REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère De L'enseignement Supérieur Et De La Recherche Scientifique

UNIVERSITE YAHYA FARES DE MEDEA
FACULTE DES SCIENCES
Département De Mathématique et Informatique

MEMOIRE DE FIN D'ETUDE

Présenté pour l'obtention du Diplôme de **MASTER**

Domaine : Mathématiques et Informatique

Filière : Mathématiques

Option : Recherche Opérationnelle

PAR :
Lahmek Chahinez & Harabi Ahlem

THEME

Ordonnancement d'ateliers sur deux machines dans un système flow shop

Devant le jury:

Mme K. SERIER ABDALLAH	Maitre assistant	UYF Médéa Présidente
Mr. N. IKHLEF ESCHOUF	Maitre de conférences	UYF Médéa Examinateur
Mr.A. BENKACI	Maitre assistant	UYF Médéa Examinateur
Mme. S.KOUIDER AIAD	Maitre assistant	UYF Médéa Promotrice

Année universitaire : 2020/2021

Résumé

Un ordonnancement est définit comme une allocation, dans le temps, des ressources (machines) disponibles aux différents travaux (tâches, jobs) à réaliser, dans le but d'optimiser un ou plusieurs objectifs.

L'axe de recherche traité dans ce mémoire recouvre un domaine très connu dans la recherche opérationnelle, il s'agit de l'ordonnancement d'atelier de production et plus particulièrement l'étude de problème Flow-Shop.

Dans un système Flow-Shop, les tâches doivent être traitées par toutes les machines et l'ordre des opérations est fixé et est le même pour toutes les tâches.

Dans ce travail, nous nous intéressons à étudier les problèmes d'ordonnancement d'ateliers dans un système Flow-Shop sur deux machines.

Mots clés : Ordonnancement, flow shop, complexité, heuristique.

Abstract

A Scheduling is defined as an allocation, in time, of the available resources (machines) to the various tasks (jobs) to be carried out, with the aim of optimizing one or more objectives.

The axis of research treated in this thesis covers a very well known field in operational research, it is the scheduling of production workshop and more particularly the study of Flow-Shop problem.

In a Flow-Shop system, the tasks must be processed by all the machines and the order of operations is fixed and is the same for all the tasks.

In this work, we are interested in studying scheduling problems in a Flow-Shop system on two machines.

Keywords : Scheduling, flow shop, complexity, heuristic.

ملخص

يتم تعريف الجدولة على أنها تخصيص، في الوقت المناسب، للموارد (الآلات) المتاحة لمختلف الوظائف التي سيتم تنفيذها ، بهدف تحسين هدف واحد أو أكثر.

يغطي محور البحث الذي تم تناوله في هذه الأطروحة مجالا مشهورا في البحث العملياتي ، وهو جدولة ورش الإنتاج وبشكل أكثر تحديدا دراسة مشكلة فلو شوب .

في نظام فلو شوب ، يجب معالجة المهام بواسطة جميع الأجهزة ويكون ترتيب العمليات ثابت وهو نفسه لجميع المهام.

في هذا العمل ، نحن مهتمون بدراسة مشاكل جدولة ورش العمل في نظام فلو شوب على جهازين .

الكلمات المفتاحية : جدولة، فلو شوب، تعقيد، استدلال.

Remerciements

Au terme de ce travail, nous tenons tout d'abord à remercier Dieu le tout puissant, le miséricordieux, qui nous a donné la force et la patience nécessaire en vue d'accomplir ce modeste travail.

que tous ceux qui ont participé à la réalisation de ce mémoire, trouvent ici l'expression de notre grande et sincère reconnaissance, et tout particulièrement :

***Mme.S. KOUIDER AIAD** notre promotrice de mémoire, pour avoir accepté de nous encadrer, pour son suivi et ses orientations enrichissantes des le commencement jusqu'à la mise en forme de ce mémoire.*

***Mme.K.SERIER ABDALLAH** , qui nous a fait l'honneur de présider le jury de soutenance .*

***Mr.N.IKHLEF ESCHOUF & Mr.A.BENKACI**, qui ont accepté à ce jury en tant qu'examinateurs du mémoire.*

nous remercions chaleureusement tous les enseignants du département de recherche opérationnelle de l'université YAHIA FARES, Médéa

Enfin, nous remercions nos amis qu'on n'a pas cités et qui ont toujours été là pour nous. Leurs soutien inconditionnel et leurs encouragements ont été d'une grande aide.

Dédicace

Je dédie ce fruit de mes longues années d'études tout d'abord :

A mes très chers parents mama "Rachida" et papa "Elhadj", qui sont la lumière de ma vie, qui ont tant soufré et sacrifiés pour que je sois heureuse, pour leurs conseils, leurs affection et leurs encouragements.

Je vous remercie pour tout vos efforts fournis pour moi, que Dieu vous garde, vous protège, et vous bénisse la vie.

