24.75	14.73	54	35.90
	Job 6	43.00	25.71	31.11	25.71	31.11	31.33	23	15.25	62.37	33.68	59	38.66

Tableau III. 13 : Résultats de simulation obtenue pour le deuxième benchmark

3. Interprétations des résultats

L'interprétation des résultats de simulation sera effectuée en deux étapes comme le cas des simulations.

Le Tableau III. 10, résume les résultats de simulation correspondant au premier exemple formé par deux tâches et trois machines. Nous avons pu dégager les constatations suivantes :

- Entre le 1er cas et le 2ème cas :

Il est évident que la présence des intervalles de maintenance a permis de retarder le Makespan C_{max} et par conséquent augmenter la durée du cycle. Concernant les coûts totaux correspondantes au 2ème cas, ils sont juste des coûts nécessaires aux interventions aux intervalles indisponibilités juste à temps (sans décalage).

De même, il est évident que l'intégration des intervalles d'indispensabilités dans un cycle de production augmente le facteur temps libre pour les différents jobs de la production (retarder l'exécution des opérations de production).

Donc notre intervention à ce niveau réside au niveau de l'amélioration de l'ordonnancement de la production en présence des tâches de maintenance (amélioration pour le deuxième cas).

- Entre le 2ème cas et le 3ème cas :

Il est clair qu'il y a une amélioration de la durée d'un cycle une fois nous avons appliqué notre nouvel algorithme de décalage pour affecter les tâches de production aux machines sans négliger leurs maintenances. Mais dans ces conditions nous ne pouvons pas nous échapper à des coûts supplémentaires de maintenance mais plutôt nous pouvons les réduire en appliquant les règles 4a et 4b.

De même l'utilisation de notre nouvel algorithme nous a permis aussi de réduire le facteur temps libre de chaque job, et ceci se traduit par la minimisation des lacunes entre les opérations du même job.

La deuxième partie de cette interprétation va porter sur les résultats donnés par le Tableau III. 13, qui résume les résultats de simulation correspondants au deuxième exemple formé par six tâches et six machines (fto6). A travers les résultats donnés par cet exemple, nous avons pu voir l'efficacité de notre nouvel algorithme ce qui n'a pas été très évident avec l'exemple précédent. Ainsi, nous avons pu dégager les constatations suivantes :

Pendant les cinq cycles de production, nous avons pu réduire conjointement la durée du cycle ainsi que le coût total de la maintenance.

Ainsi le fait d'avoir un bon compromis entre l'exécution des tâches de production et les tâches de maintenance préventive systématique ceci nous permet de garantir la qualité du produit et

aussi maitriser les coûts dépensés dans l'entreprise. En effet l'un des problèmes de la maintenance préventive est son coût élevé et pour remédier à cette anomalie, nous pouvons utiliser une planification adéquate qui peut tolérer quelques décalages avec respect à la fiabilité des machines.

5. Conclusions

La maintenance est liée à la production de plusieurs manières. La principale voie est de garantir la continuité de la production à travers la garantie de la fiabilité du matériel.

D'autre part et de nos jours, les exigences vis-à-vis de la maintenance deviennent de plus en plus élevées et ce surtout que les clients et les producteurs n'acceptent plus les pannes et la mauvaise qualité des produits. D'où et à travers ce chapitre nous avons essayé de résoudre le conflit relatif au problème d'ordonnancement des tâches de production tout en tenant compte des tâches de maintenances préventive systématique.

Dans ce chapitre, nous avons proposé un nouvel algorithme de décalage pour la résolution du problème d'ordonnancement conjoint de la production et de la maintenance dans un atelier job shop afin de minimiser conjointement deux critères : un critère lié à la production qui est le Makespan C_{max} et le second est le coût total de la maintenance par cycle de production ui est *CM*. Ce nouveau algorithme a été exposé sous forme de six règles à appliquer pour résoudre les conflits d'ordonnancement dans les files d'attente des tâches de production sans violer les contraintes de maintenances des machines.

Afin de mieux expliquer notre nouvel algorithme, nous avons exposé les étapes de simulation détaillé pas par pas sur un exemple de petite taille généré à partir d'un benchmark d'un problème d'ordonnancement job shop formé deux jobs et trois machines.

Pour mettre en évidence l'efficacité de notre algorithme, nous l'avons testé sur deux problèmes d'ordonnancement job shop pour lesquels nous avons généré des données de maintenance préventive systématique.

L'efficacité de la méthode proposée, a été atteinte par la réduction conjointement des deux critères de la production et de la maintenance sur des benchmarks d'illustration qui sont fournis. De même l'exécution du nouvel algorithme a permis aussi de fournir d'autres valeurs de paramètres tels que le temps libre de chaque machine et quelques autres statistiques liées au coût de la production et de la maintenance.

