**R**épublique **A**lgérienne **D**émocratique et **P**opulaire
**M**inistère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique
**U**niversité **M**'hamed **B**OUGARA – **B**oumerdès


Faculté des Sciences
Département d'Informatique

**Domaine** : Mathématiques-Informatique
**Filière** : Informatique
**Spécialité** : Ingénierie du Logiciel et Traitement de l'Information

***Mémoire de fin d'études en vue de l'obtention du Diplôme de Master en Informatique***

# *Thème*

# *Optimisation bi-objectif pour un ordonnancement intégré des tâches de production et de maintenance*

***Présenté par :***
AMAZOUZ Sofiane
DJEDDAI Mohamed El-Mahdi

***Soutenu le 21/06/2016 Devant le jury composé de***

| | | |
|---|---|---|
| Pr. BOULIF Menouar | : | Président |
| Mr. MERHOUM Kheireddine | : | Examinateur |
| Dr. BERRICHI Ali | : | Encadreur |

*A tous ceux que nous aimons…*

*Sofiane et Mido*

.

# *Remerciements*

*Nous souhaitons remercier les membres du Département d'Informatique de l'Université M'hamed Bougara- Boumerdes ainsi que l'ensemble des enseignants ayant contribué à notre formation.*

*Un grand merci aux membres du jury, le Pr BOULIF Menouar et Mr MERHOUM Kheireddine, qui ont bien voulu évaluer notre mémoire de Master en Ingénierie du Logiciel et Traitement de l'Information.*

*Nous sommes très reconnaissants envers notre encadreur, le Dr BERRICHI Ali, pour le temps qu'il nous a consacré ainsi que pour ses précieux conseils.*

*Bien sûr, nous n'oublions pas de remercier nos familles et nos amis respectifs pour leur présence et leur soutien inconditionnel.*

# Table des matières

# Table des figures

# Liste des tableaux

# Table des algorithmes

# Introduction générale

L'environnement économique a toujours été caractérisé par une très forte concurrence et une offre très importante. Dans ce marché concurrentiel -où la clientèle est très exigeante- est apparue la nécessité de concevoir et mettre en place des systèmes de production performants, pouvant s'adapter aux changements du marché et aux perturbations internes (panne des machines, manque de main d'œuvre) et externes (demande).

Pour s'assurer une place dans cet environnement économique, l'entreprise se doit de maintenir un niveau concurrentiel en respectant les délais de livraison et assurant un produit de très bonne qualité. Dans ce sens, l'entreprise se doit de maintenir ses outils de production fiables et en bon état de fonctionnement, ce qui est la préoccupation majeure des industriels.

Parmi les fonctions de l'entreprise, nous citons la production et la maintenance. Elles contribuent à la compétitivité des entreprises industrielles. Malgré leur interdépendance les deux fonctions sont planifiées et exécutées séparément. Cela dit, elles devraient collaborer afin d'atteindre un niveau de productivité élevé.

Ainsi, il faut déterminer simultanément le meilleur planning de production et une politique de maintenance appropriée, le tout en optimisant des critères bénéfiques à l'entreprise (le coût, etc.) et à son système de production (durée de vie des machines). Donc, comment assurer le bon fonctionnement du système de production ? En respectant les délais de livraison et en améliorant la disponibilité des ateliers et machines. Le tout en ordonnançant de manière intelligente et concise les tâches de production et en effectuant des opérations de maintenance.

Notre objectif est de proposer une modélisation du problème conjoint production/maintenance, dans le cas d'ateliers de type machine parallèle identiques.

Le présent document est divisé en cinq parties : Dans le premier chapitre, nous aborderons les systèmes de production, les notions relatives à l'ordonnancement ainsi que la maintenance. Ensuite, dans le second, nous présenterons un état de l'art sur les problèmes d'ordonnancement conjoints production/maintenance, le problème à traiter et sa modélisation puis nous introduirons l'optimisation multi-objectif. Le troisième chapitre sera un état de l'art sur les méthodes utilisées pour résoudre un problème multi-objectif dans lequel nous présenterons les deux algorithmes que nous allons développer pour implémenter notre solveur. Il s'en suivra le chapitre IV qui sera consacré à l'implémentation et l'adaptation des deux techniques présentées dans le chapitre précédant, notre application ainsi qu'à l'élaboration des tests expérimentaux

comparant les deux méthodes. Nous finirons notre travail par une conclusion générale suivie de perspectives résultant du travail effectué.

# Chapitre I: Gestion et maintenance des systèmes de production

# Introduction

Le but de ce chapitre est de présenter le contexte de notre étude. Il se compose de trois sections. Dans la première section, nous aborderons la gestion de production, ses objectifs ainsi que la typologie des systèmes de production. La seconde section exposera les caractéristiques des problèmes d'ordonnancement et leur classification. La troisième section présentera les différentes politiques de maintenance et sa politique. Enfin nous terminons le chapitre par une conclusion.

# 1. Systèmes de production

## 1.1. Définition

Produire, c'est transformer. Le lieu et les moyens de ces transformations, c'est le système de production (SP) (SASSINE, 1998). En d'autres termes, le SP représente l'ensemble du processus et des moyens de production que l'entreprise met en place afin de transformer des matières premières ou des produits semi finis, en produits finis prêts à être commercialisés.

## 1.2. Objectifs associés

L'objectif principal des Systèmes de Production est de produire un bien économique. Cependant, la fonction de production se doit de satisfaire d'autres objectifs intermédiaires que sont (SASSINE, 1998) :

***En termes de quantités produites ;*** Il faut que la fonction de production permette de satisfaire la demande des clients. Pour pouvoir la réaliser, l'entreprise doit mener des actions pour maintenir les capacités productives ou bien mettre au point des plans d'investissement en capacité.

***En termes de qualité ;*** Afin de satisfaire les besoins de la clientèle et s'assurer un niveau de compétitivité, il faudrait que les biens économiques produits soient de bonne qualité.

***En termes de coût ;*** Il faut que l'entreprise garantisse sa compétitivité par des coûts de production les plus faibles possibles.

***En termes de délais ;*** Cela consiste à respecter des délais raisonnables, conformes avec le niveau de demande à laquelle doit faire face l'entreprise. Cela permettra d'éviter le stockage des bien finaux et le stockage des produits finis et de connaître des goulets d'étranglement.

***En termes de flexibilité ;*** C'est-à-dire la capacité du SP à pouvoir s'adapter aux variations de la demande.

### 1.3. Processus de production

Généralement, les systèmes de production sont classés comme suit (SASSINE, 1998) :

- ***Processus continus ;*** Tels que la production électrique, la chimie ou la papeterie.
- ***Processus discrets ;*** Tels que l'usinage de toutes les activités d'assemblage. Cette vision des systèmes est très fréquente dans l'industrie manufacturière.
- ***Processus discontinus ;*** Ils se situent par définition à mi-chemin entre les processus continus et les processus discrets. La production est continue mais le conditionnement des produits est discret.

### 1.4. Approche systémique des systèmes de production

La théorie des systèmes appliquée aux systèmes de production permet de décomposer ces derniers en trois sous-systèmes : le système physique de production, le système d'information et le système de décision (DOUMEINGTS, 1984) (ROBOAM, 1988) (Fig. 1-1) (BENBOUZID SI-TAYEB, 2005). Le système couramment appelé système de gestion de production est constitué par la partie du système de décision et du système d'information traitant des fonctions rattachées directement à la production (par exemple, les achats, les approvisionnements, la planification, la gestion des ressources, la maintenance, etc.).

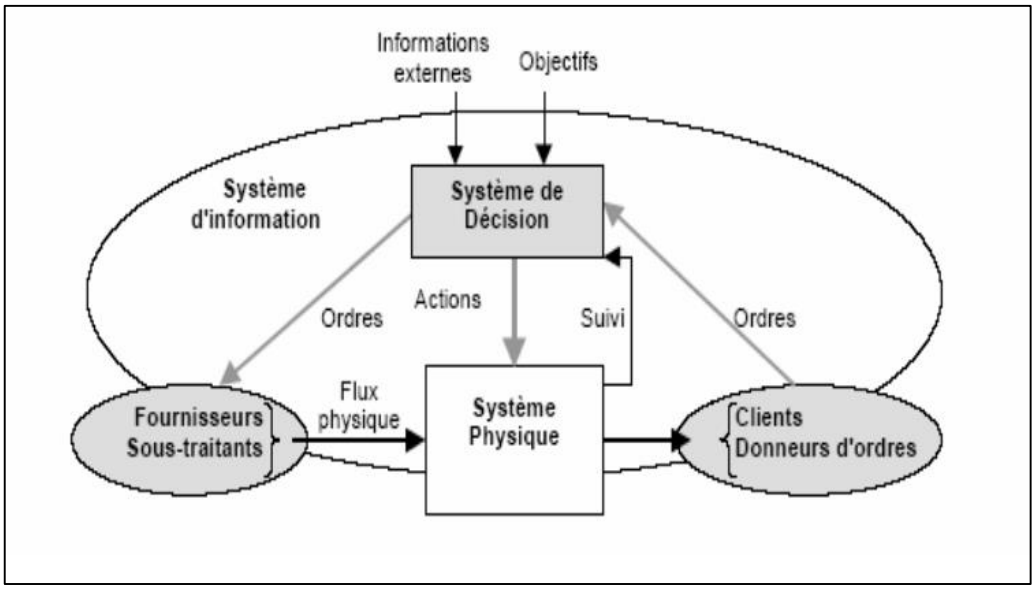



**Figure I-1: Modèle conceptuel d'un système de production**

En termes de système, le système physique transforme les matières premières en produits finis. Pour effectuer cette transformation, il est commandé par le système de gestion qui transforme les informations à caractère commercial en ordres de fabrication et ordres d'approvisionnements. Le système étant bouclé puisqu'en retour, il reçoit les informations de suivi du système physique pour pouvoir effectivement piloter ce dernier.

En termes de flux, un ensemble de flux régulés parcourent le système de production. Tout d'abord, le flux physique ou de matière qui transforme la matière première et les composants en produits finis, puis le flux d'information ou de suivi qui permet la circulation des informations nécessaires au contrôle et à la prise de décision. Enfin, le flux de décision ou ordre qui contrôle et pilote le système physique.

Le système de gestion est composé de différentes activités telles que :

–L'élaboration du Plan Directeur de Production (PDP)
–Le calcul des besoins bruts, nets et d'approvisionnements
–La gestion des stocks
–Les achats
–L'élaboration du plan de charge
–L'ordonnancement
–Le lancement.

### 1.5. Approche hiérarchique des systèmes de production

La recherche d'une grande efficacité de l'outil de production a été pendant très longtemps l'objectif principal de la gestion de production. Elle est structurée par des fonction ou services, spécialisés et liés hiérarchiquement, tout comme est structurée l'organisation de l'entreprise. Cette hiérarchie obéit à un modèle de résolution de problème, qui décompose les décisions en quatre niveaux (DOUMEINGTS, 1984) (ROBOAM, 1988) (Fig. 1-2) (BENBOUZID SI-TAYEB, 2005) :

- ***Long terme (décisions stratégiques) ;*** Cela concerne la politique à long terme de l'entreprise (en général, vision à plus de deux ans). Ce qui implique une définition cohérente du portefeuille d'activités et d'investissements.
- ***Moyen terme (décisions tactiques) ;*** C'est l'ensemble des décisions à moyen terme, fixant la gestion des stocks, la planification de la production, etc.

- ***Court terme (décisions opérationnelles) ;*** Ces sont les décisions qui déterminent l'ordonnancement et le pilotage.
- ***Très court terme (niveau d'exécution) ;*** Il transmet les ordres de fabrication et d'affectation des ressources au système physique.

Le travail que nous avons effectué se situes au ***niveau opérationnel***.

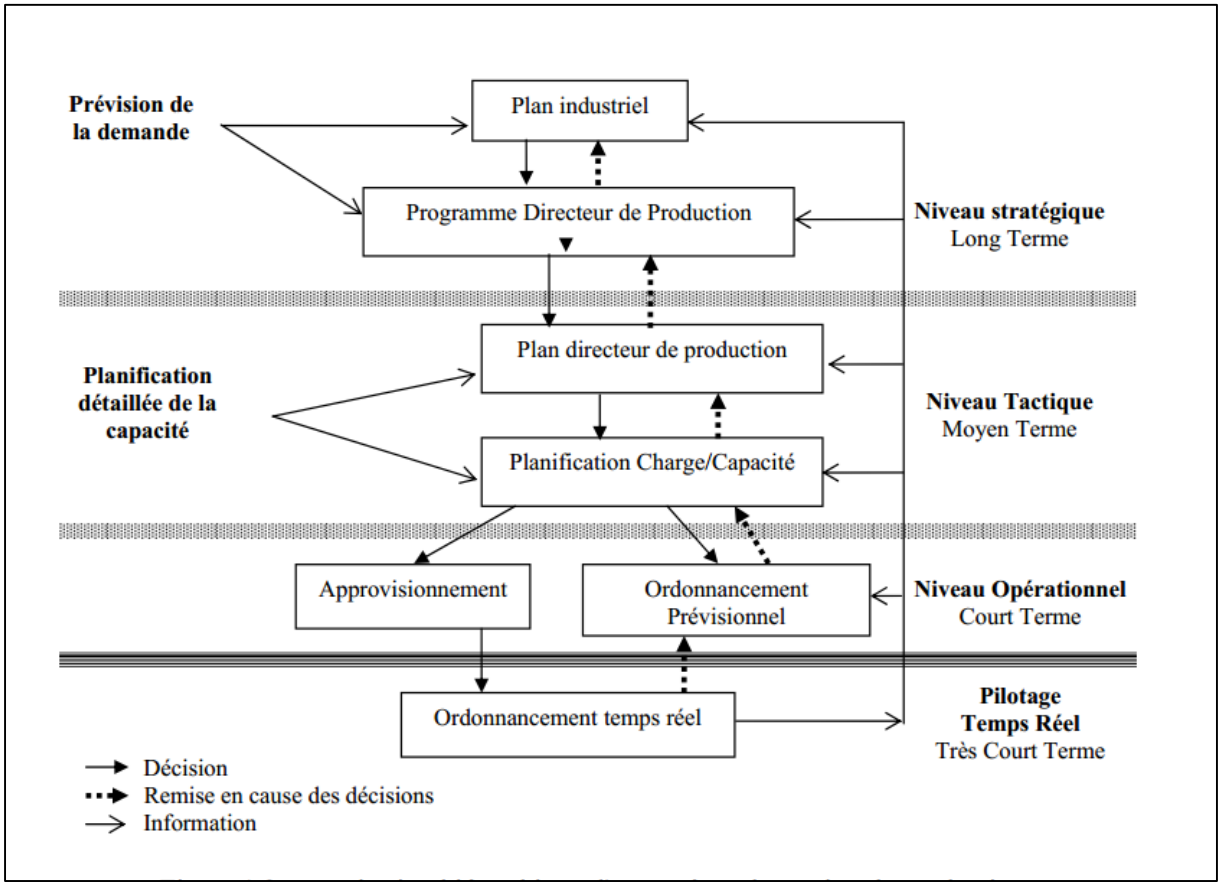



**Figure I-2: Hiérarchie d'un système de gestion de production**

### 1.6. Typologie des systèmes de production

Les SP sont caractérisés par la séquence des opérations nécessaire pour réaliser des produits finis. Face à la diversité des processus de production, Giard (GIARD, 1988) a proposé deux typologies :

***Typologie Liée à la demande :*** Les ordres de fabrication peuvent émaner, soit pour satisfaire des commandes sur mesure, soit pour mettre à niveau les stocks des produits finis.

***Typologie Liée aux ressources :*** Cette typologie dépend de la manière dont les ressources sont organisées pour traiter les matières premières ou les produits semi finis.

# 2. Problèmes d'ordonnancement

## 2.1. Paramètres de l'ordonnancement

### *2.1.1. Tâches*

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début et une date de fin et dont la réalisation nécessite une durée. Elle est aussi caractérisée par une date de début au plus tôt, avant laquelle elle ne peut être exécutée, une date de fin au plus tard, le retard, ainsi que les ressources nécessaires à sa réalisation. Elle peut être aussi préemptive ou non-préemptive.

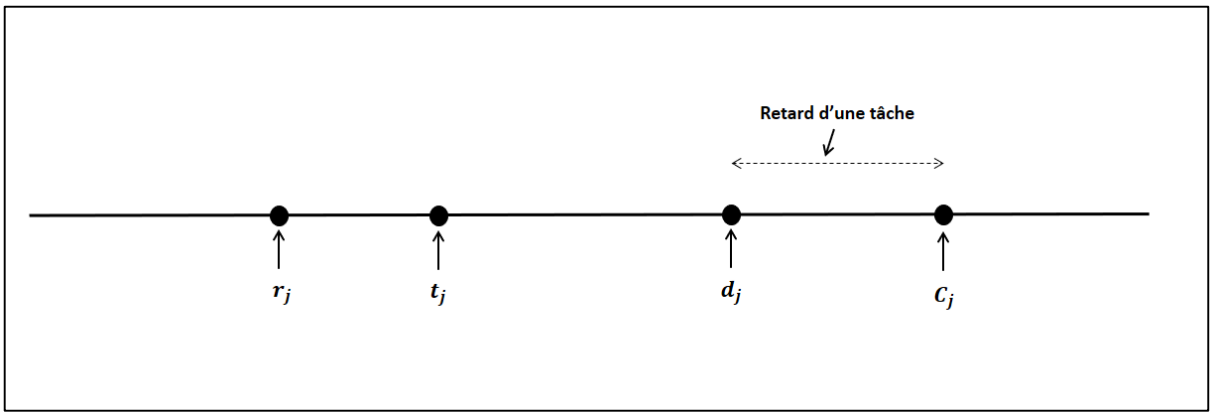



**Figure I-3: Représentation du retard d'une tâche**

Les données spécifiant chaque tâche $J_j$ peuvent être représentées comme suit :

- $p_{ij}$ Le temps de traitement de la tâche $J_j$ par la ressource $M_i$ (cas où la tâche est composée de plusieurs opérations).
- $p_j$ Le temps de traitement de la tâche $J_j$ (la tâche est équivalente à une seule opération).
- $r_j$ La date de disponibilité (***release date***) de la tâche$J_j$ (sa date de début au plus tôt).
- $d_j$ La date de fin au plus tard (***due date***) de la tâche $J_j$.
- $t_j$ La date de début d'exécution de la tâche $J_j$.
- $w_j$ Le poids de la tâche $J_j$.
- $C_j$ La date de fin de la tâche $J_j$ (***completion time***).
- $F_j$ La durée de séjour (ou de présence) de la tâche $J_j$ (flow time) : $F_j = C_j - r_j$.
- $T_j$ Le retard vrai de la tâche $J_j$ (***tardiness***) : $T_j = max\ (0, C_j - d_j)$.
- $E_j$ L'avance de la tâche $J_j$ (***earliness***) : $E_j = max\ (0, d_j - C_j)$.
- $L_j$L'écart par rapport à la fin *souhaitée* de la tâche $J_j$ (***lateness***) : $L_j = C_j - d_j$.