Et je le dédie :

A tous les membres de ma famille, mes frères « Messaoud » et « Achraf » et « Rafik » .

A ma binome "HALOMA", mon intime amie et ma sœur qui n'existait pas, merci pour tous les beaux moments que nous avons vécus, dieu m'a toujours préserver pour moi.

A tous mes amis et mes collègues avec qui
j'ai partagé de très bons moments tout le long de ces années.

A touts les enseignants et les collègues de
la promotion de la deuxième année Master 2020/2021.

LAHMEK
CHAHINEZ

Dédicace

Je dédie ce fruit de mes longues années d'études tout d'abord :

A ma chère mère "kheira"qui a toujours travaillé dur pour moi, et je prie dieu de la protéger et de prolonger sa vie, maman j'adore.

A l'âme pur de mon père , je prie dieu d'avoir pitié de lui et d'habiter dans le plus haut paradis, papa j'adore.

Et je le dédie :

A tous les membres de ma famille, mes sœurs et mes frères et leurs enfants .

A ma binome "CHANEZ", mon intime amie et ma seour qui n'existait pas, merci pour tous les beaux moments que nous avons vécus, dieu m'a toujours préserver pour moi.

A tous mes amis et mes collègues avec qui j'ai partagé de très bons moments tout le long de ces années.

A touts les enseignants et les collègues de la promotion de la deuxième année Master 2020/2021.

HARABI

AHLEM

Table des matières

Introduction générale 14

Chapitre 1 Généralités sur l'ordonnancement **16**

1.1 Introduction 16
1.2 Présentation d'un problème d'ordonnancement 16
1.2.1 Définition d'un problème d'ordonnancement 16
1.2.2 Objectif de l'ordonnancement 17
1.2.3 Ordonnancement d'atelier 17
1.3 Formulation d'un problème d'ordonnancement 17
1.3.1 Les tâches 17
1.3.2 Les ressources 17
1.3.3 Les contraintes 18
1.3.4 Les objectifs 18
1.4 Représentation des problèmes d'ordonnancement par un diagramme de Gantt 19
1.5 Caractérisation des machines d'un atelier 20
1.5.1 Les machines parallèles (parallel machines) 20
1.5.2 Les machines spécialisées ou dédiées (dedicated machines) 20
1.6 Classification $\alpha/\beta/\gamma$ 22
1.6.1 Champ α : Ressources 22
1.6.2 Champ β : Contraintes 23
1.6.3 Champs γ : critère d'optimisation 24
1.7 Analyse de la complexité 24
1.7.1 Les principes fondamentaux de complexité 24
1.7.2 Les classes de complexité 25
1.7.3 Réduction entre problème d'ordonnancement 26
1.8 Conclusion 28

Chapitre 2 Problèmes de flow-shop à deux machines **29**

2.1 Introduction 29
2.2 Problème de flow-shop 29
2.3 Méthodes exactes pour les problèmes de flow shop 30
2.3.1 Méthode de branch and bound 30

2.3.2 Programmation linéaire . . . 30
2.4 Méthodes approchées pour le problème de flow shop . . . 31
2.4.1 Heuristique . . . 31
2.4.2 Méta-heuristique . . . 31
2.5 Problème F2/ /C$_{max}$ . . . 31
2.5.1 Description du problème . . . 31
2.5.2 Résolution de problème . . . 31
2.5.3 Algorithme de Johnson pour les problèmes de flow shop à deux machines . . . 32
2.6 Problème F2 /recr(2) /C$_{max}$ . . . 33
2.6.1 Description du problème : . . . 33
2.6.2 Algorithme F2/recr(2)/C$_{max}$ . . . 34
2.7 Problème F2/coup-opr(1),ai,bi,Li,ci/C$_{max}$ . . . 35
2.7.1 Description du problème : . . . 35
2.7.2 Résultats de complexité . . . 35
2.8 Problème F2/recr(1)/C$_{max}$ . . . 36
2.8.1 Description du problème . . . 36
2.8.2 Complexité du problème . . . 37
2.9 Problème F2/r$_i$/C_{max} . . . 37
2.9.1 Description du problème . . . 37
2.9.2 Complexité du problème . . . 37
2.10 Conclusion . . . 38

Chapitre 3 Heuristiques proposées pour résoudre le problème $F2/r_i/C_{max}$ **39**
3.1 Introduction . . . 39
3.2 Représentation du problème F2/r$_i$/C_{max} . . . 39
3.3 Heuristiques pour résoudre le problème F2/r$_i$/C_{max} . . . 40
3.3.1 Définition . . . 40
3.3.2 Les heuristiques proposées . . . 40
3.3.2.1 A- Principe de la première heuristique . . . 40
3.3.2.2 B- Principe de la deuxième heuristique . . . 41
3.3.2.3 C- Principe de la 3^{me} heuristique . . . 42
3.3.3 Exemple d'application . . . 43
3.3.3.1 A- Application de la 1^{re} heuristique . . . 43
3.3.3.2 B- Application de la 2^{me} heuristique . . . 44
3.3.3.3 C- Application de la 3^{me} heuristique . . . 45
3.3.3.4 Comparaison entre les heuristiques proposées . . . 47
3.4 Conclusion . . . 48
Conclusion générale . . . 49