Conclusion générale

Le problème d'ordonnancement job shop qui traite l'allocation des tâches aux machines tout en respectant un certain nombre de contraintes à savoir les contraintes de séquencement et les contraintes de ressources afin de minimiser un critère lié à la production qui est le Makespan C_{max} a été présenté dans ce manuscrit de thèse.

Deux contributions principales ont été proposées à savoir,

a. Une méthode d'initialisation des dates de début des opérations a amélioré une métaheuristique déjà proposée dans la littérature par Willems *et al.* [WIL94] et [WIL95] pour la résolution du problème d'ordonnancement à cheminement multiple job shop. Par un réseau de neurone de type Hopfield. Cette méthode concerne uniquement les problèmes d'ordonnancement job shop sans tenir compte de l'indisponibilité des machines.
b. Une nouvelle méthode de décalage des opérations de production et de maintenance afin de résoudre le problème d'ordonnancement conjoint de la production et de la maintenance afin d'optimiser deux critères l'un lié à la production le Makespan C_{max} et l'autre lié à la maintenance c'est le coût total de la maintenance *CM*. Cette méthode se résume dans un ensemble de règles.

Il est à noter que malgré les résultats intéressants donnés par ces deux méthodes, un certain nombre d'inconvénients doivent être notés :

- La méthode d'initialisation du réseau de neurone de type Hopfield pour la résolution du problème d'ordonnancement job shop donne de moins bon résultats pour les problèmes de grandes tailles supérieures à ft06.
- La méthode de décalage des opérations de production et de maintenance a été appliquée uniquement en considérant les périodes de maintenance préventives systématiques planifiées à l'avance. Cependant, en réalité, des pannes aléatoires peuvent surgir et des interventions de maintenance corrective doivent s'effectuer. La méthode n'est pas adaptée pour ces cas.

Comme perspective :

Dans l'objectif d'améliorer l'efficacité de la méthode de décalage, nous proposons de l'étendre en considérant des tâches de maintenances correctives qui peuvent paraitre aléatoirement lors de l'ordonnancement des tâches de production et de la maintenance systématique.

Bibliographie

[ADA88] J. Adams, E. Balas, and D. Zawack, “The shifting bottleneck procedure for job shop scheduling,”*Management Science*, Vol. 34, n°3, pp. 391-401, 1988.

[ADI10] M .A. Adibi, M. Zandieh, and M. Amiri, “Multi-objective scheduling of dynamic job shop using variable neighborhood search,” *Expert Systems with Applications*, Vol. 37, pp. 282-287, 2010.

[AFNOR] AFNOR, Recueil de normes françaises X 06, X 50, X 60, AFNOR, AFNOR X 60-010-1994

[AKE56] S. B. Akers, “A graphical approach to production scheduling problems,” *Operations Research*, Vol. 4, pp. 244-245, 1956.

[AKY05] D. E. Akyol, and G. M. Bayhan, “A coupled gradient network approach for the multi machine earliness and tardiness scheduling problem,” In O. Gervasi, M. L. Gavrilova, V. Kumar, A. Lagana, H. P. Lee, & Y.Mun, et al. (Eds.). *Proceedings of ICCSA*, Vol. 3483. Lecture notes in computer science (pp. 596–605). Berlin: Springer, 2005.

[AKY07] D. E. Akyol and G. M. Bayhan, “A review on evolution of production scheduling with neural networks,” *Computers & Industrial Engineering*, Vol. 53, pp. 95-122, 2007.

[ART08] C. Artigues, and D. Feillet, “A branch and bound method for the job-shop problem with sequence-dependent setup times,” *Annals of Operations Research*, Vol. 159, pp. 135-159, 2008.

[ASA10] L. Asadzadeh, and K. Zamanifar, “An agent-based parallel approach for the job shop scheduling problem with genetic algorithms,” *Mathematical and Computer Modelling*, Vol. 52, pp.1957-1965, 2010.

[AUB04] J. M. Auberville, *Maintenance industrielle - De l'entretien de base à l'optimisation de la sûreté*, ISBN : 2-7298-2011-6, collection technosup, Ellipses édition Marketing S. A., 32 rue Bargue 75740 Paris cedex 15, 2004.

[AYT05] H. Aytug, M. A. Lawley, K. Mckay, S. Mohan, and R. Uzsoy, “Executing production schedules in the face of uncertainties: A review and some future directions,” *European Journal of Operational Research*, Vol. 161, pp. 86–110, 2005.