– $U_j$ Indicateur de retard de la tâche $J_j$ : $U_j = 1$ si $T_j > 0$, $U_j = 0$ sinon.

### *2.1.2. Ressources*

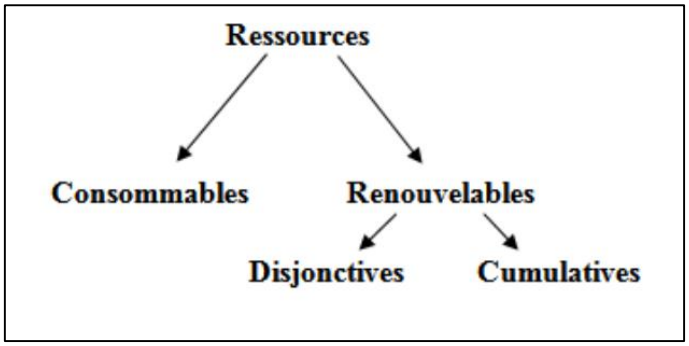



**Figure I-4: Classification des ressources**

- **Ressources consommables** Une ressource est consommable si après avoir été utilisée par une ou plusieurs tâches, elle n'est plus disponible, donc elle ne peut être utilisée plus d'une fois (LOPEZ, 1991). Les matières premières et le budget peuvent être considérés comme ressources consommables.
- **Ressources renouvelables** Une ressource renouvelable (réutilisable) est disponible à nouveau, après avoir été utilisée par une ou plusieurs tâches (les hommes, les machines, l'équipement en général…). Ce type de ressource peut être à son tour décomposé en deux types (LOPEZ, 1991):
  - **Les ressources disjonctives (non partageables) :** Qui ne peuvent n'exécuter qu'une tâche à la fois (machine-outil, robot manipulateur).
  - **Les ressources cumulatives (partageables) :** Qui peuvent être utilisées par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail).

### *2.1.3. Contraintes*

Une contrainte est une condition que doit respecter le plan de travail. Elle limite le degré de liberté d'exécution de processus (LOPEZ, 1991). Cette condition est liée aux opérations entre elles, aux ressources, etc.

- **Les contraintes potentielles** Ce sont des contraintes spécifiant les limites relatives à la localisation temporelle des tâches et à la réalisation de successions entre elles.
- **Les contraintes de localisation temporelles** Elles représentent les bornes de l'intervalle de temps durant lequel une tâche doit être traitée. L'intervalle est représenté par la date de début d'exécution au plus tôt et la date de fin d'exécution au plus tard.

$$r_i \leq t_i \qquad \text{et} \qquad c_i \leq d_i$$

- **Les contraintes de succession ou de précédence** Elles prennent en compte la succession des opérations de la gamme opératoire. Les contraintes indiquent que le début d'exécution d'une tâche j doit être après la date de fin d'exécution d'une autre opération $i$ qui la précède, d'où :

$$T_i \geq t_i + p_i$$

- **Les contraintes de ressources** Ces contraintes de limitation de ressources renouvelables expriment la nature de la quantité des moyens utilisés par les tâches. Il y a deux types de contraintes suivant la nature des ressources :
  - **Contraintes disjonctives :** Elles obligent à réaliser toute paire de tâches sur des intervalles de temps disjoints. Soit une tâche i, elle s'exécute soit avant j, soit après j.
  - **Contraintes cumulatives :** Elles interdisent la réalisation d'un nombre trop important de tâches, compte tenu de la disponibilité maximale de la ressource à chaque instant et des quantités requises individuellement par les tâches.

### *2.1.4. Critères d'optimisation*

Les critères à minimiser les plus répandus sont des fonctions des dates de fin des tâches $C_j$. Les fonctions objectif utilisées sont de type *Maximum* ou de type *Somme* (éventuellement pondérée) sur toutes les tâches à ordonnancer. Le critère le plus utilisé est la durée totale de l'ordonnancement correspondant à la date de fin de traitement de la tâche la plus tardive (*makespan*). Les critères les plus considérés sont données dans la figure 1-5 (KAABI, 2004).

| Critères | Maximum | Somme | Somme pondérée |
|---|---|---|---|
| Durée totale | $C_{max}=max_{j=1,n}\{C_j\}$ | $\overline{C}=\frac{1}{n}\sum_{j=1}^{n}C_j$ | $\overline{C}_w=\frac{1}{n}\sum_{j=1}^{n}w_jC_j$ |
| Décalage | $L_{max}=max_{j=1,n}\{L_j\}$ | $\overline{L}=\frac{1}{n}\sum_{j=1}^{n}L_j$ | $\overline{L_w}=\frac{1}{n}\sum_{j=1}^{n}w_jL_j$ |
| Retard | $T_{max}=max_{j=1,n}\{T_j\}$ | $\overline{T}=\frac{1}{n}\sum_{j=1}^{n}T_j$ | $\overline{T_w}=\frac{1}{n}\sum_{j=1}^{n}w_jT_j$ |
| Avance | $E_{max}=max_{j=1,n}\{E_j\}$ | $\overline{E}=\frac{1}{n}\sum_{j=1}^{n}E_j$ | $\overline{E}_w=\frac{1}{n}\sum_{j=1}^{n}w_jE_j$ |
| Durée de séjour | $F_{max}=max_{j=1,n}\{F_j\}$ | $\overline{F}=\frac{1}{n}\sum_{j=1}^{n}F_j$ | $\overline{F_w}=\frac{1}{n}\sum_{j=1}^{n}w_jF_j$ |
| Nombre de retards | | $\overline{U}=\frac{1}{n}\sum_{j=1}^{n}U_j$ | $\overline{U}_w=\frac{1}{n}\sum_{j=1}^{n}w_jU_j$ |

**Figure I-5:Critères d'optimisation usuels**

## 2.2. Classification des problèmes d'ordonnancement

Dans la littérature, il existe parmi les notations utilisées pour classifier les problèmes d'ordonnancement la notation de Lawler (GRAHAM, 1979). Cette notation a trois champs α | β | γ. Dans ce qui suit, nous allons énumérer tout ce que décrit chaque champ de cette notation.

### *2.2.1. Champ α*

Le champ α représente les environnement machines possibles, que sont (LEUNG, 2004) :

– **Une machine (**α = 1**)** : Il y a une seule machine dans le système. Chaque tâche est constituée d'une seule opération. Dans les problèmes à une machine, il faut déterminer une séquence des tâches optimisant le critère de performance. Les résultats obtenus pour les problèmes à une machine sont souvent utilisés pour résoudre des problèmes plus généraux.

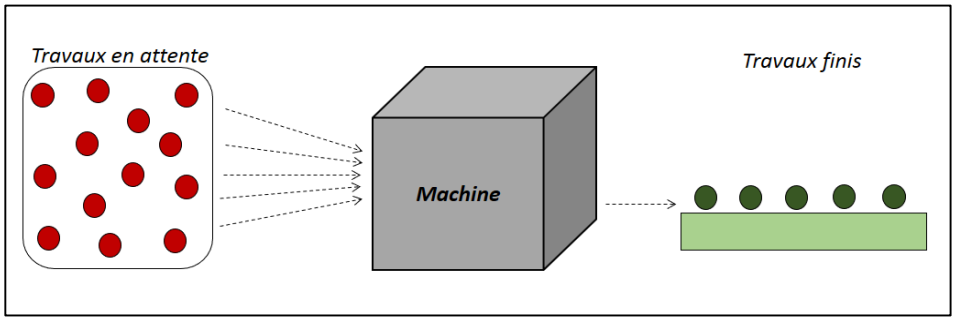



**Figure I-6: Ordonnancement sur machine unique**

– **Machines parallèles** : Le système est composé de *m* machines parallèles. Chaque tâche est constituée d'une seule opération et elle peut être traitée sur n'importe laquelle des *m* machines. Par contre, la vitesse d'exécution des tâches peut différer d'une machine à une autre. Pour cela, on distingue trois types de machines parallèles :

- Machines parallèles **identiques** (α = *Pm*) : La vitesse d'exécution est la même pour les *m* machines et pour toutes les tâches.
- Machines parallèles **uniformes** (α = *Qm*) : Les machines ont des vitesses différentes mais constantes. Une machine $i$, $1 \leq i \leq m$, a une vitesse $s_i$. Le temps $p_{ij}$ qu'une tâche $J_j$ mettra sur la machine $i$ est égal à $p_j / s_i$, en supposant que la tâche $J_j$ est complètement traitée sur la machine $i$.
- Machines parallèles **indépendantes** ou quelconques (α = *Rm*) : Une tâche peut être traitée sur chaque machine à des vitesses différentes. Une machine $i$ peut traiter une tâche $J_j$ à une vitesse $s_{ij}$. Le temps $p_{ij}$ qu'une tâche $J_j$ mettra sur la machine $i$ est égal à $p_j / s_{ij}$, en supposant que la tâche $J_j$ est complètement traitée sur la machine $i$. Dans les problèmes à

machines parallèles, il y a deux types de décisions à prendre : affecter les tâches aux machines et décider de l'ordre des tâches sur une même machine.

– **Flow shop** (α = *Fm*) : Dans un flow shop à *m* machines (Figure 1-7), les machines sont agencées en ligne et les tâches suivent toutes le même cheminement (de la première machine à la dernière machine). Cependant, deux tâches différentes peuvent avoir à passer des temps différents sur une même machine.

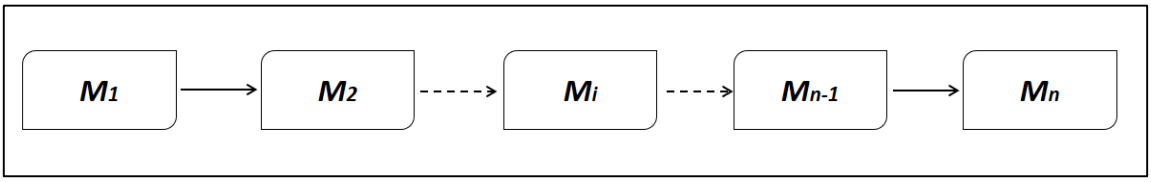



**Figure I-7: Environnement de type Flow shop**

– **Job shop** (α = *Jm*) : Dans un Job shop à *m* machines, chaque tâche a un cheminement prédéterminé à suivre. Elle peut visiter certaines machines plus d'une fois et elle peut ne pas visiter certaines machines du tout. L'ordre de visite des machines pour une tâche dépend de sa gamme (l'ordre des opérations qui la composent). Les durées des opérations sur les différentes machines font parties des données du problème. Par exemple, pour un job shop à 3 machines et 3 tâches, les gammes de chacune de tâches peuvent être comme sur la Figure I-8 :

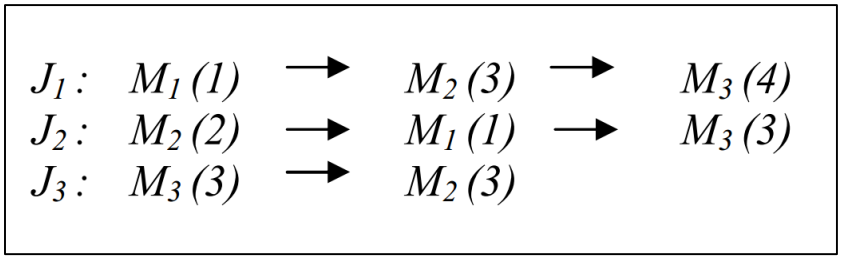



**Figure I-8: Les gammes de 3 tâches sur 3 machines dans un environnement Job shop**

– **Open shop** (α = *Om*) : Dans un open shop avec *m* machines, chaque tâche a besoin d'être traitée exactement une fois sur chacune des machines (comme dans un flow shop). Sauf que l'ordre de passage des opérations constituant la tâche n'est pas fixé *a priori* (cheminement libre). Contrairement aux autres modèles, l'open shop n'est pas courant dans les ateliers et il n'est pas beaucoup étudié dans la littérature.

### *2.2.2. Champ β*

Ce champ spécifie les caractéristiques des tâches et les contraints d'ordonnancement. Il peut contenir plusieurs paramètres (9 au total), de $\beta_1$, $\beta_2$ jusqu'à $\beta_9$.

- **Préemptions** (*pmtn*) : Si la préemption est permise, *pmtn* est inclus dans le champ β. Sinon, il n'est pas inclus.
- **Splitting** ou morcellement (*spl*) : Les tâches peuvent être découpées en morceaux et exécutées en parallèle sur plusieurs machines.
- **Sans attente** ou No-Wait (*nwt*) : La contrainte sans attente concerne seulement les flow shop. Si *nwt* est spécifiée dans le champ β alors les tâches ne sont pas autorisées à attendre entre deux machines successives.
- **Contraintes de précédence** (*prec*) : *prec* est spécifié dans le champ β quand certaines tâches doivent être achevées avant que le traitement d'autres tâches soit commencé.
- **Date de disponibilité** ou Release dates ($r_j$) : Si ce symbole n'est pas présent dans le champ β alors le traitement des tâches peut commencer à tout moment.
- **Restrictions sur le nombre de tâches** (*nbr*) : Si ce symbole est présent dans le champ β alors le nombre de tâches est restreint ; par exemple *nbr* = 5 veut dire qu'il y a au plus 5 tâches à traiter.
- **Restrictions sur le nombre d'opérations des tâches** ($n_j$) : Ce sous champ est seulement applicable au job shop. Par exemple $n_j$ =4 veut dire que chaque tâche est limitée à au plus quatre opérations. Si ce symbole n'est pas présent, alors le nombre d'opérations n'est pas restreint.
- **Restrictions sur les durées des tâches** ($p_j$) : Si ce symbole est présent, alors le temps de traitement de chaque tâche est restreint à une valeur *p* spécifiée.
- **Les délais** ($d_j$) : Si ce symbole est présent, alors il est souhaitable que chaque tâche $J_j$ soit terminée avant son délai fixé *a priori*.

### *2.2.3. Champ γ*

Le champ γ représente la fonction objectif à minimiser. Parmi les critères possibles il y a ceux donnés dans la figure 1-5.

## 2.3. Représentation des ordonnancements

Henry Gantt (1861 – 1919) a inventé le diagramme de Gantt, appelé aussi diagramme à barres. Il est devenu le moyen le plus simple et célèbre pour représenter une solution d'un ordonnancement. Sur ce graphique, on visualise l'enchaînement des opérations et les ressources les exécutant. Dans un ordonnancement sur machines parallèles identiques, l'axe des abscisses représente le temps et sur l'axe des ordonnées apparaissent les machines. Sur chaque ligne

horizontale, on met l'ordonnancement des tâches sur cette machine. Chaque tâche est représentée par une barre. La longueur de cette barre est proportionnelle à sa durée. A la fin, on obtient sur le digramme l'enchaînement de opérations sur chacune des machines, avec les dates de début et de fin de chaque tâche.

La Figure 1-8 représente un ordonnancement de 11 jobs sur 3 machines parallèles identiques.

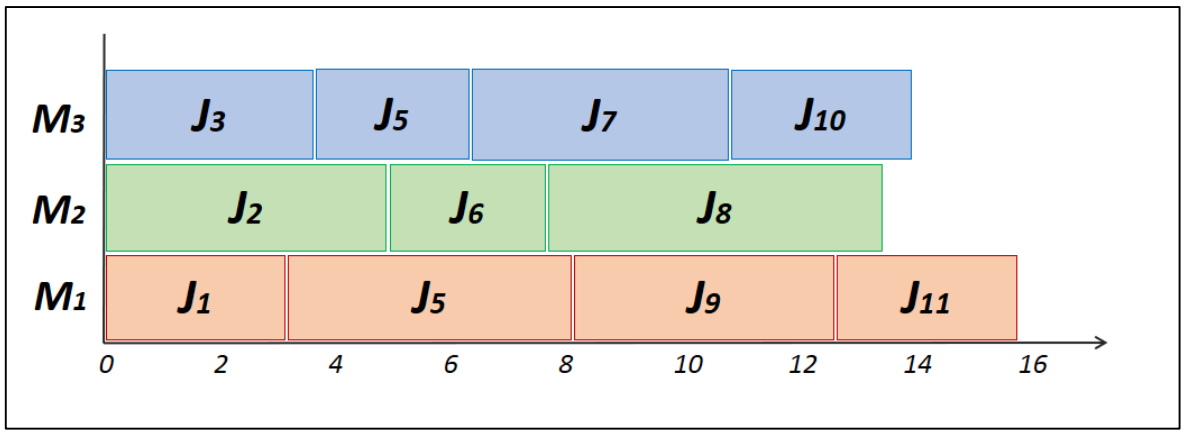



**Figure I-9: Représentation de l'ordonnancement par Diagramme de Gantt**

### 2.4. Méthodes de résolution

D'un atelier à un autre, les problèmes d'ordonnancement diffèrent. En addition à cela, les progrès technologiques de l'industrie laissent émerger encore de nouveaux problèmes. Le plus grand challenge pour les scientifiques est la recherche permanente de nouveaux algorithmes efficaces pour résoudre ces problèmes. Dans cette optique, la théorie de la complexité est un outil indispensable pour analyser les coûts de résolutions, notamment en termes de temps d'exécution des problèmes de recherche combinatoire.