Bibliographie 50

Table des figures

1.1 Diagrame de Gantt d'un ordonnancement 19
1.2 Atelier de type flow-shop 21
1.3 Atelier de type Open-shop 21
1.4 Atelier de type Job-shop 22
1.5 Schéma représentent les types de machines 22
1.6 Schéma représentatif de classes de complexité 26
1.7 Réduction entre les machines 26
1.8 Réduction entre les critères 27
1.9 Relation de précédence 27
1.10 Quelques Réductions 28

2.1 Solution optimale de l'exemple 1 33
2.2 Flow shop à deux machines avec recirculation des tâches sur la deuxième machine . . . 33
2.3 Solution optimale de l'exemple 2 34
2.4 Flow shop à deux machines avec recirculation des tâches sur la première machine . . . 36
2.5 Solution de l'exemple précédent 37

3.1 Solution générale de l'exemple par l'heuristique H1 44
3.2 Solution générale de l'exemple par l'heuristique H2 45
3.3 Solution générale de l'exemple par l'heuristique H3 47

Notations et abréviations

n	Nombre de tâches.
m	Nombre de machines.
i,k	Indice des tâches.
j	Indice des machines.
T=$\{T_1, ..., T_n\}$	Ensemble des tâches.
M=$\{M_1, ..., M_m\}$	Ensemble des machines .
r_i	Date de disponibilité du tâche T_i .
p_i	Temps de traitement du tâche T_i .
C_{max}(makespan)	Date de fin de traitement de toutes les tâches.
$\overline{C}$	Le temps de fin de traitement moyen .
$\overline{C_w}$	Le temps de fin de traitement moyen pondéré.
L_{max}	La grande tardivité.
$\overline{L}$	La tardivité moyenne .
$\overline{L_w}$	La tardivité moyenne pondérée .
T_{max}	Le grand retard.
$\overline{T}$	Le retard moyen .
$\overline{T_w}$	Le retard moyen pondéré .
F_{max}	Le grand temps de flow.
$\overline{F}$	Le temps de flow moyen .
$\overline{F_w}$	Le temps de flow moyen pondéré .
N_T	Le nombre de tâches en retard .
O_{ji}	L'opération de la tâche T_i sur la machine M_j.
d_i	Date échue du tâche T_i .
$\tilde{d}_i$	Date limite du tâche T_i.
S_i	Temps de préparation .
P	Polynomial.
NP	Non polynomial.
pmtn	la préemption.
recr(1)	Recirculation sur la première machine.
recr(2)	Recirculation sur la deuxième machine.

INTRODUCTION GÉNÉRALE

La recherche opérationnelle est un ensemble de techniques et des méthodes affin de définir la meilleure façon d'effectuer un choix pour arriver au meilleur résultat.
La recherche accompagne les décisions dans l'évaluation des enjeux qui conduisent à un arbitrage, la recherche opérationnelle est confluent de l'informatique des mathématiques appliquées, de la gestion et du génie industriel ..., concevoir des systèmes informatique, établir des horaires de travail de cours ou des calendriers sportifs.
La recherche opérationnelle est née pendant la seconde guerre mondiale des efforts conjugués d'éminent mathématiciens (dont von Neumam Dantzig, Balackett) à qu'il avait été demandé de fournir des techniques d'optimisation des ressources militaires, la première succès de cette approche a été obtenue en 1940 parle prix nobel de physique patrick Blakett qui résolut un problème d'implantation optimale de radars de surveillance, le qualification "opérationnelle" vient du fait que les premières applications de cette discipline avait trait aux opérations militaires.
La domination est restée par la suite même si le domaine n'est plus le principale champ d'application de cette discipline, le mot (opérationnelle) prenant alors plutôt le sens d"effectif"ce sont donc ces mathématiciens qui on crée une nouvelle méthodologie caractérisée par les mots-clés modélisation et optimisation.
A partir des années 50 la recherche opérationnelle fait son entrée dans les entreprises en France, comme EDF, Air France... [18]
Les problèmes d'ordonnancement apparaissent dans les domaines de l'économie, l'informatique (tâches : jobs, ressource : processeurs ou mémoire...), la constraction (suivi de projet), l'industrie (problème d'atelier, gestion de production) l'administration (emplois du temps).
La théorie de l'ordonnancement traite des modèles mathématiques mais aussi de la modélisation de situation réelles fort complexes, aussi le développement de modèles utiles ne peut être que le fruit de contacts entre la théorie et la pratique [5].
Dans notre travail nous nous intéressons principalement à l'ordonnancement d'atelier sur deux machines dans un système flow shop.