[BAK09] K. R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*, John Wiley & Sons 2009.

[BAK74] K. R. Baker, *Introduction to Sequence and Scheduling*, New York: Wiley, 1974.

[BAL67] E. Balas, “Discrete programming by the filter method,” *Operations Research*, Vol. 15, pp. 915-957, 1967.

[BAL69] E. Balas, “Machine sequencing via disjunctive graphs: an implicit enumeration algorithm,” *Operation research*, Vol. 17, pp.941-957, 1969.

[BAL70] E. Balas, “Machine sequencing: disjunctive graphs and degree-constrained sub graphs,” *Naval Research Logistics Quarterly*, Vol. 17, pp. 941-957, 1970.

[BAR99] J. F. Bard, *Practical Bilevel Optimization: Algorithms and Applications*, volume 30 of Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, 1999.

[BEA98] J. E. Beasley, OR-library: distributing test problems by electronic mail: The *Journal of the Operational Research Society* ISSN 0160-5682 CODEN JORSDZ , 1990, vol. 41, no11, pp. 1069-1072 (8 ref.)

[BEL74] R. Belmann, and S. E. Dreyfus, *Applied dynamic programming*, Princeton University Press, 1974.

[BEN06] F. Benbouzid-Sitayeb, Résolution du problème de l’ordonnancement conjoint production/maintenance par colonies de fourmis, 6e Conférence Francophone de MOdélisation et SIMulation - MOSIM’06 Rabat (Maroc), 2006.

[BEN09] G. Bencheikh, J. Boukachour, A. El Hilali Alaoui, and F. El Khoukhi “Hybrid method for aircraft landing scheduling based on a Job Shop formulation,” *IJCSNS International Journal of Computer Science and Network Security*, Vol.9 n°.8, August 2009.

[BEN10] F. Benbouzid-Sitayeb , A. Bendjoudi , S. Benkhellat , C. Varnier, and N. Zerhouni, “A study of maintenance contribution to joint production and preventive maintenance scheduling problems in the robustness framework,” *International Journal of Product Development*, Vol. 10, Number 1-3, pp. 144-164, 2010.

[BEN65] J. Bentz, "Aperçu sur les problèmes d’ordonnancement, Mathématiques et sciences humaines,"tome 13, pp. 3-21, 1965.

[BER10] A. Berrichi, F. Yalaoui, L. Amodeo, M. Mezghiche, “Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling,” *Computers & Operations Research*, Vol. 37, Issue 9, pp. 1584-1596, 2010.

[BLA07] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, "*Handbook on scheduling* from theory to application," a book chapter :10 Scheduling in Job Shops, *International Handbooks Information System*, pp. 345-396, 2007.

[BLA96] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Scheduling in computer and manufacturing processes*, Springer Verlag, 1996.

[BLA96] J. Blazewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, Vol. 93, pp.1-33, 1996.

[BOU91] D. Boucon, Ordonnancement d'ateliers: aide au choix de règles de priorité. Thèse de doctorat, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, 1991.

[BOW59] E. H. Bowman, "The schedule-sequencing problem," *Operation research*, Vol. 7, n°.5, pp. 621-624, September-October 1959.

[BRU07] P. Brucker, "The job-shop problem: Old and new challenges," *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling*: Theory and Applications (MISTA), Paris, France (28-31 August 2007), pp. 15-22, 2007.

[CAR89] J. Carlier, E. Pinson, "A Branch and Bound Method for Solving the Job Shop Problem", *Management Science*, Vol. 35, No. 2, pp. 164-176, february 1989.

[CAS05] C. R. Cassady and K. C. Erhan, "Integrating Preventive Maintenance Planning and Production Scheduling for a Single Machine," *IEEE Transactions on Reliability*, Vol. 54, n°. 2, June 2005.

[CAS96] Y. Caseau, and F. Laburthe, "Improving branch and bound for Job shop scheduling with constraint propagation," *Combinatorics and Computer Science*, Lecture Notes in Computer Science, Vol. 1120, pp. 129-149, 1996.

[CER88] A. Cernault, *La simulation des systèmes de production*, CEPADUES, Toulouse, 1988.

[CHE01] R. M. Chen, and Y. M. Huang, "Competitive neural network to solve scheduling problems," *Neurocomputing*, Vol. 37, pp. 177-196, 2001.

[CHE06] A. Chelbi, and N. Rezg, "Analysis of a production/inventory system with randomly failing production unit subjected to a minimum required availability level," *International Journal of Production Economics*, Vol. 99, pp.131–143, 2006.

[CHE96] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part I: representation," *International Journal of Computers and Industrial Engineering*, Vol.30, n° 4, pp. 983-997, 1996.