Il existe différentes méthodes de résolution résoudre les problèmes de la classe NP-Complet : les méthodes approchées (heuristiques ou méta-heuristiques) et les méthodes exactes. Les méthodes exactes réussissent à trouver la solution optimale pour des problèmes de petite taille. Cela dit, au fur et à mesure que la taille des problèmes augmente, l'obtention de la solution optimale nécessite un temps exponentiel et ces méthodes deviennent impraticables. On applique alors des méthodes approchées qui donnent, en un temps raisonnable, une solution proche de l'optimum.

# 3. Gestion de la maintenance

## 3.1. Définition

La maintenance est l'ensemble de toutes les actions techniques, administratives et de management durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise (AFNOR, 2010).

## 3.2. Terminologie

- **Stratégie de maintenance :** méthode de management utilisée en vue d'atteindre les objectifs de maintenance.
- **Management de la maintenance :** toutes les activités des instances de direction qui déterminent les objectifs, la stratégie et les responsabilités concernant la maintenance et qui les mettent en application par des moyens tels que la planification, la maîtrise et le contrôle de la maintenance, l'amélioration des méthodes dans l'entreprise, y compris dans les aspects économiques.
- **Objectifs de maintenance :** buts fixés et acceptés pour les activités de maintenance. Ces buts peuvent comprendre par exemple la disponibilité, les coûts, la qualité du produit, la protection de l'environnement, la sécurité.
- **Logistique de maintenance :** ressources, services et moyens de gestion nécessaires à l'exécution de la maintenance. La logistique de maintenance peut inclure par exemple le *personnel*, les *équipements d'essai*, les *ateliers*, les *pièces de rechange*, *la documentation*, les *outils*, etc.

## 3.3. Objectifs

La fonction maintenance a un caractère productif tout comme la fonction fabrication. On parle souvent de la maintenance productive, et il convient de lui attacher une importance aussi grande que la fonction fabrication. Les deux ont la tâche d'assurer une conduite et une qualité constante de la production. Pour cela, les activités de maintenance doivent avoir des objectifs et but fixés et validés par l'entreprise afin de :

- Maintenir l'équipement dans un bon état de marche, dans les meilleures conditions de qualité, de délai et de prix de revient.
- Améliorer la sécurité du travail.
- Remplacer l'équipement à des périodes prédéterminées.

- Former le personnel dans les spécialités spécifiques á la maintenance.
- Conseiller la direction d'usine et la fabrication.
- Assurer á l'équipement des performances de haute qualité.
- Maintenir l'installation dans un état de propreté absolue.

### 3.4. Approches de maintenance dans l'entreprise

Il existe deux tendances quant au positionnement de la maintenance dans l'entreprise :

**La centralisation**, où toute la maintenance est assurée par un service (Figure 1-10) (AFNOR, 2010). Cette approche a comme avantages :

- Standardisation des méthodes, des procédures et des moyens de communication
- Possibilité d'investir dans des matériels onéreux grâce au regroupement
- Vision globale de l'état du parc des matériels à gérer
- Gestion plus aisée et plus souple des moyens en personnels
- Rationalisation des moyens matériels et optimisation de leur usage (amortissement plus rapide)
- Diminution des quantités de pièces de rechange disponibles
- Communication simplifiée avec les autres services grâce à sa situation centralisée.

**La décentralisation**, où la maintenance est confiée à plusieurs services, de dimension proportionnellement plus modeste, et liés à chacun des services de l'entreprise. Cette approche a comme avantages :

- Meilleures communications et relations avec le service responsable et utilisateur du parc à maintenir
- Effectifs moins importants dans les différentes antennes
- Réactivité accrue face à un problème
- Meilleure connaissance des matériels
- Gestion administrative allégée

Il va de soi que les 2 modèles d'organisation étant contraires, les avantages de l'un sont souvent les inconvénients de l'autre.

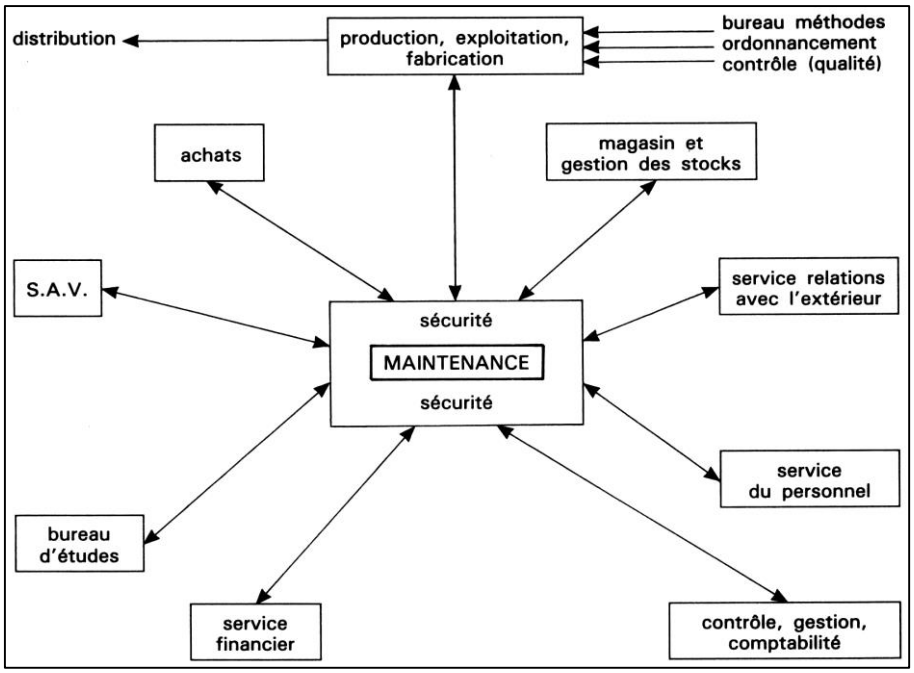



**Figure I-10: Maintenance Centralisée**

## 3.5. Méthodes ou politiques de maintenance

Le choix entre les méthodes de maintenance s'effectue dans le cadre de la politique de la maintenance et doit s'opérer en accord avec la direction de l'entreprise. Pour choisir, il faut donc être informé des objectifs de la direction, des directions politiques de maintenance, mais il faut connaître le fonctionnement et les caractéristiques des matériels, le comportement du matériel en exploitation, les conditions d'application de chaque méthode, les coûts de maintenance et les coûts de perte de production.

Le diagramme suivant (Figure 1-11) (AFNOR, 2010) synthétise les méthodes de maintenance.

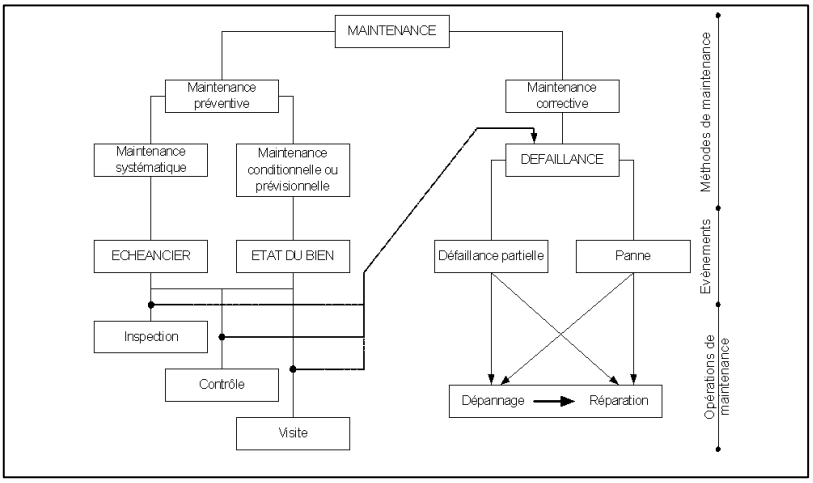



**Figure I-11: Méthodes de maintenance dans l'entreprise**

### *3.5.1. Maintenance corrective*

La maintenance corrective appelée parfois curative (terme non normalisé) a pour objet de redonner au matériel des qualités perdues nécessaires à son utilisation. La maintenance corrective peut être (AFNOR, 2010) :

- Différée : maintenance corrective qui n'est pas exécutée immédiatement après la détection d'une panne, mais est retardée en accord avec des règles de maintenance données.
- D'urgence : maintenance corrective exécutée sans délai après détection d'une panne afin d'éviter des conséquences inacceptables.

Les défauts, pannes ou avaries divers exigeant une maintenance corrective entraînent une indisponibilité immédiate ou à très brève échéance des matériels affectés et/ou une dépréciation en quantité et/ou qualité des services rendus.

### *3.5.2. La maintenance préventive*

La maintenance préventive est la maintenance exécutée à des intervalles prédéterminés ou selon des critères prescrits. Elle est destinée à réduire la probabilité de défaillance ou la dégradation du fonctionnement d'un bien (AFNOR, 2010). Elle doit permettre d'éviter les défaillances des matériels en cours d'utilisation. L'analyse des coûts doit mettre en évidence un gain par rapport aux défaillances qu'elle permet d'éviter.

Les objectifs principaux de la maintenance préventive sont :

- Augmenter la durée de vie des matériels
- Diminuer la probabilité des défaillances en service
- Diminuer les temps d'arrêt en cas de révision ou de panne
- Prévenir et aussi prévoir les interventions coûteuses de maintenance corrective
- Permettre de décider la maintenance corrective dans de bonnes conditions
- Eviter les consommations anormales d'énergie, de lubrifiant, etc.
- Améliorer les conditions de travail du personnel de production
- Diminuer le budget de maintenance
- Supprimer les causes d'accidents graves

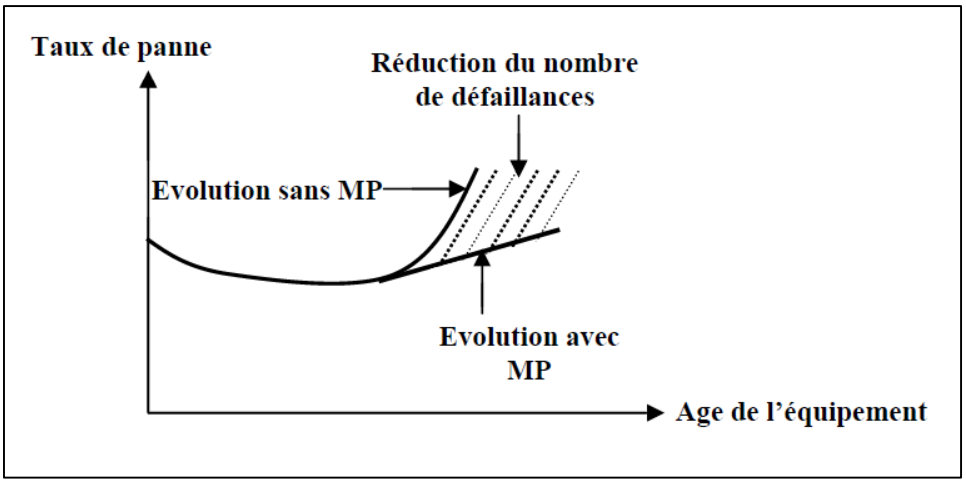



**Figure I-12: Effet d'une bonne maintenance préventive**

***Maintenance préventive systématique*** : C'est la maintenance préventive exécutée à des intervalles de temps préétablis ou selon un nombre défini d'unités d'usage mais sans contrôle préalable de l'état du bien (AFNOR, 2010).

***Maintenance préventive conditionnelle*** **:** Basée sur une surveillance du fonctionnement du bien et/ou des paramètres significatifs de ce fonctionnement intégrant les actions qui en découlent. La surveillance du fonctionnement et des paramètres peut être exécutée selon un calendrier, ou à la demande, ou de façon continue (AFNOR, 2010).

Il faut noter que la maintenance conditionnelle est donc une maintenance dépendante de l'expérience et faisant intervenir des informations recueillies en temps réel.

***Maintenance préventive prévisionnelle*** **:** C'est une maintenance préventive conditionnelle exécutée en suivant les prévisions extrapolées de l'analyse et de l'évaluation de paramètres significatifs de la dégradation du bien (AFNOR, 2010).

## Conclusion

Dans ce chapitre, nous avons présenté les systèmes de production tant au niveau fonctionnel qu'au niveau structurel, les notions relatives à l'ordonnancement ainsi que la gestion de la maintenance. Les informations disponibles dans ce chapitre nous permettent de pouvoir bien comprendre les concepts énoncés précédemment pour se projeter au prochain chapitre, contenant une étude bibliographique sur le problème d'ordonnancement conjoint Production/Maintenance, ainsi que les approches de résolution relatives à notre problématique.

# Chapitre II: Problème d'ordonnancement conjoint production/maintenance.

# Introduction

Plusieurs travaux de recherche ont été réalisés dans Les domaines d'ordonnancement de la production et la planification de la maintenance et la grande majorité de ces travaux les traitent indépendamment même si les deux fonctions ont un champ d'action commun (les machines) et contribuent toutes les deux à la productivité du système.

Vu la relation étroite liant ces deux domaines et l'importance gagnée par la maintenance, Les problèmes conjoints de production et de maintenance ont attiré l'attention des chercheurs durant ces deux dernières décennies puisque le couplage de la production et de la maintenance peut s'avérer une solution ambitieuse qui permettrait aux industriels de rester compétitifs et de répondre aux exigences des clients.

# 1. Problématique

Le problème que nous nous proposons d'étudier est le problème d'ordonnancement conjoint production/maintenance sur machines parallèles identique, ou la préemption des taches n'est pas permise. Les critères de performance sont la somme des retards des tâches pour la partie production et la somme des avances et des retards pour la partie maintenance.

# 2. Etat de l'art

Dans cette section nous décrivons brièvement les différentes politiques d'ordonnancement conjoint de la production et de la maintenance recensées dans la littérature, ensuite nous présentons un état de l'art sur l'ordonnancement intégré.

## 2.1. Politiques d'ordonnancement conjoint Production/Maintenance

L'objectif de l'ordonnancement conjoint de la production et de la maintenance est de planifier l'exécution des tâches de maintenance, en altérant le moins possible le plan de production, et tout en respectant au mieux la périodicité de maintenance des équipements. Trois politiques de planification ont été recensées dans la littérature l'ordonnancement séparé, le séquentiel et l'intégré.

### *2.1.1. Ordonnancement séquentiel*

Cette politique consiste à planifier l'une des deux activités, production ou maintenance, et à utiliser cet ordonnancement comme une contrainte supplémentaire d'indisponibilité des

ressources dans la résolution du problème d'ordonnancement de l'ensemble des deux types de taches. De manière générale, la maintenance est planifiée en premier, ensuite l'ordonnancement de la production est réalisé en prenant les opérations de maintenance comme des contraintes fortes d'indisponibilité des ressources (AGGOUNE, 2002).

### *2.1.2. Ordonnancement intégré*

Cette politique consiste à créer un ordonnancement conjoint et simultané des tâches de production et de maintenance (T. W. SLOAN, 2000). Une telle politique de planification limite les risques d'interférence entre la production et la maintenance et permet ainsi d'optimiser la qualité des ordonnancements conjoints.

## 2.2. Etude bibliographique

Les travaux de recherche traitants le problème d'ordonnancement des tâches de production et de maintenance peuvent être classés en deux catégories : l'approche stochastique et l'approche déterministe. Dans l'approche stochastique (intégrée), les dates de début des actions de maintenance aussi bien que les taches de production sont considérées comme des variables de décision du problème, les taches de production et de maintenance sont ordonnancées simultanément. Dans l'approche déterministe (séquentielle), les intervalles de temps des actions de maintenance préventive (MP) ainsi que leur nombre sont connus et fixés à l'avance. Dans la littérature la plupart des travaux d'ordonnancement avec des taches de maintenance adoptent cette approche qui est appelée « ordonnancement avec contraintes de disponibilité des machines ». Dans cette approche toutes les configurations machines connus ont été approchés par les chercheurs : machine unique, machines parallèles, flow shop, job shop, open shop et systèmes hybrides.

La plupart des études rencontrées qui s'intéressent à la production ignorent la maintenance, ou bien considèrent que les périodes de maintenance sont déjà planifiées à priori. Dans ce contexte, quelques études ont été consacrées à l'optimisation des deux fonctions relatives à la maintenance et la production, mais séparément. La majorité des études d'ordonnancement prenant en compte la maintenance supposent que l'intervalle de temps des actions de maintenance préventive ainsi que leur nombre sont connus à l'avance.

Cependant, dans les modèles intégrés, les actions de maintenance préventive doivent être considérées comme des variables de décision, ce qui est de même pour les actions de production. Lee (LEE, 1996) et Schmidt (SCHMIDT, 2000) analysent cette approche en

étudiant différentes contraintes et différents modèles d'ateliers de production. Ils s'intéressent aussi aux problèmes dans lesquels les dates d'exécution des tâches de maintenance ne sont pas fixées à l'avance.

Dans cette approche, la maintenance est souvent prioritaire par rapport à la production. Kaabi et al. (KAABI, 2002) proposent différentes heuristiques pour la maintenance et l'ordonnancement sur une seule machine afin de minimiser le retard maximal. Les auteurs supposent que les tâches de maintenance sont effectuées dans un intervalle fixé à l'avance.