Pour cela nous avons divisé ce travail en trois chapitres, ils sont organisé comme suit :
Dans le premier chapitre nous rappelons les notions de base nécessaires de la théorie de l'ordonnancement, notamment la présentation, la formulation et la classification d'un problème d'ordonnancement, ainsi que la théorie de la complexité.

Nous nous concentrons dans ce chapitre au problème de flow-shop. Dans un premier lieu nous étudions ces différents propriété et restrictions. Ensuite nous étudions quelques problèmes faciles et difficiles dans un système flow-shop en deux machines pour terminer ce chapitre avec la résolution du ces problèmes.

Nous traitons dans le dernier chapitre le problème d'ordonnancement $F2/r_i/C_{max}$, nous avons commencé par la représentation de ce problème ainsi nous proposons trois heuristiques pour trouver une solution approchée de ce dernier avec un exemple d'application de ces heuristiques.

nous nous terminons notre travaille par une conclusion générale .

CHAPITRE

1

GÉNÉRALITÉS SUR L'ORDONNANCEMENT

1.1 Introduction

Le premier chapitre présente les notions de base nécessaires de la théorie de l'ordonnancement, notamment la présentation, la formulation et la classification d'un problème d'ordonnancement, ainsi que la théorie de la complexité.

1.2 Présentation d'un problème d'ordonnancement

1.2.1 Définition d'un problème d'ordonnancement

Le problème d'ordonnancement consiste à placer dans le temps des activités ou tâches, compte tenu de contraintes (délais, contraintes d'enchainement...) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches .
Le problème d'ordonnancement apparaissant dans tous les domaines, le domaine industrie, informatique, construction, administration [7].
Les principes éléments d'un problème d'ordonnancement sont les tâches, les ressources, les contraintes et les critères [22].
En résoudre un problème d'ordonnancement, c'est choisir pour chaque opération une date de début,

de telle sorte que les contraintes du problème soient respectées (la solution est alors dite admissible ou réalisable) et en qu'un ou plusieurs critères donnés soient optimisés.

1.2.2 Objectif de l'ordonnancement

Le processus d'ordonnancement n'est pas banal car il doit permettre de trouver le meilleure compromis entre deux objectifs contradictoires : maximiser la satisfaction du client tout en minimisant les coûts de production [9].

1.2.3 Ordonnancement d'atelier

L'ordonnancement d'atelier consiste à organiser dans le temps le fonctionnement d'un atelier pour utiliser au mieux les ressources humaines et matérielles disponible dans le but de produire les quantités désirées dans le temps imparti [**?**].

1.3 Formulation d'un problème d'ordonnancement

Les éléments nécessaires pour formuler un problème d'ordonnancement sont :

1.3.1 Les tâches

Une tâche est une entité élémentaire localisée dans le temps, par une date de début et/ou de fin, et dont la réalisation nécessite une durée préalablement définie.
Elle est constituée d'un ensemble d'opérations qui requiert, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif.
On distingue deux types de tâches :
Tâches morcelables (préemptibles) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.
Tâches non morcelables (indivisibles) qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

1.3.2 Les ressources

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources :
Les ressources renouvelables qui, après avoir été allouées à une tâche, redeviennent disponibles (machines, personnel, etc).
Les ressources consommables, qui, après avoir été allouées à une tâche, ne sont plus disponibles (argent, matières premières, etc).
Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Par ailleurs, dans le cas des ressources renouvelables, on distingue principalement, les ressources

disjonctives qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité .

1.3.3 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre simultanément les variables représentant les relations reliant les tâches et les ressources. On distingue deux types de contraintes, les contraintes temporelles et les contraintes de ressources.

A- Les contraintes temporelles : Les contraintes temporelles concernent les délais de fabrication imposées. Ces contraintes peuvent être :

- **Des contraintes de dates butoirs** : certaines tâches doivent être achevées avant une date préalablement fixée.
- **Des contraintes de précédence** : une tâche T_i doit précéder T_k.
- **Des contraintes de dates au plus tôt** : liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

B- Les contraintes de ressources : Ces contraintes concernent la limitation de la quantité des ressources de chaque type. Dans ce cadre, deux types de contraintes de ressources sont distingués :

- **Les contraintes disjonctives** : induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource.
- **Les contraintes cumulatives** : impliquant la limitation du nombre de tâches à réaliser en parallèle.