[CHE98] H. Chen, C. Chu, and J. M. Proth, "An Improvement of the Lagrangean Relaxation Approach for Job Shop Scheduling: A Dynamic Programming Method," *IEEE Transaction on Robotics and Automation*, Vol. 14, No. 5, pp. 786-795, 1998.

[CHE99] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part I: representation," *Computers and Industrial Engineering*, Vol. 36, pp. 343-364, 1999.

[CHI10] S.W. Chiu, "Robust planning in optimization for production system subject to random machine breakdown and failure in rework," *Computers & Operations Research*, Vol. 37, pp. 899-908, 2010.

[CLA99] J. Clausen, "Branch and Bound Algorithms - Principles and Examples," department of comp sc., university of Copenhagen, Universitetsparken 1, DK-2100 Corpenhagen, Denmark. (March 12, 1999), http://www.imm.dtu.dk/_jha/

[CON67] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Reading, MA: Addison-Wesley, 1967.

[CRA03] Y. Crama., *Eléments de gestion de la production, Notes de cours*, Ecole d'Administration des Affaires. Université de Liège, Année académique 2002-2003.

[DAV91] L. Davis, *Handbook of Genetic Algorithms*, Editions Van Nostrand Reinhold, N.Y, 1991.

[DEN82] E.V. Denardo, *Dynamic Programming: Models and Applications*, Prentice-Hall, NJ, 1982.

[DHI02] B. S. Dhillon, *Engineering Maintenance: A Modern Approach*, ISBN: 1587161427, CRC Publication, 2002.

[DOM04] P. D. D. Dominic, S. Kaliyamoorthy, and M. S. Kumar, "Efficient dispatching rules for dynamic job shop scheduling," *The International Journal of Advanced Manufacturing Technology*, Vol. 24, n°1-2, pp. 70-75, 2004.

[DOR95] U. Dorndorf, and E. Pesch, "Evolution Based Learning in a Job-Shop Scheduling Environment," *Computers and Operations Research*, Jan, Vol. 22, n°1, pp. 25-40, 1995.

[DRE02] G. Dreyfus , J. -M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria, and L. Hérault, *Réseaux de neurones :Méthodologie et applications*, Eyrolles, 2002.

[ESQ99] P. Esquirol & P. Lopez, *L'ordonnancement*, Edition Economica, Paris, ISBN 2-7178-3798-1199.

[FON02] D. J. Fonseca, and D. Navaresse, "Artificial neural networks for job shop simulation" *Advanced Engineering Informatics*, Vol. 16, pp. 241-246, 2002.

[FOO 88] Y. P. S. Foo, and Y. Takejuji, "Integer Linear Programming Neural Networks for Job-Shop Scheduling," in Proc. *IEEE IJCNN'88*, pp. 341-348, 1988.

[FOO88a] S. Y-P. Foo, and Y Takefuji, "Stochastic neural networks for solving job-shop scheduling: Parts I. Problem representation," *In Proceedings of the IEEE International Conference on Neural Networks*, Vol. 11, pp. 275-282, 1988.

[FOO88b] S. Y-P. Foo, and Y Takefuji, "Stochastic neural networks for solving job-shop scheduling: Part II, architecture and simulations," *In Proceedings of joint international conference on neural networks*, Vol. 2, pp. 283-290, 1988.

[GAR00] A. Garrido, M.A. Salido, F. Barber, and M.A. López, "Heuristic methods for solving job-shop scheduling problems. Technical report," *Universidad Politécnica Universidad Politécnica de Valencia*, 2000. http://www-is.informatik.uni-oldenburg.de/~sauer/puk2000/papers/Garrido_Job-Shop.pdf

[GAR79] R.Gary and D. S. Johnson: Computers and Intractability: A guide to the theory of NP-Completeness. Co. 1997.

[GEN97] L. Geneste, and B. Grabot, (1997). "Implicit versus explicit knowledge representation in a job shop scheduling decision support system," *International Journal of Expert Systems*, Vol. 10, n°.1, pp. 37-52, 1997.

[GEO08] V. Geoffrey, and B. Jean-Charles, A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem," *European journal of operational research*, Vol. 190, n°. 2, pp. 398-411, 2008.

[GHE06] K. Ghédira, *Logistique de la production: Approches de modélisation et de résolution*, Editions TECHNIP, Collection Sciences et Technologies, 2006.

[GLO86] F. Glover, "Future paths for integer programming and links to artificial intelligence," Computer and Operations Research, Vol. 13, n°.5, pp. 533-549, 1986.