Kaabi et al. (KAABI, 2003) considèrent un problème de production et de maintenance dans un système de type flow shop et supposent que les dates d'interventions de la maintenance varient dans un intervalle déterminé à l'avance. Ils présentent un algorithme génétique pour la résolution du problème étudié.

Cassady et Kutanoglu (CASSADY & KUTANOGLU, 2003) formulent un modèle intégré pour minimiser le retard total pondéré de production sur une seule machine. Ils proposent une méthode exacte de résolution inefficace pour les problèmes de taille industrielle.

Ruiz et al. (RUIZ, 2007) mettent en place un modèle intégré afin de résoudre un problème de type flow shop où les périodes de maintenance sont fixés afin de conserver un niveau minimum de fiabilité pour chaque machine. Le niveau de fiabilité est considéré comme une contrainte. Leur objectif est de minimiser le makespan.

Xu et al. (XU, 2008) considèrent un problème intégré dans un environnement de machines parallèles dans lesquelles activités de maintenance sont quasi-périodiques. Les auteurs considèrent le makespan comme objectif mais ils défavorisent l'aspect production dans leurs recherches.

Berrichi et al. (BERRICHI, 2009) proposent un modèle intégré de la maintenance préventive et d'ordonnancement de la production sur des machines parallèles. Ils utilisent un modèle de fiabilité pour prendre en compte l'aspect de la maintenance. Ils proposent deux algorithmes génétiques pour optimiser deux critères qui sont : la minimisation du temps d'exécution total pour la partie production et la minimisation de l'indisponibilité du système pour la partie maintenance.

Berrichi et al. (BERRICHI, 2010) traitent le même problème décrit précédemment, ils proposent une méta-heuristique basée sur les colonies de fourmis pour minimiser le makespan

pour l'aspect production et pour optimiser l'indisponibilité du système relative à l'aspect maintenance.

Moradi et al. (MORADI, 2011) considèrent un problème intégré de type job shop flexible avec une activité de maintenance préventive. Ils proposent un algorithme génétique pour la minimisation du makespan pour la partie production et la minimisation de l'indisponibilité du système pour la partie de maintenance.

Berrichi et al. (BERRICHI, 2013) proposent une extension de la problématique étudiée par Berrichi et al. (BERRICHI, 2009) en considérant une approche de colonies de fourmis pour minimiser le retard maximal (pour le service de production) et l'indisponibilité du système (pour le service de maintenance).

Dans le papier de Belkaid et al. (BELKAID, 2013), les auteurs entament une première étude et ceci afin de minimiser le temps de sortie des taches en essayant d'intégrer les critères relatifs à la production avec celle de la maintenance.

A travers l'état de l'art réalisé, nous pouvons remarquer qu'une mauvaise gestion des ressources peut dégrader les performances du système.

## 3. Modélisation du problème

Afin de pouvoir présenter notre modèle, nous allons utiliser les notations suivantes :

- Le nombre de tâches de production : $n$ .
- Le nombre de tâches de maintenance préventive : $k$.
- Le temps de traitement de la tâche $J_j$ (la tâche est équivalente à une seule opération) : $p_j$.
- La date de fin au plus tard de la tâche $J_j$ : $d_j$
- La date de fin de la tâche $J_j$ : $C_j$
- Le retard de la tâche $J_j$ : $T_j = max\,(0, C_j - d_j)$.
- L'avance de la tâche $J_j$ : $E_j = max\,(0, d_j - C_j)$.

Les critères sont représentés comme suit

- Somme des retards des tâches de production.

$$f_1 = \sum_{j=1}^{n} T_j = \sum_{j=1}^{n} max\,(0, C_j - d_j)$$

– Somme des retards et des avances des tâches de maintenance préventive.

$$f_2 = \sum_{i=1}^{k} T_i + E_i = \sum_{i=1}^{k} [\, max\,(0, C_i - d_i) + max\,(0, d_i - C_i)\,]$$

Cependant, les contraintes liées à notre problème sont les suivantes :

– Une machine ne peut exécuter qu'une seule tâche à la fois

– Une tâche ne peut être exécutée que par une seule machine.

– La préemption des tâches n'est pas autorisée.

La collaboration entre les services de production ainsi que ceux de la maintenance est primordiale, afin de maximiser la productivité du système en question. Pour cela, il faudrait que les objectifs liés aux deux services soient considérés aussi importants l'un vis-à-vis de l'autre.

Cependant, il existe un contraste très fort entre les deux services. Les critères liés à la production et à la maintenance sont antagonistes. La décroissance de l'un accroit l'autre. La solution est donc de trouver un compromis entre les deux objectifs. Donc le problème est un ordonnancement bi-objectif. Pour résoudre ce problème, les méthodes d'optimisation multi-objectif sont les plus appropriées. Les méthodes les plus utilisées sont les méthodes Pareto d'optimisation multi-objectif. Le but principal des décideurs et de concevoir un solveur capable d'assurer un système en bon état de fonctionnement avec maximalisation de sa disponibilité.

Il est à noter que pendant notre recherche bibliographique, nous n'avons pas rencontré de travaux qui considèrent la somme des avances et des retards comme critère d'optimisation pour la maintenance préventive. Nous pouvons dire que nous sommes parmi les premiers -si ce n'est les premiers- à proposer ce modèle.

## 4. Optimisation Multi-Objectif

### 4.1. Problème d'optimisation multi-objectif

L'optimisation multi-objectif est une partie indissociable de l'optimisation. Il est évident que la plupart des problèmes pourraient être modélisés par des critères contradictoires. Le moyen le plus courant est de les convertir en problèmes scalaires, c'est-à-dire de les traduire en problèmes mono-objectif même s'il existe des méthodes évolutionnaires pour les résoudre.

Un problème d'optimisation multi-objectif (Noté MOOP en littérature ou bien PMO en français) traite plusieurs fonctions objectif. Dans la plupart des problèmes de décision, on trouve évidemment soit plusieurs objectifs ou plusieurs critères d'optimisation. Au tout début,

pour cause de manque de méthodes adéquates de résolution multi-objectif, les PMO étaient principalement modélisés puis résolus en tant que problème classique mono-objectif. Néanmoins, Il existe des différences fondamentales entre leurs principes de fonctionnement.

Dans un problème mono-objectif, le but est de trouver une seule solution qui optimise l'unique fonction objectif. Si on étend ce raisonnement aux PMO, on pourrait déduire que le but serait de trouver des solutions optimales qui correspondent une à une à chacune des fonctions objectif.

## Conclusion

Dans ce chapitre, nous avons présenté notre problématique, un état de l'art sur les approches relatives à sa résolution puis quelques notions essentielles afin de pouvoir commencer à chercher des approches nous permettant d'implémenter notre solveur bi-objectif. Ces approches sont énoncées et décrites dans le prochain chapitre.

# Chapitre III: Méthodes d’optimisation Multi-objectif

# Introduction

Résoudre un problème d'optimisation combinatoire, c'est trouver l'optimum d'une fonction, parmi un nombre fini de choix, souvent très grand. Les applications concrètes sont nombreuses, que ce soit dans le domaine de l'informatique, la production industrielle, des transports ou de l'économie -partout où se fait sentir le besoin de minimiser des fonctions numériques, dans des systèmes où interviennent simultanément un grand nombre de paramètres-. A ces problèmes de minimisation, il existe deux classifications : Point de vue décideur, et point de vue concepteur. Dans ce chapitre, nous allons proposer une définition pour les deux approches, ainsi que quelques techniques utilisées, puis présenter les algorithmes que nous allons utiliser pour résoudre notre problématique.

## 1. Classification des approches de résolution

Dans les différentes publications rencontrées, nous avons retrouvé deux classifications différentes des approches de résolution des problèmes multi-objectifs. La première catégorie adopte un point de vue décideur, les approches sont classées en fonction de l'usage qu'on désire en faire. La seconde adopte un point de vue concepteur, les approches sont triées de leur façon de traiter les fonctions objectif.

Donc, avant de se lancer dans la résolution d'un PMO, il faut se poser la question du type d'approches de résolution que l'on veut utiliser.

### 1.1. Classification « point de vue décideur »

On distingue trois schémas possibles. Soit le décideur intervient dès le début de la définition du problème, en exprimant ses souhaits et préférences, afin de transformer un PMO en un problème simple objectif. Soit le décideur effectue son choix dans l'ensemble des solution proposées par le solveur multi-objectif.

#### *1.1.1. Approches à priori*

Le décideur intervient en aval du processus d'optimisation, pour définir la fonction d'agrégation modélisant le compromis que l'on désire faire entre les différents objectifs. Dans ce cas, le décideur est supposé connaître à priori le poids de chaque objectif afin de les mélanger dans une fonction unique. Cela revient à résoudre un problème mono-objectif.

Cependant, dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, parce que les différents objectifs sont non commensurables (exprimés dans des unités différentes).

### *1.1.2.Approches interactives (Ou bien progressives)*

Elles combinent de manière cyclique et incrémentale les processus de décision et d'optimisation. Le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi interactivement le compromis entre ses préférences et les résultats. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche.

### *1.1.3. Approches à posteriori*

Elles cherchent à fournir au décideur un ensemble de bonnes solutions bien réparties. Il peut ensuite, au regard de l'ensemble des solutions, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contrepartie fournir un ensemble de solutions bien réparties, ce qui peut également être difficile et requérir un temps de calcul important (mais ne nécessite pas la présence du décideur).

## 1.2. Classification « point de vue concepteur »

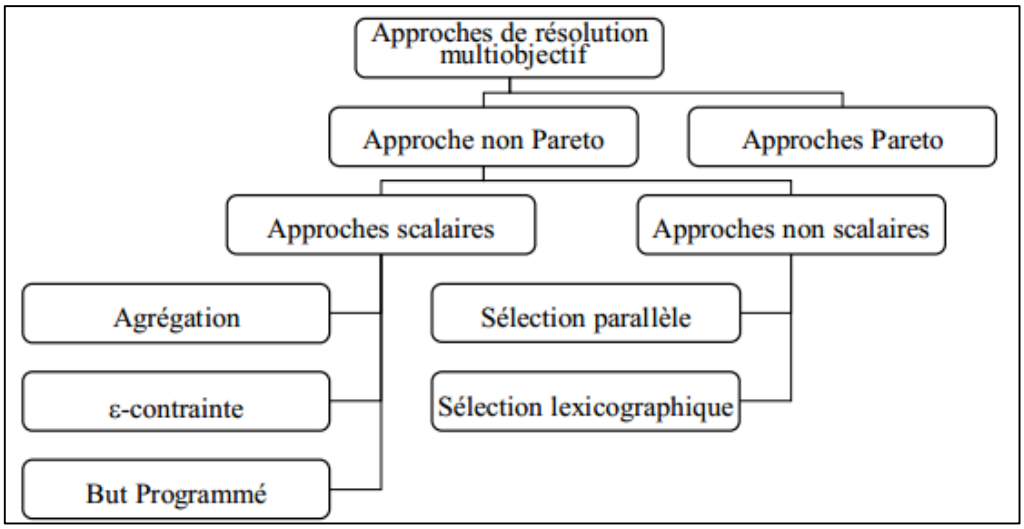



**Figure III-1: Classification "point de vue concepteur"**

Cette classification adopte un point de vue plus théorique articulé autour des notions d'agrégation et d'optimum *Pareto.* Les approches utilisées pour la résolution des problèmes multi-objectifs peuvent être classées en deux catégories (BARICHARD, 2003) : les approches non Pareto et les approches Pareto.

### *1.2.1. Approches non Pareto*

Les approches non Pareto sont classées en deux catégories : les approches scalaires, qui transforment le problème multi-objectif en problème mono-objectif et les approches non scalaires, qui gardent l'approche multi-objectif, mais en traitant séparément chacun des objectifs.

#### *1.2.1.1. Approches scalaires*

A l'origine, les problèmes multi-objectifs étaient transformés en problèmes mono-objectifs. Plusieurs approches différentes ont été mises au point pour transformer les problèmes multi-objectifs en problèmes mono-objectifs : les approches agrégées (ISHIBUCHI, 1998), programmation par but (COELLO COELLO, 1998), et les approches $\varepsilon$-contraintes, etc. Ces approches sont de type à priori.

#### *1.2.1.2. Approches non scalaires non Pareto*

Ces approches ne transforment pas le problème multi-objectif en un problème mono-objectif, mais utilisent des opérateurs qui traitent séparément les différents objectifs, elles n'utilisent pas non plus la notion de dominance Pareto : sélection parallèle (VEGA) (SCHAFFER, 1985), sélection lexicographique (FOURMAN, 1985). Ces approches sont de type à posteriori.

### *1.2.2.Approches Pareto*

Au 19éme Siècle, Vilfredo Pareto, un mathématicien Italien, formule le concept suivant : dans un problème multi-objectif, il existe un équilibre tel que l'on ne peut pas améliorer un objectif sans détériorer au moins un des autres objectifs. Les approches Pareto utilisent directement la notion de **dominance** dans la sélection des solutions générées. Le principal avantage de ces approches, c'est l'optimisation simultanée d'objectifs contradictoires. Ces approches sont de type à posteriori. Nous donnerons plus de détails sur ces approches (COLETTE & SIARRY, 2002) dans la section suivante.

# 2. Approche Pareto

## 2.1.Définitions

### *2.1.1.Dominance au sens Pareto*

Soient deux vecteurs objectifs $Y^1, Y^2 \epsilon \varphi \ / \ Y^1 = F(S^1),\ Y^2 = F(S^2)$. On dit que la solution $S^1$ domine $S^2$ ($Y^1$ Domine $Y^2$) si et seulement si : $Y^1 \geq Y^2$ Et $Y^1 \neq Y^2$. (i.e., $y_k^1 \geq y_k^2$ pour tout $k = 1 \ldots p$, et $y_k^1 > y_k^2$ pour au moins un $k$). On notera alors : $S^1 \geq S^2$. Si $S^1$ Est meilleur que $S^2$ Sur tous les objectifs (ie $y_k^1 > y_k^2$ pour tout $k = 1 \ldots p$) alors on dit que $S^1$ Domine fortement $S^2$; On notera alors $S^1 > S^2$. Lorsque ni $S^1 \geq S^2$, ni $S^2 \geq S^1$, alors on dit qu'elles sont incomparables ou Pareto équivalentes, $S^1 \sim S^2$. La relation de dominance est une relation d'ordre partiel stricte transitive, non réflexive et non antisymétrique (DUPAS, 2004).

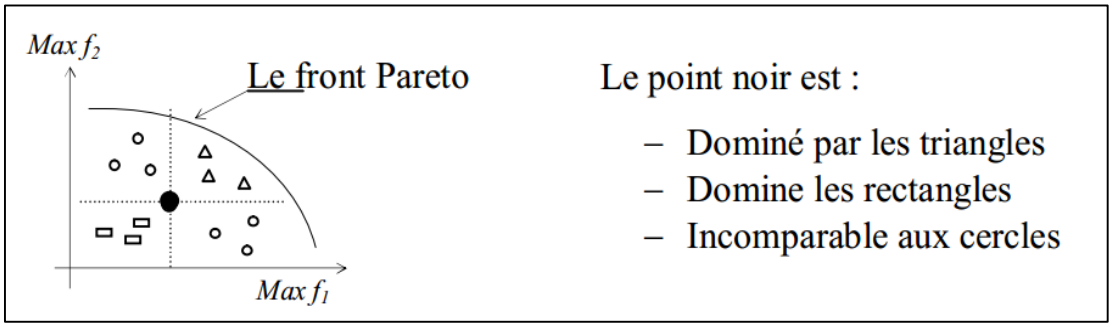



**Figure III-2:Relation de dominance ; cas bi-objectif**

### *2.1.2.Solution Pareto-optimale*

Une solution est dite Pareto optimale si elle n'est dominée par aucune autre solution réalisable.

### *2.1.3.Front Pareto*

L'ensemble Pareto optimal $PO^* = \{ S \in \omega \mid \neg \exists S' \in \omega, F(S') \geq F(S) \}$.

L'image de l'ensemble Pareto optimal $F\ (P\ O)$ dans l'espace objectif $Y$ est appelée **frontière** Pareto, ou surface de compromis. L'allure de cette frontière prend des formes différentes selon que les objectifs doivent être minimisés ou maximisés, (Figure III-3) cas de deux objectifs.

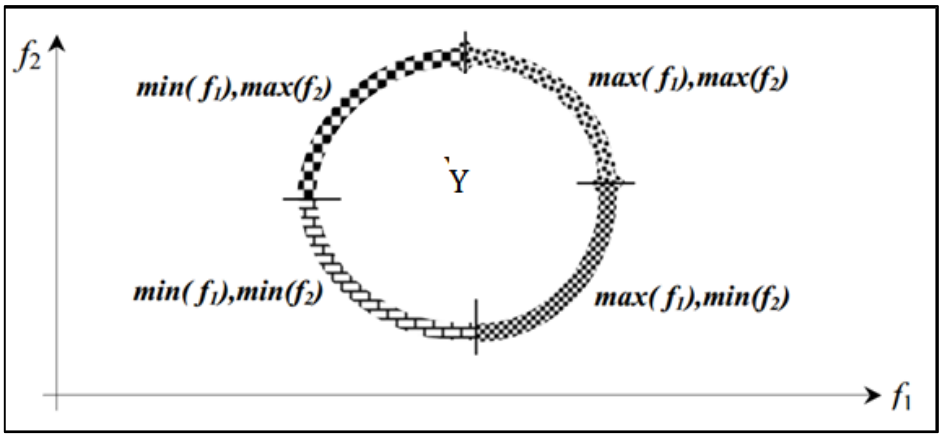



**Figure III-3:Allure de la frontière Pareto selon l'optimisation (min, max) des différents objectifs**

L'ensemble Pareto optimal regroupe des solutions dites **supportées** correspondants aux sommets de la fermeture convexe de la frontière et des solutions **non-supportées** n'appartenant pas à cette fermeture convexe.