1.3.4 Les objectifs

Généralement l'objectif d'un problème d'ordonnancement consiste à optimiser un ou plusieurs critères ou à trouver une solution admissible qui respect les contraintes du problème. Les critères liés aux paramètres temporels caractérisant les tâches les plus utilisés sont :

- Longueur d'ordonnancement : $C_{max} = \max_{1 \leqslant i \leqslant n} \{C_i\}$
- Le temps de fin de traitement moyen : $\overline{C} = \frac{1}{n} \sum_{i=1}^{n} C_i$
- Le temps de fin de traitement moyen pondéré :$\overline{C_w} = \frac{\sum_{i=1}^{n} w_i C_i}{\sum_{i=1}^{n} w_i}$
- La grande tardivité : $L_{max} = \max_{1 \leqslant i \leqslant n} \{l_i\}$
- La tardivité moyenne :$\overline{L} = \frac{1}{n} \sum_{i=1}^{n} l_i$
- La tardivité moyenne pondérée : $\overline{L_w} = \frac{\sum_{i=1}^{n} w_i l_i}{\sum_{i=1}^{n} w_i}$

— Le grand temps de flow : $F_{max} = \max_{1 \leqslant i \leqslant n} \{f_i\}$

— Le temps de flow moyen :$\overline{F} = \frac{1}{n} \sum_{i=1}^{n} f_i$

— Le temps de flow moyen pondéré : $\overline{F_w} = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i}$

— Le grand retard : $T_{max} = \max_{1 \leqslant i \leqslant n} \{t_i\}$

— Le retard moyen : $\overline{T} = \frac{1}{n} \sum_{i=1}^{n} t_i$

— Le retard moyen pondéré : $\overline{T_w} = \frac{\sum_{i=1}^{n} w_i t_i}{\sum_{i=1}^{n} w_i}$

— Le nombre de tâches en retard :$N_T = \sum_{i=1}^{n} u_i$ [21].

1.4 Représentation des problèmes d'ordonnancement par un diagramme de Gantt

Le diagramme de Gantt est un outil permettant de modéliser la planification des tâches nécessaires à la réalisation d'un projet. Il s'agit d'un outil élaboré en 1917 par Henry L.Gantt.
Étant donné la facilité relative de lecture des diagrammes de Gantt, cet outil est utilisé par la quasi-totalité des chefs de projet dans tous les secteurs. Il permet de représenter graphiquement l'avancement du projet et constitue également un bon moyen de communication entre les différents acteurs d'un projet. Le diagramme de Gantt présente en ordonnée la liste des tâches T_i à exécuter par les machines notées M_j et en abscisse l'échelle du temps, comme le montre la figure 1.1 dans le cas ou $i = 1, 2, ..., 5$ et $j = 1, 2$ [6].

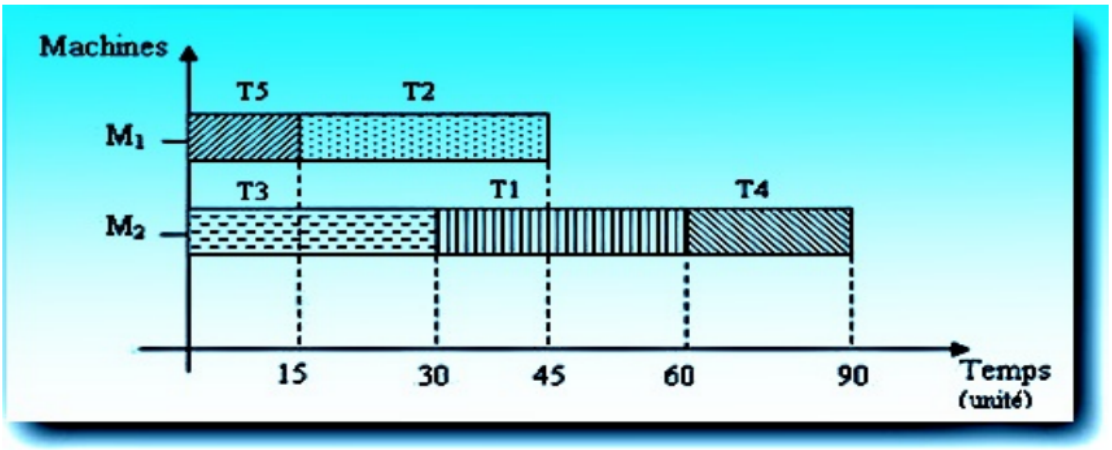


FIGURE 1.1: Diagrame de Gantt d'un ordonnancement

1.5 Caractérisation des machines d'un atelier

Il existe en général 2 types de machines [13] :

1.5.1 Les machines parallèles (parallel machines)

Ces machines exécutent les même traitement et sont donc capable de traiter n'importe quelle tâche. En fonction de la vitesse de traitement, nous distinguons 3 types de machines parallèles :

Machines identiques (identical machines)

Pour ces machines, les vitesses de traitement des tâches sont toutes égales.

Machines uniformes (uniform machines)

Pour ces machines, les vitesses de traitement des tâches sont différentes mais que la vitesse de chaque machine est constante et ne dépend pas de la tâche T_i.

Par example si une opération est exécutée deux fois plus rapidement sur une machine M_1 que sur une machine M_2, cela reste vraie pour toute autre opération exécutable par ces machines parallèles.

Machines générales (unrelated machines)

Contrairement au cas précédent, pour ces machines, la vitesse de traitement d'une machine dépend de la tâche qui est traitée.