[GLO89] F. Glover, "Tabu search: Part I", *ORSA Journal on Computing*, Vol. 1, n°.3 , pp. 190-206, 1989.

[GOL89] D. E. Goldberg, *Genetic Algorithms in Search Optimisation and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

[GRA79] R. L. Graham, E. L. Lawler, J. K. Lenstra., A. H. G. Rinnooy Kan, "Optimisation and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, Vol. 5, pp. 287-326, 1979.

[HAO99] J.K. Hao, "Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes," *Revue d'Intelligence Artificielle*, Vol. 13, n°2, pp. 283-324, 1999.

[HAN86] P. Hansen, "The steepest ascent mildest descent heuristic for combinatorial optimisation," Congress on Numerical Methods in Combinatorial Optimisation, Capri, Italy, 1986.

[HAR03a] Y. Harrath, B. Chebel-Morello, and N. Zerhouni, "Lower Bounds and Multiobjective Evolutionary Optimization for Combined Maintenance and Production Scheduling in Job Shop," *IEEE Conference on Emerging Technologies and Factory Automation*, 2003.

[HAR03b] Y. Harrath, Contribution à l'ordonnancement conjoint de la production et de la maintenance: Application au cas d'un job shop, Thèse de doctorat soutenue à l'U.F.R. des Sciences et Techniques de L'Université de Franche-Comté, 2003.

[HAR63] W. W. Hardgrave and G. L. Nemhauser, "A geometric model and graphical algorithm for a sequencing problem," *Operations research*, Vol. 12, n. 2, 1963.

[HAS09] S. M. K. Hasan, R. Sarker, and D. E. David Cornforth "Memetic algorithms for solving job-shop scheduling problems," *Memetic Computing*, Vol.1, n°.1, pp. 69–83, 2009.

[HAU04] R. L. Haupt, and S. E. Haupt, *Practical Genetic Algorithms*, second edition, Wiley/Interscience, New York, 2004.

[HAU89] R. Haupt, "A survey of priority rule-based Scheduling," *OR Spectrum*, *Springer-Verlag*, Vol. 11, pp.3-16, 1989.

[HAY99] S. Haykin, *Neural Networks – A Comprehensive Foundation*, 2nd Edition, Prentice Hall, 1999.

[HOL75] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.

[HOP82] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of National Academy of Sciences*, Vol.79, pp.2554-2558, 1982.

[HOP85] J. J. Hopfield, and D. W. Tank, "Neural computation of decision in optimization problems," *Biological Cybernetics*, Vol. 52, pp. 141-152, 1985.

[JAC56] J. R. Jackson, "An extension of Johnson's results on job-lot scheduling," *Naval Research Logistics Quarterly*, vol. 3, n°3, 1956.

[JAI98a] A. S. Jain, and S. Meeran, "Job shop scheduling using neural networks," *International Journal of Production Research*, Vol. 36, n°.5, pp.1249-1272, 1998.

[JAI98b] A. S. Jain, S. Meeran, "A state-of-the-art review of job-shop scheduling techniques," Technical report, Department of Applied Physics, *Electronic and Mechanical Engineering*, University of Dundee, Dundee, Scotland, 1998.

[JAI99] A. S. Jain & S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operational Research*, Vol.113, pp. 390-434, 1999.

[JEF06] H. W. Jeffrey, *Handbook of production scheduling*, University of Maryland, College Park. 2006.

[JOH56] S. M. Johnson, "Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included," *Naval Research Logistics Quarterly*, Vol 1, pp. 61-68, 1956.

[JON98] A. Jones, and L.C. Rabelo, "Survey of job shop scheduling techniques," Technical Paper, NISTIR, National Institute of Standards and Technology, Gaithersburg, MD, 1998. (Téléchargeable du site Web http://www.mel.nist.gov/msidlibrary/doc/luis.pdf).

[KAM10] S. M. Kamrul Hasan, R. Sarker, and D. Essam, "Genetic Algorithm for Job-Shop Scheduling with Machine Unavailability and Breakdown," *International Journal of Production Research*, DOI: 10.1080/00207543.2010.495088, 2010.

[KAR08] B. Kareem and A. A. Aderoba, "A Cost Estimation Model of Maintenance Job shop Operations," AU J.T. 12(1): pp. 25-33, Jul. 2008.

[KÄS99] J. Käschel, T. Teich, G. Köbernik, and B. Meier, "Algorithms for the job-shop scheduling problem: A comparison of different methods," *In European symposium on intelligent techniques*, Greece, pp.3-4, 1999.