### *2.1.4.Point idéal*

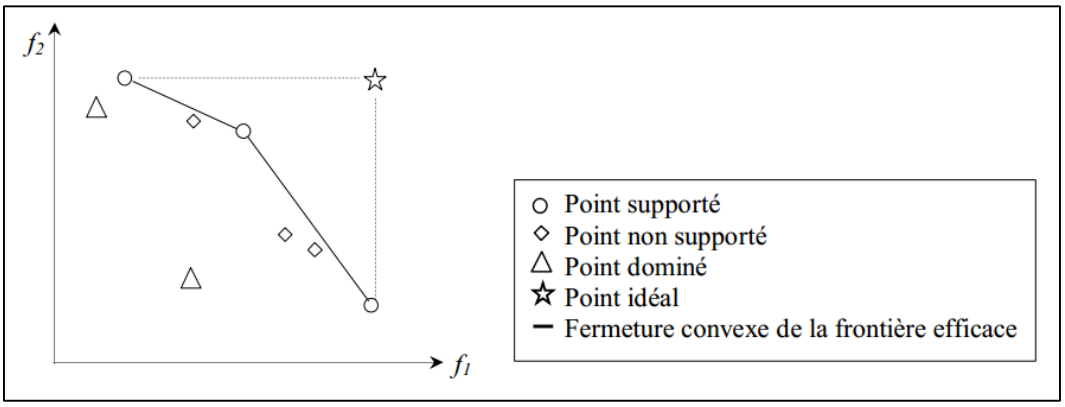



**Figure III-4 Points caractéristiques d'un problème de maximisation bi-objectif**

Les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des points du front Pareto. Les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objectif séparément. Dans Y c'est le point de coordonnées$(y_1^*, \dots, y_p^*)$, avec $y_k^* = max\ f_k(S),\ S \in\ \omega\ et\ k = 1 \dots p$. Ce point ne correspond pas à une solution réalisable car si c'était le cas, cela sous-entendrait que les objectifs ne sont pas contradictoires et qu'une solution optimisant un objectif, optimise simultanément tous les autres, ce qui ramènerait le problème à un problème ayant une seule solution Pareto optimale. L'ensemble de ces définitions est visualisé dans la figure précédente.

# 3. Méthodes de résolution

## 3.1.Méthodes Exactes

Ce sont des méthodes qui explorent l'espace de recherche de façon implicite. Parmi les plus célèbres on cite (LACOMME, 2005) **:**

– La programmation dynamique : Basée sur des relations de récurrence et recherche rétrograde de parcours dans un graphe, avec absence de retour en arrière.
– Les méthodes arborescentes de type séparation/évaluation (Branch-and-Bound).
– La programmation mathématique du type linéaire en variables mixtes, décomposée en 2 phases itératives :
  - Relaxation des contraintes d'intégrité et résolution par une méthode de simplex ou de points intérieurs.
  - Réintégration des contraintes d'intégrité et résolution par méthodes arborescentes ou des méthodes de coupe.

## 3.2.Méthodes Approchées

On retrouve deux types de méthodes approchées :

### *3.2.1.Heuristiques*

Une heuristique est un algorithme approché qui permet d'identifier en temps raisonnable au moins une solution réalisable (pas nécessairement optimale). L'utilisation d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution. Généralement, elle est conçue pour un problème particulier. Elles peuvent être classées en deux catégories :

– ***Méthodes constructives*** : qui génèrent des solutions à partir d'une solution initiale, en essayant d'ajouter des éléments petit à petit jusqu'à ce qu'une solution complète soit obtenue.
– ***Méthodes de recherche locale*** : qui démarrent d'une solution initialement complète (Probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage.

### *3.2.2.Méta-heuristiques*

Face aux problèmes rencontrés par les heuristiques pour avoir une solution réalisable de bonne qualité, les méta-heuristiques sont apparues. Une méta-heuristique peut être adaptée à différents types de problèmes alors qu'une heuristique est utilisée à un seul en particulier (LABED, 2013). La plupart des méta-heuristiques sont inspirées par des systèmes naturels dans plusieurs domaines tels que la biologie, l'éthologie, la physique, etc. Les méta-heuristiques peuvent être classées de la manière suivante :

- ***Méta-heuristique à solution unique*** : Tels que la recherche tabou (Tabu search), l'Algorithme a seuil, le recuit simulé (Simulated Annealing) ainsi que le ***V.N.S.*** (Recherche a voisinage variable / Variable Neighbourhood Search).
- ***Meta-heuristique à population de solutions :*** Tels que les Algorithmes Génétiques, les Algorithmes de colonie de fourmis ainsi que l'algorithme à évolution différentielle (***DE***).

## 4. Variable Neighbourhood Search (Recherche à Voisinages Variables)

### 4.1.Définition

Variable Neighbourhood Search (V.N.S) est une méta-heuristique introduite par HANSEN et MLADENOVIC (HANSEN & MLADENOVIC, 1999). Elle a pour but de résoudre les problèmes d'optimisation combinatoire, dont l'idée de base est le changement systématique de voisinage au sein d'une recherche locale.

Soit $L = (N(1), \dots, N(T))$ une liste finie de voisinages, où $N(t)(s)$ est l'ensemble des solutions dans le T$^{ieme}$ voisinage de $s$. Dans la plupart des méthodes de recherche locale, on a T=1. Ces voisinages interviennent de la manière suivante :

Dans le processus de recherche du V.N.S, étant donnée une solution initiale $s$, on génère une solution voisine $s'$ dans $N(1)\ (s)$ et on lui applique une procédure de recherche locale afin d'obtenir une solution $s''$. Si $f(s'') < f(s)$, alors on pose $s = s''$ et on génère une nouvelle solution voisine dans $N(1)(s)$. Sinon, la solution courante reste s et on change de voisinage en générant une solution $s'$ dans $N(2)\ (s)$. Plus généralement, on change de voisinage à chaque fois que l'un d'entre eux n'est pas parvenu, après application de la procédure de recherche locale, à améliorer la solution courante $s$. Par contre, dès qu'un voisinage permet d'améliorer $s$, alors on recommence le processus avec le premier voisinage de la liste $L$. (HANSEN & MLADENOVIC, 1999)

Le V.N.S utilise les constats suivants :

- Un minimum local par rapport à un voisinage n'en est pas nécessairement un par rapport à un autre.
- Un minimum global est un minimum local par rapport à tous les voisinages possibles.
- Pour de nombreux problèmes, les minimaux locaux par rapport à un ou à plusieurs voisinages sont relativement proches les uns des autres.

### 4.2. Fonctionnement

Soit $S$ un ensemble de solutions à un problème d'optimisation, et soit $f$ la fonction objectif. Une structure de voisinage (ou tout simplement un voisinage) est une fonction $N$ qui associe un sous-ensemble de $S$ à toute solution $s \in S$. Une solution $s' \in N(s)$ est dite voisine de $s$ (HANSEN 1999).

Une solution $s \in S$ est un minimum local relativement à la structure de voisinage $N$ si :

$$f(s) \leq f(s') \; \forall \; s' \in N(s)$$

Une solution $s \in S$ est un minimum global si :

$$f(s) \leq f(s') \; \forall \; s' \in S$$

Certaines méthodes d'optimisation, qui partent d'une solution initiale et qui l'améliorent en explorant son voisinage immédiat, présentent l'inconvénient de s'arrêter au premier minimum local trouvé.

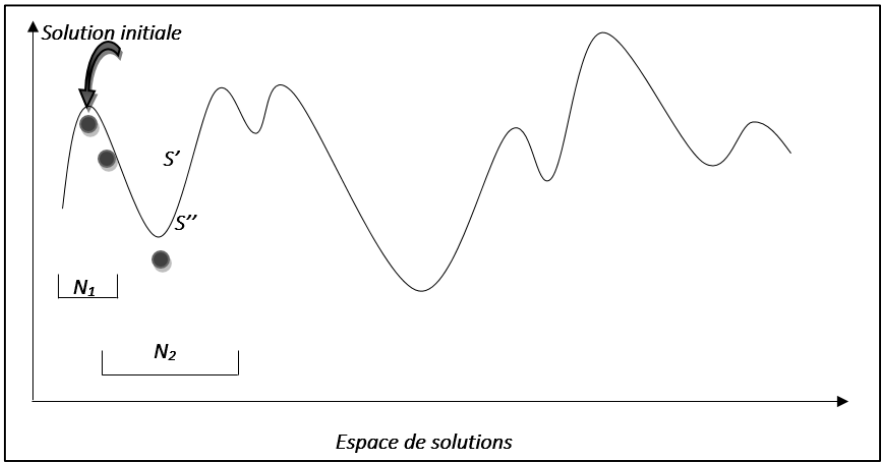



**Figure III-5: Recherche d'un minimum local**

Les méta-heuristiques contiennent souvent une technique ou une astuce permettant d'éviter de se retrouver piégé dans ces minima locaux, en explorant davantage tout l'espace des solutions, de façon à augmenter la probabilité de rencontrer le minimum optimal, c'est-à-dire le minimum global.

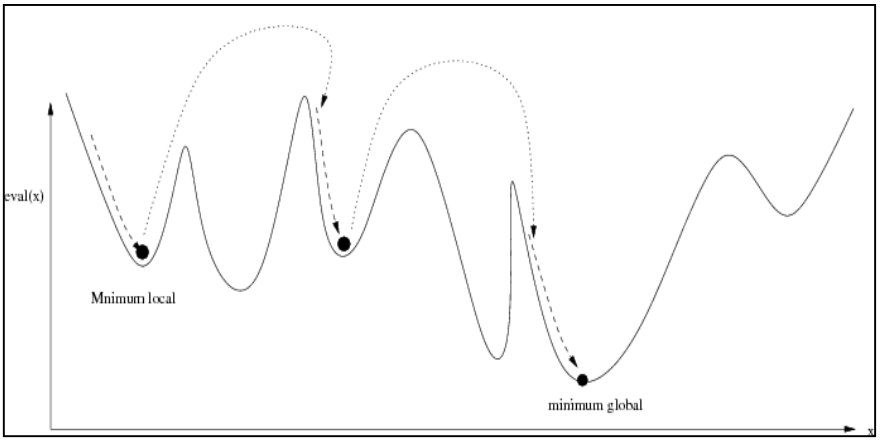



**Figure III-6: Recherche d'un minimum Global**

Dans le cadre de l'optimisation combinatoire, en pratique, on aura tout intérêt à définir le voisinage en considérant l'ensemble des modifications élémentaires que l'on peut appliquer à une solution *s* donnée, par exemple l'ensemble des permutations (si les solutions peuvent s'écrire sous la forme d'une séquence finie d'éléments, comme le cas se présente fréquemment en optimisation combinatoire)

Si cet ensemble est trop grand, on pourra toujours le réduire à un sous-ensemble, aléatoirement, ou en fonction d'un critère précis.

### 4.3.Initialisation

- On choisit un ensemble de structures de voisinages$N_k$, avec $k = 1, .. k_{max}$,
- On choisit une solution initiale x et une condition d'arrêt.

### 4.4.Principe

- Changer systématique de voisinage pour s'extraire des maximas locaux.
- Sauter d'une solution à une autre si et seulement s'il y a amélioration.
- Utiliser la recherche locale pour obtenir un minimum local.

### 4.5.Avantages

- Elle est simple à utiliser puisqu'elle est basée sur un principe simple qui nous permet de changer systématiquement de voisinage lorsqu'on se retrouve bloqué dans un minimum local.
- Etant très généraliste, elle s'applique à un grand nombre de problèmes d'optimisation combinatoire.
- Elle est facile à utiliser : les différentes étapes de l'algorithme sont faciles à comprendre et à mettre en œuvre.
- Elle est efficace : les meilleures solutions sont obtenues en un temps de calcul modéré.
- Le fait d'utiliser plusieurs voisinages permet de diversifier l'exploration de l'espace de solution afin d'accéder à un plus grand nombre de régions intéressantes, ce qui conduit à une méthode plus robuste que le recuit simulé ou la recherche tabou.

## 5. Differential Evolution Algorithm (Algorithme d'évolution différentielle)

### 5.1. Définition

L'algorithme d'évolution différentielle (noté DE dans la littérature) est une méta-heuristique appartenant à la famille des algorithmes évolutionnaires à base de populations, introduite par STORN et PRICE en 1997. Cette méthode est inspirée des Algorithmes Génétiques et des stratégies évolutionnaires. Elle a été conçue à l'origine pour les problèmes d'optimisation continus et sans contraintes.

Le D.E est un algorithme d'optimisation itératif pour lequel on fait converger une population d'individus vers des solutions proches de l'optimale. Un individu est une solution candidate à un problème multi-variable. La capacité d'un individu à répondre au problème est donnée par la valeur de sa fonction objectif.

### 5.2. Principes de l'algorithme à évolution différentielle

Dans un premier temps, un tirage aléatoire uniforme est appliqué sur l'ensemble des valeurs possibles de chaque variable, afin de générer une population initiale. Les bornes inférieures et supérieures des variables sont spécifiées selon la nature du problème. Après l'initialisation, l'algorithme effectue une série de transformations sur les individus, dans un processus appelé *évolution* (STORN & PRICE, 1997).

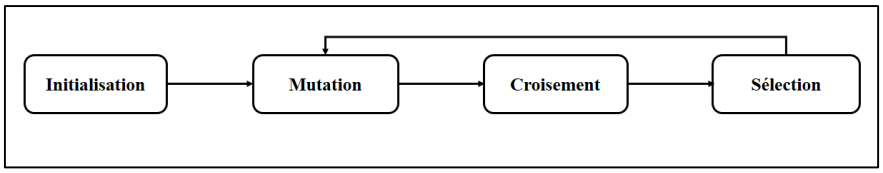



**Figure III-7 : Fonctionnement d'un D.E**

La population contient $N$ individus. Chaque individu est un vecteur de dimension $D$, où $G$ désigne la génération :

$$X_{i,G} = (X_{1i,G}, X_{2i,G}, \dots, X_{Di,G}) \text{ avec } i = 1, 2, \dots, N$$

A chaque génération, l'algorithme applique successivement trois opérations sur chaque vecteur (mutation, croisement, sélection) pour produire un vecteur d'essai (*trial vector*) :

$$U_{i,G+1} = (U_{1i,G+1}\, U_{2i,G+1}, \dots, U_{Di,G+1}) \text{ avec } i = 1, 2, \dots, N$$

Une opération de sélection permet de choisir les individus à conserver pour la nouvelle génération $(G + 1)$.

### 5.2.1. Mutation

Pour chaque vecteur courant $X_{i,G}$, on génère un vecteur mutant $V_{i,G+1}$ qui peut être crée en utilisant une des stratégies de mutation. Il existe plusieurs stratégies parmi lesquelles on peut citer :

***DE/rand/1 :*** Pour chaque vecteur $X_{i,G}$ de la génération $G$, on construit le vecteur mutant $V_{i,G+1}$ à partir de trois vecteurs $X_{r1,G}$, $X_{r2,G}$ et $X_{r3,G}$ aléatoirement choisis dans le reste de la population, tous différents et différents de $X_{i,G}$. Le facteur $F$ contrôle l'amplitude du vecteur d'exploration $(X_{r2,G} - X_{r3,G})$:

$$V_{i,G+1} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G})$$

***DE/best/1 :*** La création d'un nouvel individu $V_{i,G+1}$ est réalisée en ajoutant une perturbation au meilleur individu de la population précédente, à travers deux autres individus choisis aléatoirement.

$$V_{i,G+1} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G})$$

***DE/current to best/1 :*** Le vecteur mutant est créé à l'aide de deux vecteurs choisis au hasard, ainsi que le meilleur vecteur de la génération précédente.

$$V_{i,G+1} = X_{i,G} + F * \left(X_{r1,G} - X_{r2,G}\right) + F * (X_{best,G} - X_{i,G})$$

***DE/best/2 :*** le vecteur mutant est créé en ajoutant une perturbation au meilleur individu à travers deux différences pondérées d'individus sélectionnés aléatoirement dans le reste de la population.

$$V_{i,G+1} = X_{best,G} + F * \left(X_{r1,G} - X_{r2,G}\right) + F * (X_{r3,G} - X_{r4,G})$$

Où :

- Les indices $r1, r2, r3$ et $r4$ sont des entiers aléatoires tous différents, qui appartiennent à l'ensemble $\{1, 2, \ldots, N\}$, et ils sont également choisis différents de l'indice courant $i$.
- $X_{best,G}$ est le meilleur individu à la $G^{\text{ème}}$ génération.
- $F \in [0, 2]$ est une valeur constante, appelée « *differential weight* », qui contrôle l'amplification de la variation différentielle de $\left(X_{ri,G} - X_{rj,G}\right)$

### 5.2.2. Croisement

Après la mutation, une opération de croisement doit être appliquée pour augmenter la diversité des vecteurs de paramètres perturbés. Cette opération permet de former un vecteur d'essai final $U_{i,G+1}$, selon le vecteur $X_{i,G}$et le vecteur mutant correspondant $V_{i,G+1}$.

Le nouveau vecteur $U_{i,G+1}$ est donné par la formule suivante :

$$U_{i,G+1} = \begin{cases} V_{ji,G+1} \; si \; (rand\; j \; \leq CR) \; ou \; j = j \; rand \\ X_{ji,G} \; sinon \end{cases}$$

- $rand\; j$ est la $j^{\text{ême}}$ valeur procurée un générateur de nombre aléatoire uniforme appartenant à l'intervalle [0,1].
- $CR$ est le coefficient de croisement, déterminé par l'utilisateur et appartient à l'intervalle [0,1].

- $j\,rand$ est un indice choisi au hasard dans l'ensemble $\{1, 2, \ldots, N\}$.