1.5.2 Les machines spécialisées ou dédiées (dedicated machines)

Ces machines sont spécialisées à l'exécution de certaines tâches. Dans le cas de machines spécialisées, nous distinguons 3 systèmes ou mode de traitement :

Flow shop

Dans un système Flow shop, les tâches doivent être traitées par toutes les machines, et l'ordre des opérations est fixé et est le même pour toutes les tâches.

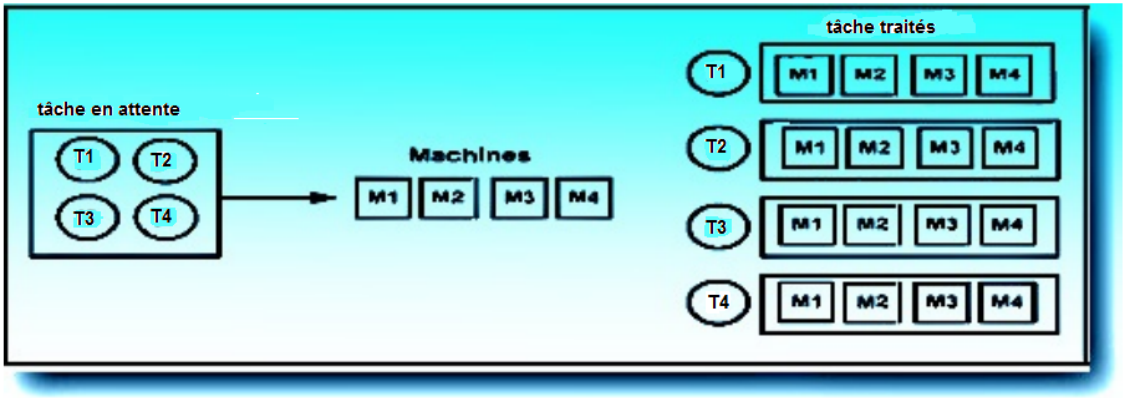


FIGURE 1.2: Atelier de type flow-shop

Open shop

Dans un système Open shop, les tâches doivent être traitées par toutes les machines mais l'ordre de traitement n'est pas nécessairement le même pour toutes les tâches. L'ordre de traitement est quelconque.

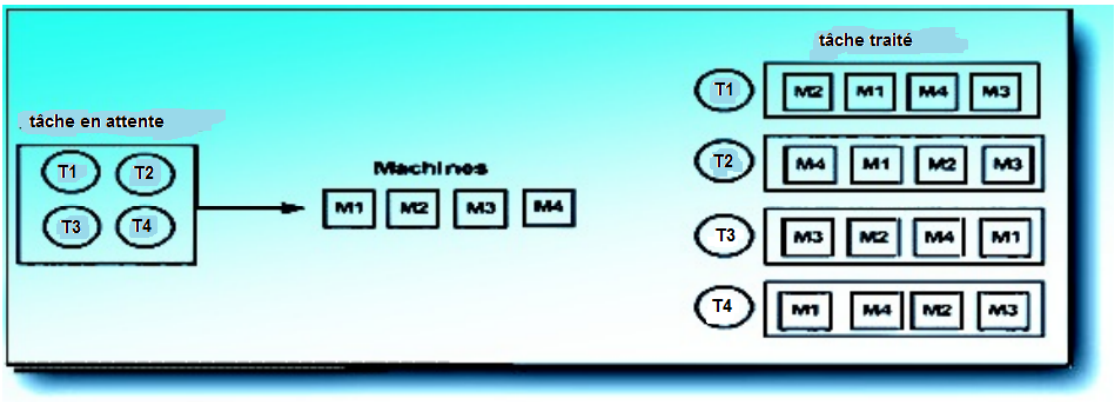


FIGURE 1.3: Atelier de type Open-shop

Job shop

Dans un système Job shop, un sous-ensemble de machines devant traiter une tâche et l'ordre du traitement est arbitraire, mais doit être spécifié à l'avance.