[KIN06] V. T_Kindt, and J. C. Billaut, *Multicriteria Scheduling: Theory, Models and Algorithms*, Springer Verlag, 2006, ISBN-10 3-540-28230-0 2nd ed. Springer Berlin Heidelberg New York, 2006.

[KIR83] S. Kirkpatrick, C. D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science, New Series*, Vol. 220, n°. 4598, pp. 671–680, 1983.

[KUB02] W. Kubiak, J. Blazewicz, P. Formanowicz, J. Breit, G. Schmidt, 2002. "Two-machine flow shops with limited machine availability," *European Journal of Operational Research*, Vol. 136, pp. 528–540,2002.

[LEE09] Y.F. Lee, Z.B. Jiang, H.R. Liu, "Multiple-objective scheduling and real-time dispatching for the semiconductor manufacturing system," *Computers and Operations Research*, Vol. 36, pp. 866-884, 2009.

[LAN60] A. H. Land et A. G. Doig, "An automatic method for solving discrete programming problems", Econometrica, Vol. 28, pp. 497-520, 1960.

[LAW66] L. E. Lawler, and D. E. Wood, "A branch-and-bound methods: a survey," *Operations research*, Vol. 14, pp. 699-719, 1966.

[LIA92] C. J. Liao, and C. T. You, "An improved formulation for the job-shop scheduling problem," *Journal of Operation Research Society*, Vol. 43, no.11, 1992.

[LO93] Z. P. Lo, and B. Bavarian, "Multiple job scheduling with artificial neural networks," *Computers and Electrical Engineering*, Vol. 19, n°2, pp. 87-101, 1993.

[LOP01] P. Lopez, et F. Roubellat, *Ordonnancement de la production*, Edition Hermès Science, 2001.

[LOW10] C. Low, M. Ji, C-J. Hsu, and C.T. Su, "Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance," *Applied Mathematical Modeling*, Vol. 34, pp. 334–342, 2010.

[MAN60] S. Manne, "On the job shop scheduling problem," *Operation research*, Vol. 8, pp. 219-223, 1960.

[MAS97] M. S. Marseille, and J. B. Lapointe, *La gestion des équipements vers l'entretien préventif*, ISBN : 2-921360-01-3, Association paritaire pour la santé et la sécurité du travail-Secteur fabrication de produits en métal et de produits électriques. 2271, rue Fernand- Lafontaine, bur.301. Longueuil (Québec) J4G 2R7, 1997.

[MCC43] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity. "*Bulletin of Mathematical Biophysics*, Vol. 7, pp. 115-133,1943.

[MCM75] G. B. McMahonand and M. Florian, "On scheduling with ready times and due dates to minimize maximum lateness," *Operations Research*, Vol. 23, n°. 3, pp. 475-482, May-June 1975.

[MET53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equations of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, Vol. 21, n°6, pp.1087–1092, 1953.

[MON06] J. Montgomery, C. Fayad, S. Petrovic, Solution representation for job shop scheduling

problems in ant colony optimization, Faculty of Information & Communication Technologies, Swinburne University of Technology, 2006.

[MUT63] J. F. Muth, and G. L. Thompson, *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, N. J, 1963.

[NAK91] R. Nakano, and T. Yamada, "Conventional genetic algorithm for job shop problems," in: R.K. Belew, L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 474-479, 1991.

[NIC02] N. Terrier, La maintenance [archive] - Projet universitaire de, DESS QUASSI 2001-2002, Université d'Angers [pdf]. http://www.univ-angers.fr/docs/etudquassi/maintenance.pdf.

[OMB04] B. Ombuki and M. Ventresca, Local search genetic algorithms for the job shop scheduling problem, Applied Intelligence, Vol. 21, pp. 99-109, 2004.

[PEZ00] F. Pezzella, and E. Merelli, "A tabu search method guided by shifting bottleneck for the job shop scheduling problem," *European Journal of Operational Research*, Vol. 120, pp. 297-310, 2000.

[PHA08] D-N. Pham, and A. Klinkert, "Surgical case scheduling as a generalized job shop scheduling problem," *European Journal of Operational Research,* Vol.185, pp.1011–1025, 2008.

[PHI94] P. R. Philipoom, L. R. Rees, and L. Wiegmann, "Using neural networks to determine internally-set due date assignments for shop scheduling," *Decision Sciences*, Vol. 25(5/6), pp. 825-851, 1994.

[PIN05] M. L. Pinedo, *Planning and Scheduling in Manufacturing and Services,* ISBN 0-387-22198-0, Springer, New York, 2005.

[PIN08] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems, Third Edition*, Springer Science+Business Media, LLC. ISBN: ISBN: 0387789340, 2008.