### 5.2.3. Sélection

Après le croisement, un opérateur de sélection est introduit pour décider quel vecteur, parmi $U_{i,G+1}$ ou $X_{i,G}$ doit être choisi dans la génération $G + 1$.

On garde le vecteur ayant la plus petite valeur de fonction objectif en cas de minimisation. Un nouveau vecteur est généré selon l'expression suivante :

$$U_{i,G+1} = \begin{cases} U_{i,G+1}\ si\ f\left(U_{i,G+1}\right) < f(X_{i,G}) \\ X_{i,G}\ sinon \end{cases}$$

Il est clair qu'un bon réglage des principaux paramètres de l'algorithme (taille de la population $N$, facteur de mutation $F$ et facteur de croisement $CR$) contribue de façon importante à l'efficacité de l'algorithme.

## Conclusion

Dans ce chapitre, nous avons présenté les méthodes d'optimisation Multi-Objectif, et plus précisément les approches Pareto. Nous avons aussi énuméré les méthodes de résolution des problèmes d'optimisation. Nous avons détaillé les méta-heuristique V.N.S et D.E qui seront adaptées à notre problème dans le prochain chapitre.

# Chapitre IV:.Implémentation, tests et résultats.

# Introduction

Lors de la réalisation de cette étude, nous avons été emmenés à concevoir une application permettant de résoudre le problème d'ordonnancement intégré des tâches de production et de maintenance. Dans ce chapitre nous présenterons l'adaptation des deux méta-heuristiques (DE, VNS) du mono au bi-objectif ainsi que leur implémentation. Il est à noter que le V.N.S implémenté est à base de populations, alors que le V.N.S classique est à base de solution unique. Nous pouvons donc dire que cette méthode est un V.N.S hybride.

Nous allons aussi mettre en valeur l'application conçue ainsi que les tests effectués et les résultats obtenus, pour mieux voir le comportement des deux méthodes vis-à-vis de leurs paramètres.

# 1. Implémentation

## 1.1. Plateforme et outils de développement

Notre application a été réalisée en utilisant le langage de programmation *C++* sous le système d'exploitation *WINDOWS 7*, et l'IDE *VISUAL STUDIO 2012*. L'interface graphique a été conçue par *QT 5.2.1*. Les tests ont été effectués sur un processeur *Intel Core i5*, avec 2.6 GHz et 4 Go de mémoire RAM.

## 1.2. Codage utilisé

***Codage de la solution***

L'ordonnancement des taches de productions est représenté par un vecteur d'entier ou chaque élément désigne l'indice de la tâche à exécuter. L'ordre d'exécution des tâches est défini par séquence définie sur ce vecteur.

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 5 | 3 | 4 |

**Tableau IV-1 : Représentation d'une solution ou le nombre de tâches de production n=5**

***Tâches de maintenance préventive***

Les tâches de maintenance préventive sont représentées par un matrice $(m * 2)$ ou $m$ est le nombre de machines.

Dans le tableau suivant (exemple de la machine $M_1$), la date de MP signifie qu'après chaque 6 unités de temps de production sur la machine on prévoit de faire une MP durant 4 unités de temps.

| Machines | Période MP prévue | Durée MP |
|---|---|---|
| $M_1$ | **6** | **4** |
| $M_2$ | **5** | **3** |
| $M_3$ | **4** | **4** |

**Tableau IV-2 : Représentation des tâches de maintenance préventive (MP)**

***Stratégie d'affectation des tâches de production et de maintenance aux machines***

Les taches de production sont affectées une à une selon leur ordonnancement a la machine la plus disponible tout en appliquant la stratégie rationnelle pour inclure les taches de maintenance.

```
Pour i=1 jusqu'à N faire
      Recherche la machine la plus disponible ;
      Si ((date de disponibilité de la machine + durée de la tâche de production)
      - date de MP de la machine) > (date de MP de la machine – date de
      disponibilité de la machine))
            Intégrer la tâche de maintenance ;
            date disponibilité de la machine = date de disponibilité de la
            machine + durée de la maintenance ;
      Sinon
            Exécuter la tâche de production ;
            date de disponibilité de la machine = date de disponibilité de la
            machine + durée de la tache de production ;
      Fin Si
Fin pour
```

**Algorithme IV-1: Algorithme d'affectation des tâches de production et de maintenance préventive**

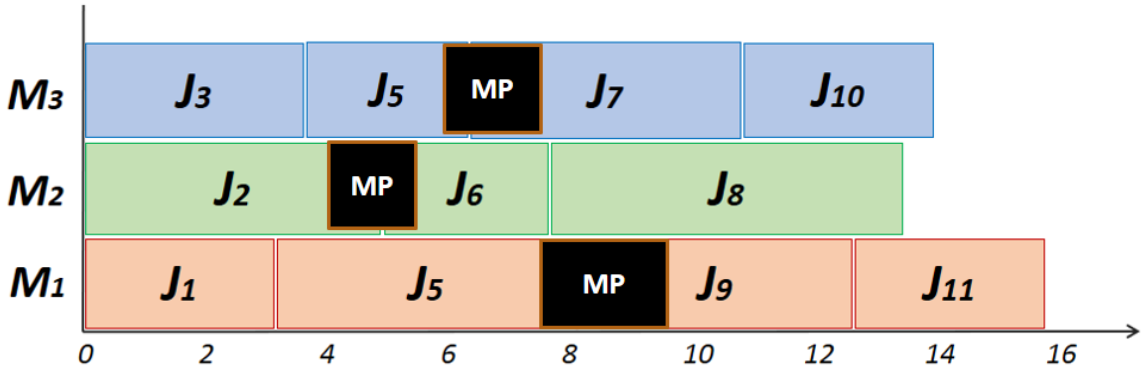



**Figure IV-1: Exemple d'ordonnancement simultané des tâches de maintenance et de production**

***Stratégie de sélection de solution***

Pour la sélection de la solution nous avons implémenté un mécanisme permettant de choisir une solution, même si aucune des deux ne domine l'autre.

Puisque notre problème est un problème bi-objectif si aucune des deux solutions ne domine l'autre donc une est meilleure que l'autre dans un critère et moins bien dans l'autre.

Nous avons donc choisi de prendre la solution ou la différence entre les valeurs des critères est grande. (Voir exemple Tableau IV-3)

```
Si (solution 1 > solution 2)
      Alors solution 1 domine solution2 ; sélectionner solution1 ;
Sinon Si (solution 2 > solution 1)
      sélectionner solution 2 ;
      Sinon
      Sélectionne solution celle qui domine le plus dans sur un
      critère ;
      Fin Si
Fin Si
```

**Algorithme IV-2: Algorithme de sélection des solutions**

Sur l'exemple énoncé dans le tableau suivant, nous avons choisi la solution 1 car $(8-5) > (7-6)$.

| *Critères* | $F_1$ | $F_2$ |
|---|---|---|
| ***Solution 1*** | **8** | **7** |
| ***Solution 2*** | **5** | **6** |

**Tableau IV-3 : Exemple de solutions ou aucune ne domine l'autre**

### 1.3. Variable Neighbourhood Search

Suite à nos travaux précédents, nous avions eu comme perspective d'implémenter un VNS hybride, qui applique ses mécanismes sur une population de solutions au lieu d'une solution unique.

Lors de ce travail, nous avons mis en pratique cette perspective, tout en implémentant un VNS qui résout un problème bi-objectif d'ordonnancement intégré. Pour cela, nous nous somme basés sur les algorithmes suivants.

***Algorithme de génération de population initiale***

```
Algorithme de génération de la population initiale

DEBUT
      Fixer la taille de la population initiale (nbpop) ;
      Pour i de 0 à nbpop faire
                Générer une solution ;
                Stocker la solution ;
      Fin Pour
FIN
```

**Algorithme IV-3 : Génération de population initiale**

***Algorithmes de voisinage***

```
Algorithme du 1er voisinage :
DEBUT
      Générer un nombre aléatoirement (a) entre [0, taille/2].
      Générer un nombre aléatoirement (b) entre [taille/2, taille].
      tmp = solution[a].
      solution [a] = solution[b].
      solution [b] = tmp.
FIN
```

**Algorithme IV-4: Algorithme du premier voisinage**

```
Algorithme du 2ème  voisinage :
DEBUT
      Générer un nombre aléatoire (a) entre [1,taille] ;
      Pour i de 0 à (a) FAIRE
         Générer un nombre aléatoire (b) entre [0,a].
         temp = solution[i] ;
         solution [i]= solution[b] ;
         solution [b]=temp ;
      Fin Pour
FIN
```

**Algorithme IV-5 : Algorithme du deuxième voisinage**

```
Algorithme du 3ème voisinage :
DEBUT
       Pour i de 0 à taille Faire
              solution2 [i] = solution[i];
       Fin Pour
       Générer un nombre aléatoire (a) entre [1,taille].
       j=0; i=a;
       Tant Que (i>=0)
              solution[j] = solution2[i];
              j++;
              i--;
       Fin Tan Que
       Pour b de(a+1) à taille FAIRE
       DEBUT
              Solution [j] = solution2 [b];
              j++;
       FIN
FIN
```

**Algorithme IV-6 : Algorithme du troisième voisinage**

## ***Algorithme de résolution à base de V.N.S***

```
Algorithme de résolution :
DEBUT TQ(Indiceitération<Itérationmax)
  DEBUT
       kmax =3 (nbr de voisinages).
       K=1.
       TQ(k<=kmax)
       DEBUT
              Cas de k Parmi :
              Cas 1 :
              DEBUT
                     Chercher une solution dans le 1 er voisinage.
                     SI (SolutionInitiale < NouvelleSolution)
                     DEBUT
                            SolutionInitiale = NouvelleSolution.
                            k=1.
                     SINON
                            k=k+1.
                     FIN
              Cas 2 :
              DEBUT
                     Chercher une solution dans le 2eme voisinage.
                     SI (SolutionInitiale < NouvelleSolution)
              DEBUT
                            SolutionInitiale = NouvelleSolution.
                            k=1.
                     SINON
                            k=k+1.
                     FIN
              Cas 3 :
              DEBUT
                     Chercher une solution dans le 3eme voisinage.
                     SI (SolutionInitiale < NouvelleSolution)
                     DEBUT
                            SolutionInitiale = NouvelleSolution.
                            k=1.
                     SINON
                            k=k+1.
                     FIN
       FIN
       Indiceitération= Indiceitération+1
  FIN
    Sélectionner les meilleurs individus // front pareto

FIN
```

**Algorithme IV-7: Algorithme de résolution à base de V.N.S**

### 1.4. Differential Evolution

Comme nous avons défini dans le chapitre précédant, le D.E est une méthode adaptée aux problèmes d'optimisation mono-objectif et continus. La difficulté de notre travail réside sur le fait qu'il fallait adapter le D.E aux problèmes d'optimisation bi-objectif avec variables discrètes. Pour cela, nous avons implémenté l'algorithmes de résolution à base de D.E suivant.

```
Algorithme de résolution :

Entrées : Taille De la popultaion NP,Nombre de genration NB_GEN,
Dimension du probleme D , Taux de croisement CR, Poids differentiel
F ;

Var : I,Jrand,G;

Debut

      G=0;

      Créer aléatoirement une population initiale xi,G ;

      Pour  G=1 à NB_GEN faire

            Pour  i =1 a NP faire

                  Selectioner aleatoirement les indices r1≠r2 ≠r3 ;

                  Jrand=randint(1,D);

                  Randj=rand(0,1);

                  Pour j=1 à D faire

                        Si(randj<= CR ou j=jrand) alors

                              Uij.G+1=Xr1j.G+F.(Xr2j.G-Xr3j.G) ;

                        Sinon

                              Uij,G+1=Xij,G ;

                        Finsi

                  Finpour

                  Evaluer la fonction fitness f(ui,G+1) ;

                  Si f(Ui,G+1)<f(Xi,G) alors // selection

                        Xi,G+1 = Ui,G+1 ;

                  Sinon

                        Xi,G+1 = Xi,G ;

                  Finsi

            Finpour

Sélection les meilleurs individus // front pareto ;

G=G+1 ;

      Finpour

Fin
```

**Algorithme IV-8: Algorithme de résolution à base de DE**

## 1.5. Interface graphique de l'application réalisée

L'interface conçue contient quatre fenêtres différentes. La première est une page d'accueil, la seconde est celle de la résolution par D.E. La troisième fenêtre contient la résolution par VNS et enfin la quatrième compare les résultats du D.E avec ceux du VNS.