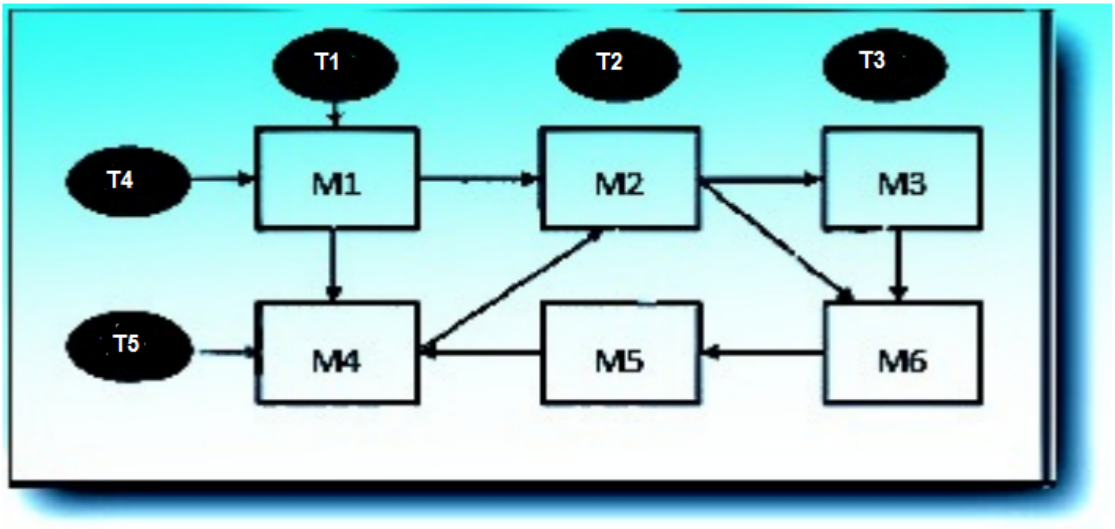


FIGURE 1.4: Atelier de type Job-shop

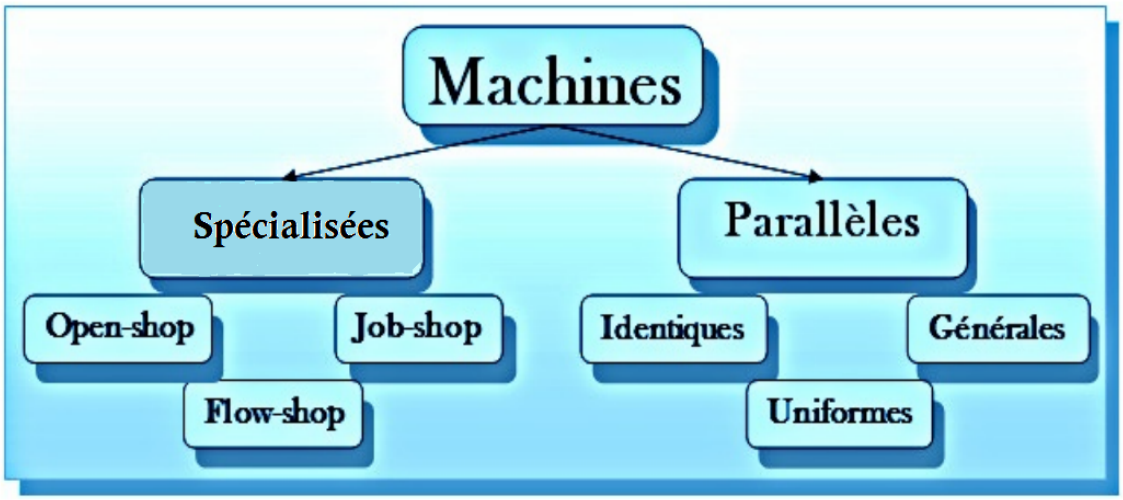


FIGURE 1.5: Schéma représentent les types de machines

1.6 Classification $\alpha/\beta/\gamma$

Nous adoptons dans ce qui suit la notation la plus utilisée en ordonnancement à savoir celle introduite par Graham et al.[1979]qui décrit les problèmes d'ordonnancement en trois champs $\alpha/\beta/\gamma$ [5].

1.6.1 Champ α : Ressources

Renseigne les informations relatives aux ressources, à savoir le nombre de machines, type de machines et les types d'ateliers. Il est composé de deux sous champs et désigné par $\alpha_1; \alpha_2$:

$\alpha_1 \in \{\emptyset; P; Q; R; F; O; J\}$: Type de machine.

$\emptyset$: Une seule machine.
P : Machine parallèle identique : la vitesse d'exécution est la même pour toutes les machines M_j et toutes les tâches T_i.
Q : Machine parallèle uniforme : chaque machine M_j a une vitesse d'exécution propre et constante et la vitesse d'exécution est la même pour toutes les tâches T_i d'une même machine M_j.
R : Machine parallèle générale : la vitesse d'exécution est différente pour chaque machine M_j et pour chaque tâche T_i.
F : Machine spécialisée (système flow shop) : chaque tâche doit être exécutée sur l'ensemble des machines disposées en série, selon un même routage.
O : Machine spécialisée (système open shop) : les tâches doivent être exécutées sur certaines machines sans restriction sur le routage des tâches.
J : Machine spécialisée (système job shop) : les tâches doivent être exécutées sur l'ensemble des machines disposées en série, mais peuvent avoir des routages différents.

$\alpha_2 \in \{\emptyset, m\}$: Nombre de machine.
$\alpha_2 = \emptyset$: Le nombre de machine est variable.
$\alpha_2 = m$: Le nombre de machine est constant et égal à m.