[PRO07] J. M. Proth, "Scheduling: New trends in industrial environment Annual Reviews in Control", Vol. 31, pp. 157-166, 2007.

[ROG92] P. Roger, *Gestion de production*, Edition Dalloz-Sirey, 1992.

[RON08] R. L. Rondon, A. S. Carvalho, and G.I. Hernández, "Neural network modelling and simulation of the scheduling," book chapter, Innovation in Manufacturing Networks, in IFIP *International Federation for Information Processing* PY-PB, *Innovation in Manufacturing Networks; ed. A. Azevedo*; (Boston: Springer), Vol. 266, pp.231-238, 2008.

[ROY64] B. Roy et B. Sussmann, *Les Problèmes d'Ordonnancement avec Contraintes Disjonctives*, Technical Report Note DS no. 9bis, SEMA, Paris, 1964.

[SAT94] T. Satake, K. Morikawa, and N. Nakamura, "Neural network approach for minimizing the makespan of the general job-shop," *International Journal of Production Economics*, Vol.33, pp. 67-74, 1994.

[SMI34] S. A. Smith, "Service reliability measured by probabilities of outage," *Electrical World*, Vol. 103, pp. 371–374, 1934.

[SON09] D. P. Song, "Production and preventive maintenance control in a stochastic manufacturing system," *International Journal of Production Economics*, Vol. 119, pp. 101-111. 2009.

[SRI71] V. Srinivasan, "A hybrid algorithm for the one machine sequencing problem to minimize total tardiness," *Naval Research Logistics Quarterly*, Vol. 18, pp. 317-327, 1971.

[SUG10] B. Sugato, G. Soumyadip, and K. Ayant R. "Integration of job shop scheduling with discreet event simulation for manufacturing facilities," United States International Business Machines, n°.7702411, 2010.

[SUN10] K. Sun, and H. Li, "Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines," *International Journal of Production Economics*, Vol. 124, n. 1, pp. 151-158, 2010.

[TAN07] F. Tangour., Ordonnancement Dynamique dans les Industries Agroalimentaires, Thèse de doctorat soutenue à L'École Centrale de Lille, 2007.

[THA09] R. Thamilselvan, and Dr. P. Balasubramanie "Integrating Genetic Algorithm, Tabu Search Approach for Job Shop Scheduling," *International Journal of Computer Science and Information Security*, Vol. 2, n°.1, 2009.

[VAN92] P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra "Job Shop Scheduling by Simulated Annealing," *Operations research,* Vol. 40, n°.1, pp. 113-125, 1992.

[VAN91] M. M. Van Hulle, "A goal programming network for mixed integer linear programming: A case study for the job shop scheduling problem," *International Journal of Neural Systems*, Vol. 2, n. 3, pp. 201-209, 1991.

[VEL07] P. S. Velmurugan, Z. V. Selladurai, "A Tabu Search Algorithm for Job Shop Scheduling Problem with Industrial Scheduling Case Study," *International Journal of Soft Computing, Medwell © Medwell Journals,* Vol. 2, n°4, pp. 531-537, 2007.

[WAG59] H. M. Wagner, "An integer linear programming model for machine sequencing," *Naval Research, Logistic Quarterly*, Vol.6, issue. 2, pp. 131-140, Juin 1959.

[WAT03] J-P. Watson, J. C. Beck, A. E. Howe, and L. D. Whitley, "Problem difficulty for tabu search in job-shop scheduling," *Artificial Intelligence,* Vol. 143, pp.189-217, 2003.

[WEN04] H. Wenqi, and Y. Aihua, "An improved shifting bottleneck procedure for the job shop scheduling problem," *Computers & Operations Research,* Vol. 31, pp. 2093-2110, 2004.

[WIKIPEDIA] http://fr.wikipedia.org/wiki/Algorithmique#D.C3.A9finition

[WIL93] T. M. Willems, and J. E. Rod "Neural nets for job-shop scheduling will they do the job?" *in Proc. IFAC 12th World Conference*, III, pp. 53-56, 1993.

[WIL94] T. M. Willems, and J.E. Roda, "Neural networks for job-shop scheduling," *Control Engineering Practice,* Vol. 2, n°.1, pp. 31-39, 1994.

[WIL95] T. M. Willems, and L. E. M. W. Brandts, "Implementing heuristics as an optimization criterion in neural networks for job-shop scheduling," *Journal of Intelligent Manufacturing*, Vol. 6, pp. 377-387, 1995.

[YAH06] A. Yahyaoui, and F. Fnaiech, "Recent Trends in Intelligent Job Shop Scheduling," *IEEE ICELIE*'2006 First *International Conference on E-Learning in Industrial Electronics*, Yasmine Hammamet (Tunisia), December, 18-20, pp. 191-195, 2006.