**Figure IV-2 : Fenêtre d'accueil de l'application**

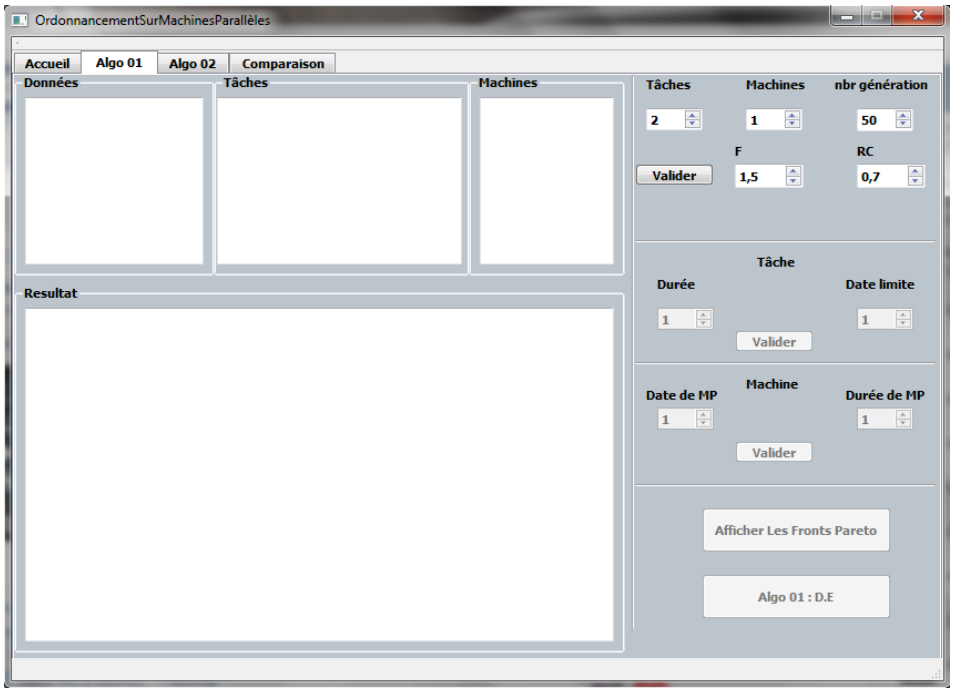



**Figure IV-3 : Fenêtre de résolution par D.E**

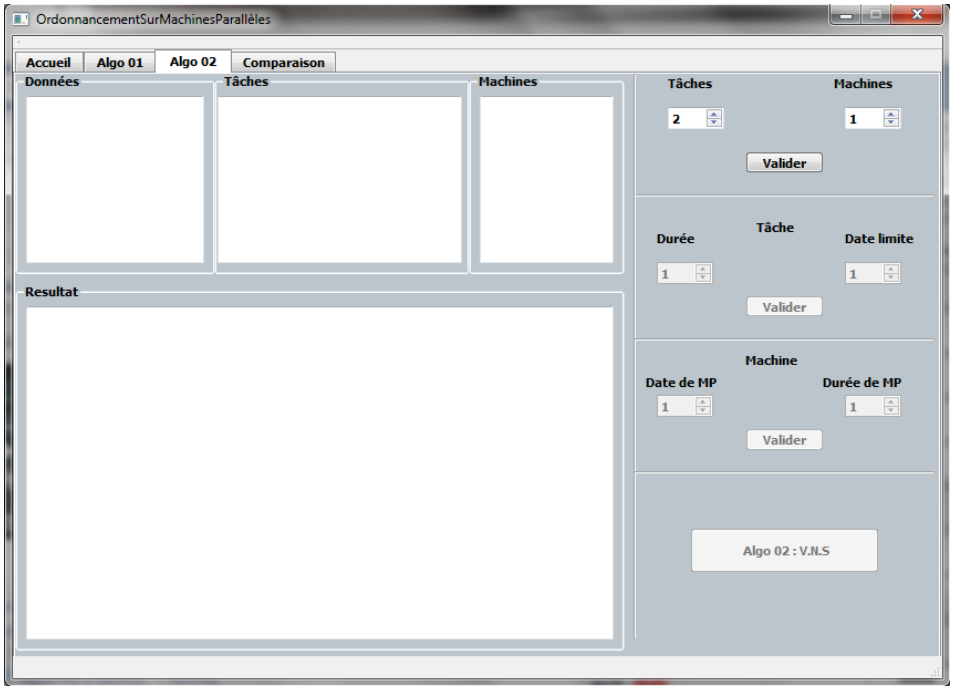



**Figure IV-4 : Fenêtre de résolution par V.N.S**

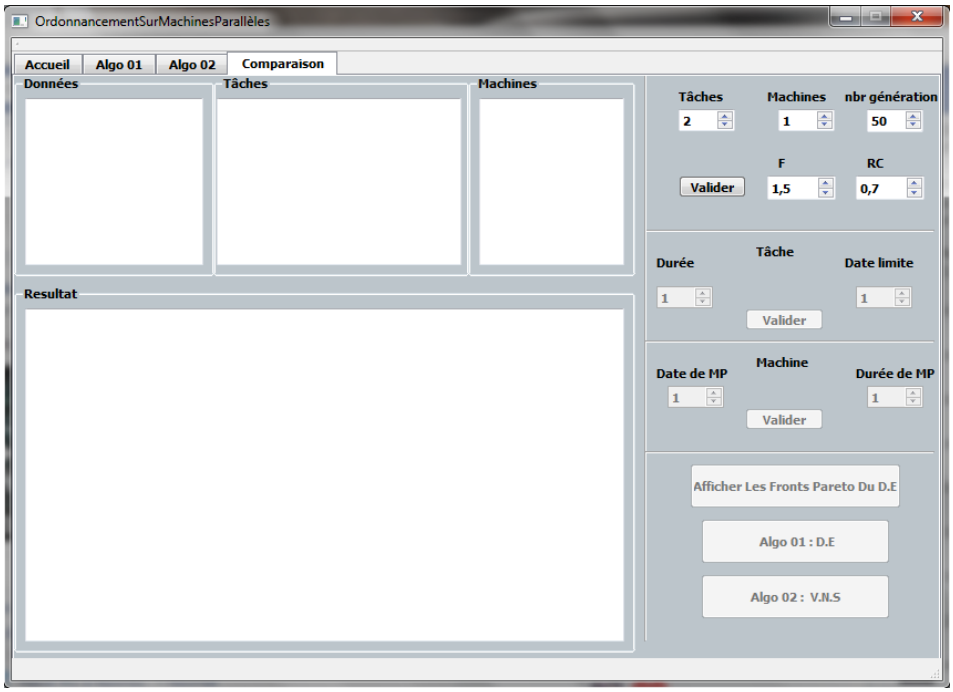
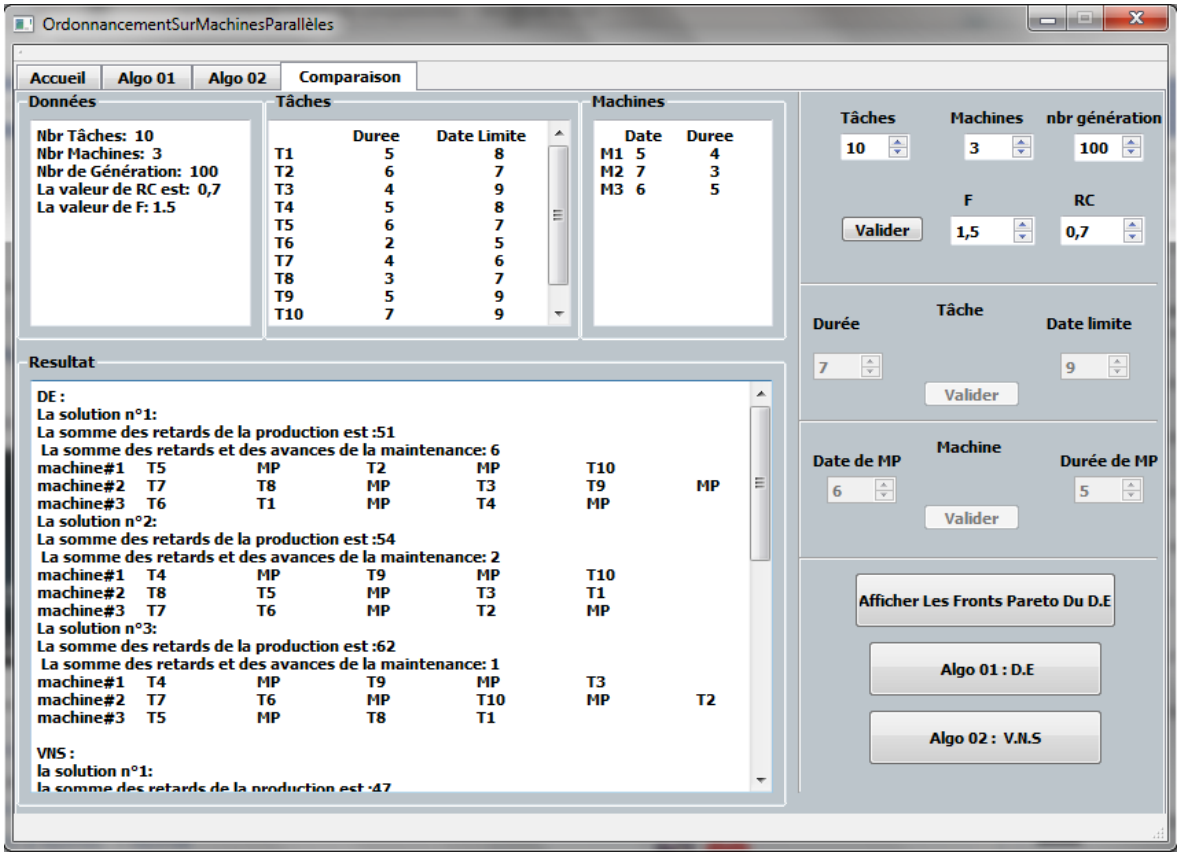



**Figure IV-5 : Fenêtre de comparaison entre les résultats des deux algorithmes**



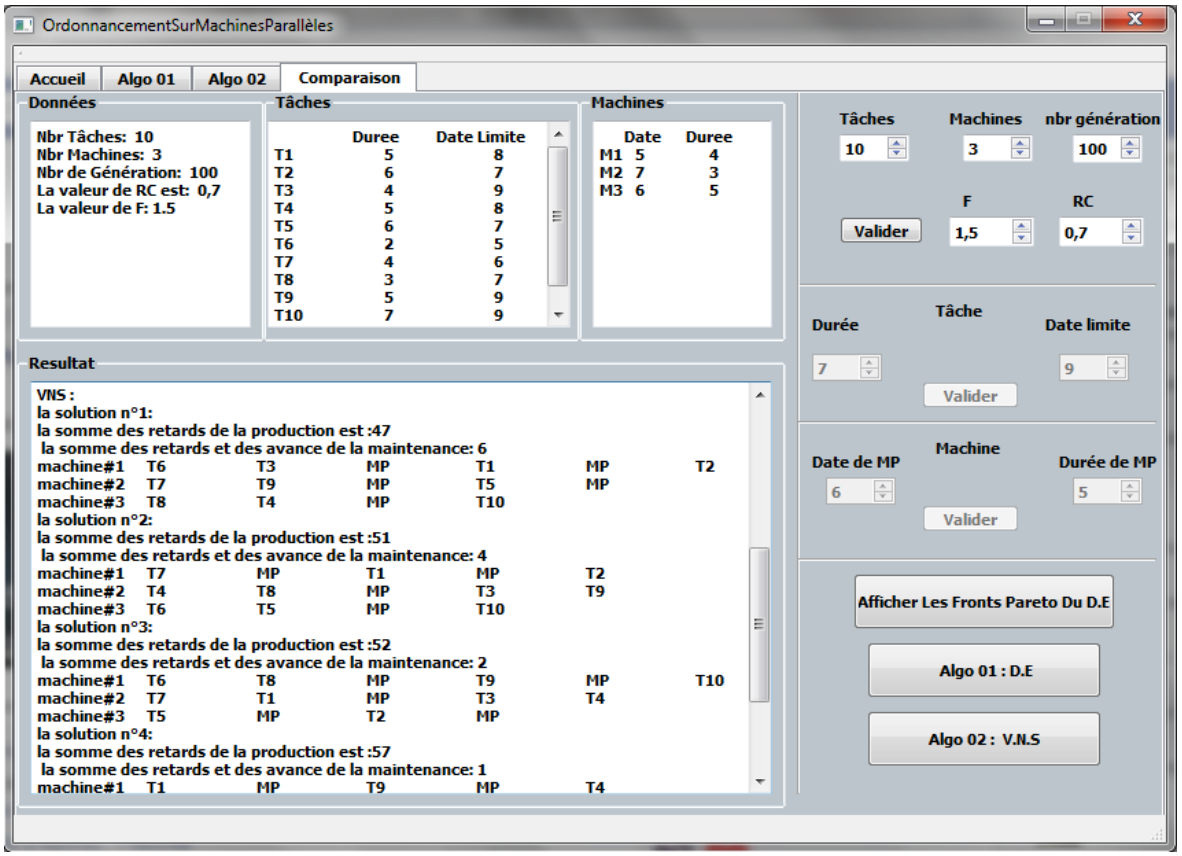




**Figure IV-6: Exemple de comparaison entre les solutions des deux algorithmes**

## 2. Tests expérimentaux

Nous allons illustrer dans ce paragraphe l'influence de quelques paramètres principaux des deux méthodes utilisées (V.N.S et D.E). Nous avons choisi de faire plusieurs tests sur les instances. Nous avons modifié des paramètres, mais les changements varient d'un test à l'autre. Ensuite nous avons collecté et présenté les résultats par des graphes appropriés.

### 2.1. Paramétrage des algorithmes

***Variable Neighbourhood Search***

Afin de fixer la taille de population de notre V.N.S nous avons exécuté des instances et tracé la frontière Pareto pour chaque population initiale proposée. La taille de la population varie entre 10 et 100 individus.

Pour chaque taille de population on exécute des instances. Chaque instance est exécutée cinq fois.

La figure suivante représente les résultats de ce test.

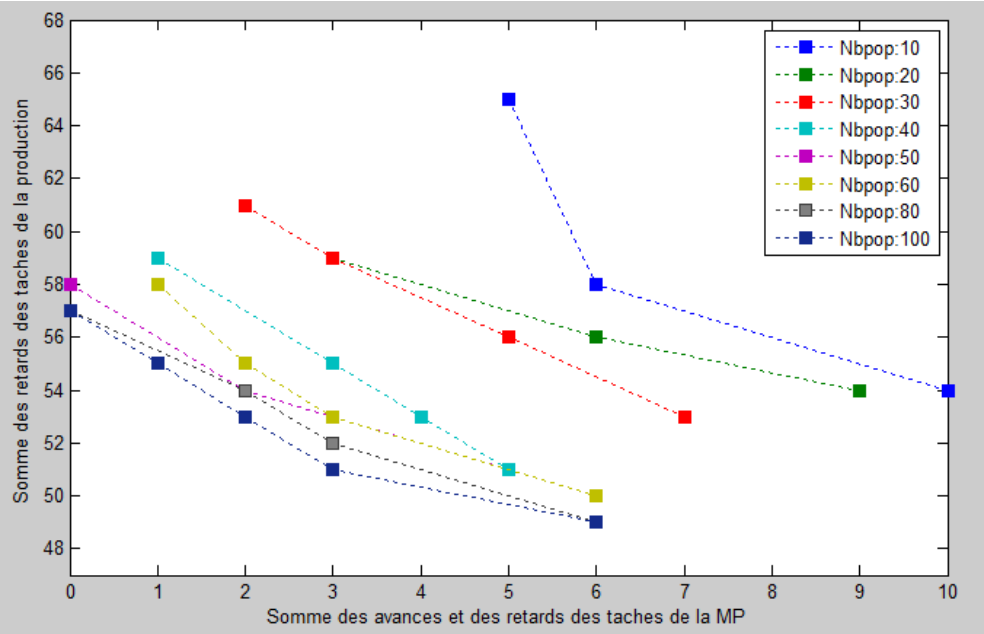



**Figure IV-7 : Résultats du test de paramétrage pour le V.N.S**

Nous constatons que les meilleurs résultats sont obtenus à partir de population initiale de 100 individus.

Nous concluons qu'en prenant une population de 100 individus, nous convergerons rapidement vers des solutions de bonne qualité.

***Differential Evolution Algorithm***

Afin de paramétrer notre algorithme d'Evolution Différentielle, nous devons mettre au point les meilleures valeurs pour les coefficients de croisement CR et de poids différentiel (*differential Weight*) F. Mais avant cela, nous devons fixer la taille de population initiale ainsi que le nombre de générations

Tout d'abord, nous fixons la taille de population initiale (figure suivante).

Nous constatons que les meilleurs résultats sont obtenus pour une population initiale de 100 individus.

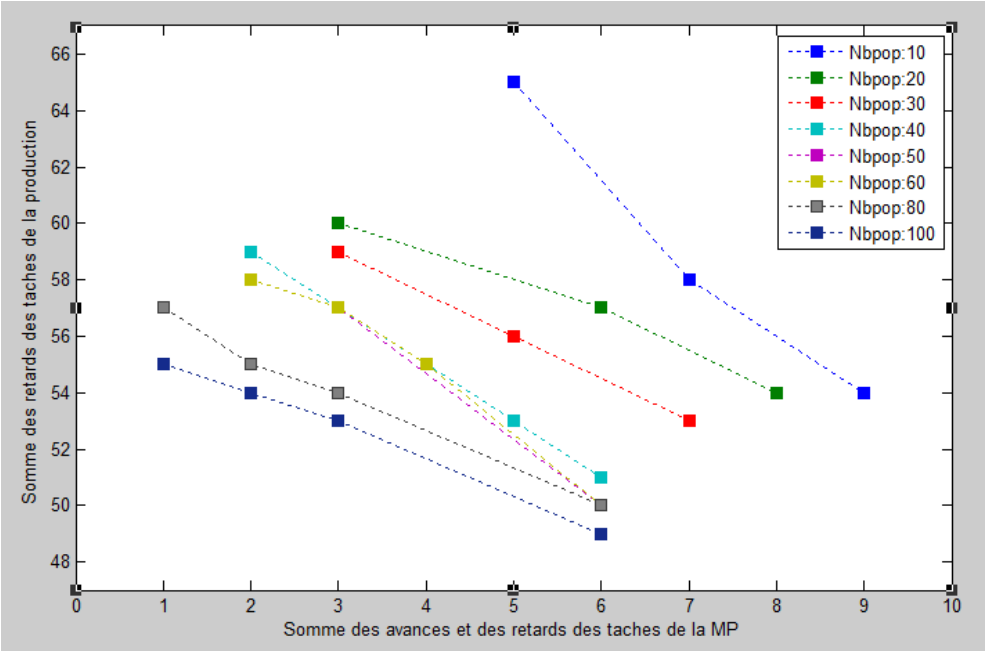



**Figure IV-8 : Taille de population initiale du D.E**

Nous concluons qu'en prenant une population de 100 individus, nous convergerons plus rapidement vers des solutions de bonne qualité.

Nous fixons maintenant le nombre de générations du D.E (Figure IV-9).

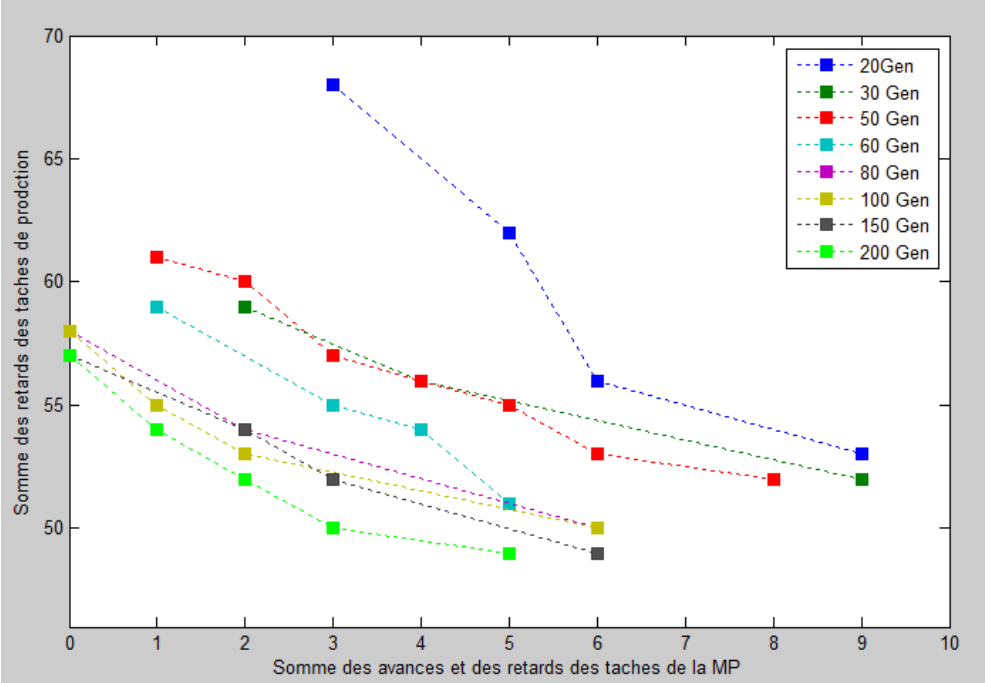



**Figure IV-9 : Tests sur le nombre de générations du D.E**

Nous remarquons que les meilleures solutions se situent pour un nombre de générations de 200.

Nous avons décidé de prendre un nombre de générations de 150 car elle est faiblement dominée par le nombre de générations 200. Et cela dans le souci de diminuer le temps de calcul de notre algorithme.

Enfin nous allons fixer les coefficient $CR$ et $F$. Pour cela nous avons fait 3 tests. Nous avons pris à chaque fois un coefficient de $CR$ et on la comparé avec trois coefficients $F$ différents (0.3, 0.9 et 1.5). Les coefficients de $CR$ testés sont 0.4, 0.7 et 0.9.