[YAH09] A. Yahyaoui, N. Fnaiech, and F. Fnaiech, "New Shifting Method for Job Shop Scheduling Subject to Invariant Constraints of Resources Availability," *Industrial Electronics, IECON '09. 35th Annual Conference of IEEE*, pp. 3387-3392, 2009.

[YAH11a] A. Yahyaoui, N. Fnaiech, and F. Fnaiech, "A Suitable Initialization Procedure for Speeding a Neural Network Job-Shop Scheduling," *IEEE Transactions on Industrial Electronics*, Vol. 58, Issue. 3, pp. 1052-1060, March 2011.

[YAH11b] A. Yahyaoui, N. Fnaiech, and F. Fnaiech, "New shifting method for combined production scheduling and maintenance cost in job shop," *submitted to the International Review on Modelling and Simulations (I.RE.MO.S.),* 2011.

[YAM03] T. Yamada, Studies on Metaheuristics for Job shop and Flow shop Scheduling Problems, Ph. D Thesis, Kyoto University, Kyoto, Japan, November 2003.

[YAM95] T. Yamada and R. Nakano, "Job shop scheduling by simulated annealing combined with deterministic local search," *Metaheuristics International Conference*. Hilton, Breckenridge, Colorado, USA, pp. 344-349, 1995.

[YON07] C.Y. Zhang, P. Li, Z. Guan, and Y. Rao, "A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem," *Computers and Operations Research*, Vol. 34 , Issue 11, pp. 3229-3242, 2007.
http://archive.numdam.org/ARCHIVE/MSH/MSH_1965__13_/MSH_1965__13__3_0/MSH_1965__13__3_0.pdf
http://dsie.fe.up.pt/site/docs/pedroabreu/optimizacao-20090204-en.pdf,
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html
In: Männer, R., Manderick, B. (Eds.), *Proceedings of the Second European Conference on Parallel Problem Solving from Nature*. Elsevier, Amsterdam, pp. 281–290, 1992.

[ZHA05] F. Zhao, Y. Hong, and D. Yu, "A Hybrid Approach Based on Artificial Neural Network and Genetic Algorithm for Job-shop Scheduling Problem," *International Conference on Neural Networks and Brain, ICNN&B '05*, pp. 1687-1692, 2005.

[ZHA95] C.S. Zhang, and P.F. Yan, "Neural network method of solving job-shop scheduling problem," *ACTA Automation Sinica*, Vol. 21, pp. 706-712, 1995.

[ZHO91] D.N. Zhou, V. Cherkassky, T.R. Baldwin, and D.E. Olson, "A neural network approach to job-shop scheduling," *In Proceedings of the IEEE International Conference on Neural Networks*, Vol.2, Issue 1, pp. 341-348, 1991.

[ZIM04] W. B. J. Zimmerman, Process "*Modelling and Simulation with Finite Element Methods,"* SINGAPORE, World Scientific Publishing Co. Pte. Ltd. series on stability, vibration and control of systems, 2004. ISBN 981-238-793-5.

[ZOB08] G.I. Zobolas, C.D. Tarantilis, and G. Ioannou, "Exact Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems," *Studies in Computational Intelligence*, Vol. 28, pp.1-40, 2008.

Liste des publications de la candidate

1. Yahyaoui, F. Fnaiech “Recent Trends in Intelligent Job Shop Scheduling “publié à la conférence ICELIE’2006 (IEEE) First International, *Conference on E-Learning in Industrial Electronics* á Yasmine-Hammamet (Tunisia) December 18-20, 2006.
2. A. Yahyaoui, N. Fnaiech, and F. Fnaiech, “New Shifting Method for Job Shop Scheduling Subject to Invariant Constraints of Resources Availability,” *IEEE, IECO*N09, 2009.
3. A. Yahyaoui, N. Fnaiech, and F. Fnaiech “A Suitable Initialization Procedure for Speeding a Neural Network Job-Shop Scheduling”, *IEEE Transactions on Industrial Electronics*, Vol. 58, n°. 3, pp. 1052-1060, March 2011. (Journal indexé THOMSON REUTER Facteur d’impact: 4.67)
4. N. Fnaiech, A. Yahyaoui and F. Fnaiech “New shifting method for combined production scheduling and maintenance cost in job shop” soumis au journal: *The International Review on Modelling and Simulations (I.RE.MO.S.)*, 2011.

Publications dans le cadre du DEA

5. A. Yahyaoui, M. Sassi, " Modélisation et simulation d’un système de préhension robotisée " « JTEA 2002 »)