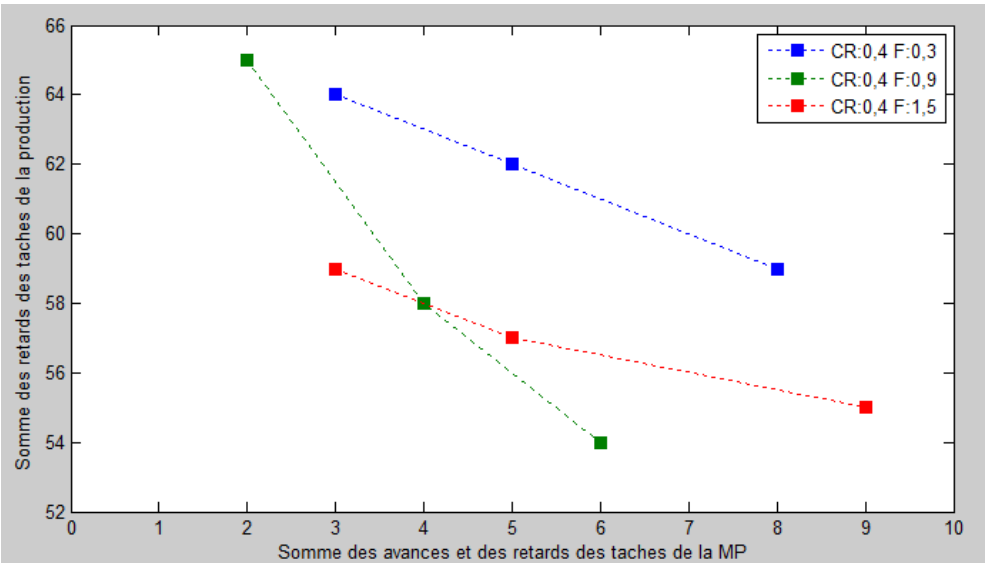



**Figure IV-10 : Tests sur les coefficients CR et F (1)**

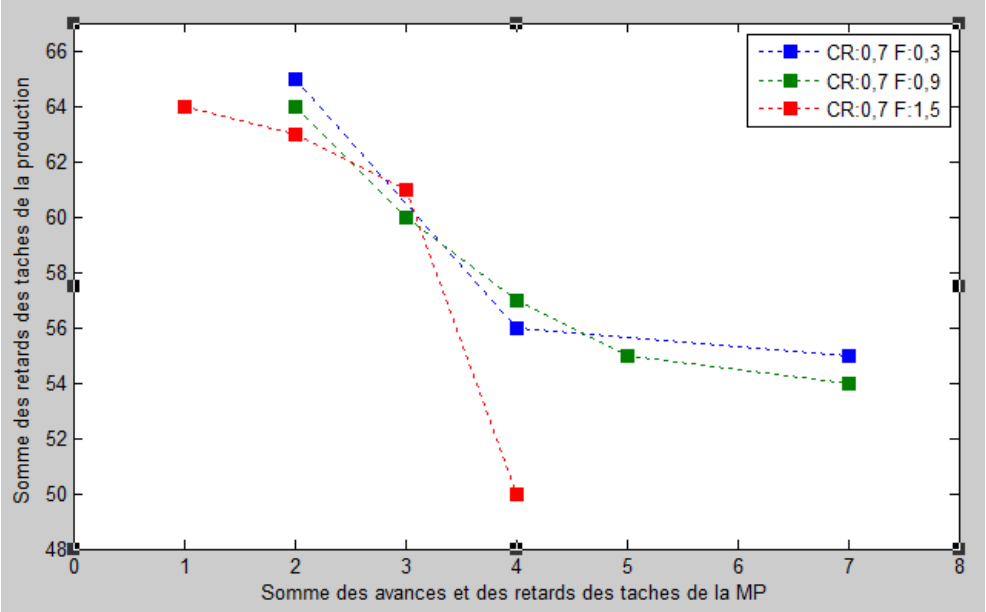



**Figure IV-11 : Tests sur les coefficients CR et F (2)**

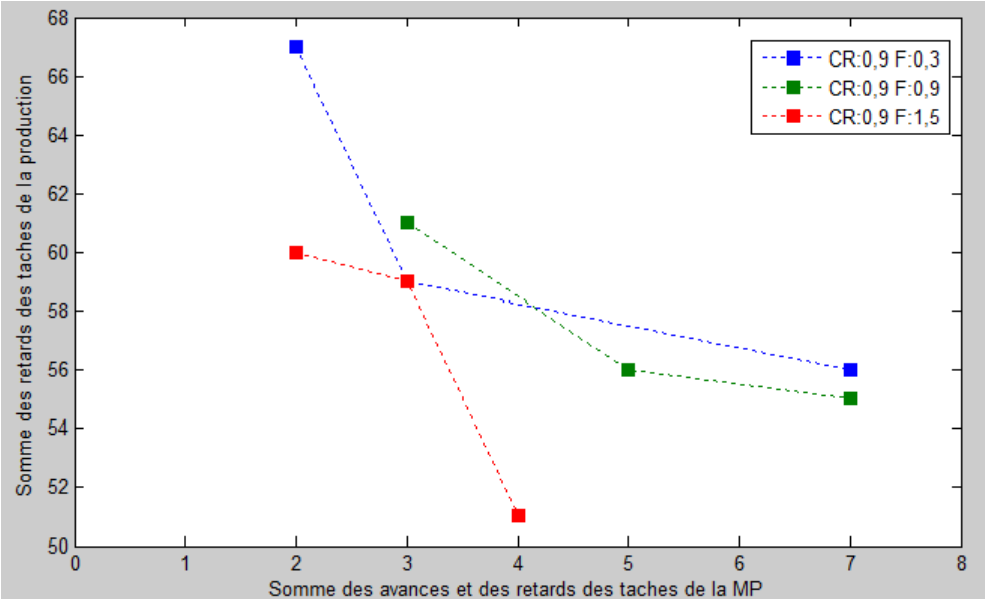



**Figure IV-12 : Tests sur les coefficients CR et F (3)**

Nous remarquons qu'à chaque fois que le $CR$ et $F$ augmentent, les solutions sont de moins en moins dominés au sens de Pareto.

Nous concluons donc que les meilleures valeurs sont $CR = 0.9$ et $F = 1.5$ .

### 2.2. Comparaison entre V.N.S et D.E

Pour pouvoir comparer entre la qualité des solutions émises par V.N.S et D.E nous avons fait appel à la métrique C. C'est une mesure relative, introduite par Zitzler (ZITZLER, 1999), qui permet de différencier clairement entre deux fronts A et B quand d'autres métriques donnent

des valeurs très similaires. La valeur de C (A, B) représente le pourcentage de solutions du front B dominées par au moins une solution du front A.

La valeur $C\ (A,B)\ =\ 1$ veut dire que toutes les solutions du front B sont dominées par les solutions du front A. A l'opposé, $C\ (A,B)\ =\ 0$, représente la situation où aucune solution dans B n'est dominée par le front A. Donc, plus proche la valeur de $C\ (A,B)$ à 1, meilleur est le front A par rapport au front B. Puisque cette métrique n'est pas symétrique, $C\ (A,B)\ \neq\ 1\ -\ C\ (A,B)$, il est nécessaire de calculer aussi $C\ (B,A)$. Par conséquent, un front A est meilleur qu'un front B si $C\ (A,B)\ >\ C\ (B,A)$.

La tableau IV-4 donne les meilleurs, les moyennes et les plus mauvaises valeurs de la métrique $C$ ($C$ (D.E, V.N.S) et $C$ (V.N.S, D.E)) obtenues en fonction des tailles des problèmes (nombre de machines, nombre de tâches) sur 10 instances aléatoires.

| | DE | | | VNS | | |
|---|---|---|---|---|---|---|
| **Problème(m,n)** | **Meill.** | **Moy.** | **Mauv.** | **Meill.** | **Moy.** | **Mau.** |
| **(2.5)** | 1 | 0.23 | 0 | 1 | 0.30 | 0 |
| **(2,10)** | 0,66 | 0.17 | 0 | 1 | 0.62 | 0 |
| **(3,10)** | 0.66 | 0.25 | 0 | 1 | 0.71 | 0 |
| **(3,20)** | 0,5 | 0,16 | 0 | 1 | 0,5 | 0 |
| **(4,40)** | 0.5 | 0.17 | 0 | 1 | 0,57 | 0,2 |
| **(5,50)** | 0,33 | 0,16 | 0 | 1 | 0,57 | 0,33 |
| **(8,50)** | 0,33 | 0,10 | 0 | 1 | 0,72 | 0,33 |
| **(8,60)** | 0,25 | 0,09 | 0 | 1 | 0,74 | 0,33 |

**Tableau IV-4 : Résultats obtenus en utilisant la métrique C**

Pour chaque problème test, on peut voir à partir de cette table que le V.N.S domine à chaque fois au moins une fois le D.E pour chaque problème test. Le D.E domine complètement au moins une fois le V.N.S lors du problème test à 2 machines et 5 tâches de production. Nous remarquons aussi que le V.N.S a une moyenne supérieure ou égale à 50% de probabilité de dominance complète sur le D.E sauf pour le cas où le problème est de taille (2,5). De son côté le V.N.S à une moyenne ne dépassant pas 25%.

La tableau IV-5 donne les meilleurs, les moyennes et les plus mauvaises valeurs concernant le nombre de solutions générées par les deux méthodes.

Nous remarquons que lorsque le problème ne dépasse pas la taille (4,40) la frontière Pareto du D.E donne en moyenne plus de résultats que celle du V.N.S. Cela dit lorsque la taille du problème augmente, le V.N.S donne plus de résultats en moyenne que le D.E.

| | DE | | | VNS | | |
|---|---|---|---|---|---|---|
| **Problème(m,n)** | **Meill.** | **Moy.** | **Mauv.** | **Meill.** | **Moy.** | **Mauv.** |
| **(2.5)** | 4 | 3,4 | 2 | 4 | 2,8 | 2 |
| **(2,10)** | 4 | 3,2 | 2 | 4 | 2,8 | 2 |
| **(3,10)** | 4 | 2,8 | 2 | 5 | 2,6 | 2 |
| **(3,20)** | 4 | 3 | 2 | 3 | 2,8 | 2 |
| **(4,40)** | 5 | 3,8 | 2 | 4 | 3 | 1 |
| **(5,50)** | 5 | 3,4 | 2 | 6 | 4,2 | 2 |
| **(8,50)** | 6 | 3,2 | 2 | 6 | 4 | 2 |
| **(8,60)** | 6 | 3,4 | 1 | 5 | 3,6 | 2 |

**Tableau IV-5 : Nombres de solutions obtenues par D.E et V.N.S**

Nous concluons que le V.N.S à une plus grande probabilité de donner de meilleurs résultats que le D.E. Concernant le nombre de solutions générés, nous ne pouvons émettre un jugement entre les deux algorithmes, car dans ce cas de figure, tout dépend si le décideur désire avoir plusieurs ou bien peu de possibilités pour choisir une solution.

Donc le V.N.S est meilleur que le D.E en termes de qualité de solutions générées.

## Conclusion

Les tests expérimentaux effectués nous ont permis de conclure que si le problème est de taille très réduite, les résultats du V.N.S et le D.E sont de même niveau. Cela dit, si le problème augmente, le V.N.S donnent de bien meilleurs résultats

# Conclusion générale

Le sujet de notre mémoire consiste en l'étude et la résolution du problème d'ordonnancement conjoint intégré Production/Maintenance.

Dans le premier chapitre nous avons abordé les systèmes de production, les problèmes d'ordonnancement en présentant leurs caractéristiques ainsi que la gestion de la maintenance.

Ensuite dans le second, nous avons présenté un état de l'art sur le problème d'ordonnancement conjoint production/maintenance en général et l'ordonnancement intégré spécifiquement. Nous avons choisi d'étudier un problème d'ordonnancement conjoint production/maintenance sur machines parallèles identiques, où la préemption des tâches de production n'est pas permise. Nous avons pris en compte deux critères de performance à optimiser : la somme des retards des tâches de production, et la somme des avances et des retards pour l'aspect maintenance. Le modèle obtenu est un modèle bi-objectif sujet à des contraintes de production et de maintenance.

Dans le troisième chapitre nous avons présenté les différentes approches de résolution des problèmes multi-objectif. Ensuite nous nous sommes focalisés sur deux méta-heuristiques. L'une basée sur la recherche à voisinages variable et l'autre sur l'évolution différentielle.

Enfin le quatrième chapitre a été consacré à l'élaboration et l'implémentation de notre application. Nous avons effectué des tests expérimentaux pour la configuration des paramètres des deux approches et aussi afin de comparer leur efficacité.

Les tests expérimentaux montrent que le V.N.S et meilleur comparé au D.E si on prend comme critère la qualité des solutions. Cela dit, ils sont relativement équivalent concernant le nombre de solutions obtenues.

Nous envisageons, de faire des travaux de recherche sur l'optimisation multi-objectif à l'avenir.

# Références


AFNOR. (2010). *Norme Française de maintenance industrielle AFNOR NF EN 13306.*

AGGOUNE, R. (2002). *Ordonnancement d'ateliers sous contraintes de disponibilité des machines.* Thèse de Doctorat. Université de Metz (France).

BARICHARD, V., & HAO, J. K. (2003). Une approche hybride pour l'optimisation multi-objectif sous contraintes. *Techniques et Sciences Informatiques*, 33-46.

BELKAID, F., SARI, Z., & SOUIER, M. (2013). A Genetic Algorithm for the Parallel Machine Scheduling Problem with Consumable Resources. *International Journal of Applied Metaheuristic Computing (IJAMC)*, 17-30.

BENBOUZID SI-TAYEB, F. (2005). *Contribution à l'étude de la performance et de la robustesse des ordonnancements conjoints production/maintenance - cas du flow-shop.* Thèse de doctorat.

BERRICHI, A., & YALAOUI, F. (2013). Efficient bi-objective ant colony approach to minimize total tardiness and system unavailability for a parallel machine scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 2295-2310.

BERRICHI, A., AMODEO, L., YALAOUI, F., & MEZGHICHE, M. (2010). Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. *Computer & Operation Research*, 1584-1596.

BERRICHI, A., AMODEO, L., YALAOUI, F., CHATELET, E., & MEZGHICHE., M. (2009). Bi-Objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem. *Journal of Intelligent Manufacturing*, 389-400.

CASSADY, C. R., & KUTANOGLU, E. (2003). Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. *IIE Trans*, 503-513.

COELLO COELLO, C. A., CHRISTIANSEN, A. D., & AGUIRRE, A. H. (1998). Using a new GA-based multiobjective optimization technique for the design of robot arms. *Robotica*, 401-414.

COLETTE, Y., & SIARRY, P. (2002). *Optimisation Multi-Objectif.* EYROLLES.

DOUMEINGTS, G. (1984). *Méthode GRAI : méthode de conception des systèmes en productique.* Thèse d'état soutenue à l'université de Bordeaux I (France).

DUPAS, R. (2004). *Amélioration de performance des systèmes de production: Apport des Algorithmes Evolutionnaires aux problèmes d'ordonnancement cycliques et flexibles.* Habilitation à diriger des recherches présentée à l'Université d'Artois (France).

FLIPO-DHAENES, C. (1998). *Optimisation d'un réseau de production et de distribution.* Thèse de Dotorat, INPG de Grenoble (France).

FOURMAN, M. P. (1985). Compaction of Symbolic Layout Using Genetic Algorithms. *ICGA*, 141-153.

GIARD, N. (1988). *Gestion de la production.* Economica.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., & RINOOY KAN, A. H. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 287–326.

HANSEN, P., & MLADENOVIC, N. (1999). An Introduction to Variable Neighborhood Search. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, 433-458.

HAO, J. K., GALINIER, P., & HABIB, M. (1999). Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*.

ISHIBUCHI, H., & MURATA, T. (1998). A multi-objective genetc local search algorithm and its applications to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 392-403.

KAABI, J. (2004). *Contribution à l'ordonnancement des activités de maintenance dans les systèmes de production.* Thèse de doctorat soutenue à l'Université de Franche Comté (France).

KAABI, J., VARNIER, C., & ZERHOUNI, N. (2002). Heuristics for scheduling maintenance and production on a single machine. *IEEE Conference on Systems, Man and Cybernetics*.

KAABI, J., VARNIER, C., & ZERHOUNI, N. (2003). *Genetic algorithm for scheduling production and maintenance in a Flow Shop.* Laboratoire d'automatique de Besançon (France).

LABED, S. (2013). *Méthodes bio-inspirées hybrides pour la résolution de problèmes complexes.* Thèse de Doctorat soutenue à l'Université de Constantine (Algérie).

LACOMME, P. (2005). *Méthodes exactes et approchées pour l'optimisation des systèmes à moyen de transport.* Habilitation à diriger des recherches présentée à Clermont-Ferrand (France).

LEE, C. Y. (1996). Machine scheduling with an availability constraint. *Journal of Global Optimization*, 395-416.

LEE, C. Y. (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics*.

LEUNG, J. Y.-T. (2004). *Handbook of Scheduling : Algorithms, Models and Performance analysis.* Chapman & Hall/CRC Computer and information sciences series.

LOPEZ, P. (1991). *Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources.* Thèse de Doctorat soutenue le 23 Septembre 1991 à Toulouse (France). LAAS CNRS.

MORADI, E., FATEMI GHOMI, S. M., & ZANADIEH, M. (2011). Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem. *Expert systems with application*, 7169-7178.

ROBOAM, M. (1988). *Modèles de référence et intégration des méthodes d'analyse pour la conception des systèmes de production.* Thèse de doctorat soutenue à l'université de Bordeaux (France).

RUIZ, R., GARCIA-DIAZ, J. C., & MAROTO, C. (2007). Considering scheduling and preventive maintenance in the flowshop-sequencing problem. *Computers & Operations Research*, 3314–3330.

SASSINE, C. (1998). Intégration des politiques de maintenance dans les systèmes de production. *Thèse de doctorat soutenue à l'INP de Grenoble (France)*.

SCHAFFER, J. D. (1985). Multiple Objective Optimisation with Vector Evaluated Genetic Algorithms. *ICGA*, 93-100.

SCHMIDT, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 1-15.

STORN, R., & PRICE, K. (1997). Differential Evolution -A simple and efficient heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 341-359.

T. W. SLOAN, J. G. (2000). Combined production & Maintenance Scheduling for a Multiple-Product, Single-Machine production system. *Production and Operations Management*, 379-399.

XU, D., SUN, K., & LI., H. (2008). Parallel machine scheduling with almost periodic non-preemptive maintenance and jobs to minimize makespan. *Computers and Operations Research*, 1344-1349.

ZITZLER, E. (1999). *Evolutionary algorithmes for multi-objective optimization: Methods and applications.* Thèse de doctorat, Institut Federal de Technologie de Zurich.