1.6.2 Champ β : Contraintes

Permet d'obtenir des informations sur les contraintes prises en compte aussi bien sur les tâches qui doivent être ordonnancés. Il est composé de 6 sous champs $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6$:

$\beta_1 \in \{\emptyset, pmtn\}$ indique l'autorisation ou non de la préemption.
$\beta_1 = \emptyset$: La préemption n'est pas autorisée (lorsqu'une opération est commencée, elle doit être terminée avant de pouvoir exécuter une autre opération sur la même machine)
$\beta_1 = pmtn$: La préemption est autorisée (l'exécution d'une opération peut être interrompue puis reprise sur la même machine ou sur une autre).

$\beta_2 \in \{\emptyset, prec, tree, chain, etc...\}$ décrit les contraintes de précédence.
$\beta_2 = \emptyset$: Pas de contraintes de précédence.
$\beta_2 = prec$: Contraintes de précédence quelconque.
$\beta_2 = tree$: Contraintes de précédence sont données sous forme d'un arborescence.
$\beta_2 = chain$: Contraintes de précédence sont données sous forme de chaines.

$\beta_3 \in \{\emptyset, ri\}$ décrit les dates de disponibilité des différentes tâches.
$\beta_3 = \emptyset$: Toutes les dates de disponibilité sont nulles.
$\beta_3 = ri$: Il existe des dates de disponibilité pour les tâches non identique.

$\beta_4 \in \{\emptyset, p_i = p, \underline{p} \leq p_i \leq \overline{p}\}$ définit les particularités des durées de traitement.

$\beta_4 = \emptyset$: Les temps de traitement des tâches sont quelconques et non identique.

$\beta_4 = p_i = p$: Les temps de traitement des tâches sont égaux à p.

$\beta_4 = \underline{p} \leq p_i \leq \overline{p}$: Les temps de traitement de chaque tâche est compris entre $\underline{p}$ et $\overline{p}$.

$\beta_5 \in \left\{\emptyset, d_i, \tilde{d}_i\right\}$ les dates échues et les dates limites

$\beta_5 = \emptyset$: Il n'y a pas des tâches échues pour les tâches, ou il existe des dates échues pour les tâches dans le cas ou les critères d'optimisation est en fonction de ces derniers.

$\beta_5 = d_i$: Il existe des dates échues pour les tâches.

$\beta_5 = \tilde{d}_i$: Il existe une date limite pour chaque tâche.

$\beta_6 \in \{\emptyset, S_i\}$ définit le temps de préparation

$\beta_6 = \emptyset$: Il n'existe pas de temps de préparation.

$\beta_6 = S_i$: S'il existe un temps de préparation.

1.6.3 Champs γ : critère d'optimisation

Le troisième champ décrit le critère d'optimalité.

On cherche à minimiser Y avec $Y \in \{C_{max}, \overline{C}, \overline{C_w}, T_{max}, L_{max} ...\}$

1.7 Analyse de la complexité

1.7.1 Les principes fondamentaux de complexité

La complexité

A première vue, comment déterminer un algorithme efficace ? Pour un problème donné, chercher un algorithme efficace, veut dire trouver un algorithme où le temps nécessaire à son exécution ne soit pas trop important. Un problème est dit facile si on peut le résoudre facilement, c'est-à-dire s'il ne fait pas trop de temps pour arriver à la solution. Donc, s'il existe un algorithme efficace pour un problème donné, alors ce dernier est dit facile.

Un problème pour lequel on ne connaît pas d'algorithme efficace, est ce qu'il est facile ou difficile ? De nombreux chercheurs se sont penchés sur ce genre de problèmes et ils ont développé une théorie appelée de la complexité. Nous n'allons pas détailler cette théorie, mais nous allons, quand même donner une idée globale du sujet en question [8].

Problème de décision

Le problème d'optimisation est appelé aussi problème de reconnaissance, est un problème dont la résolution se limite à la réponse par "oui" ou par "non" à la question de savoir s'il existe une solution à

un problème donner.

Pour chaque problème d'optimisation, on peut associer un problème de décision défini par la question suivante : « Existe-t-il une solution telle que son évaluation est majorée (Resp. minorée) par une constante fixée ? ». Ceci est pour un problème d'optimisation, dont la fonction objectif est la minimisation (resp. maximisation) [10].

Alors si la réponse à la question existe et est obtenue grâce à un algorithme polynomial alors il existera un algorithme polynomial résolvant le problème d'optimisation associé. Cette propriété nous permet de classer les problèmes d'optimisation grâce à leurs problèmes de décision. Ces problèmes peuvent être classés en deux classes principales : la classe P et la classe NP.

1.7.2 Les classes de complexité

Classe NP

La classe NP (Non déterministe Polynomial) est celle des problèmes d'existence dont une proposition de solution est Oui et qui est vérifiable pronominalement. Parmi les problèmes décidables, les plus simples à résoudre sont regroupés dans la classe.

Classe P

Un problème est dit polynomial s'il existe un algorithme de complexité polynomiale permettant de répondre à la question posée dans ce problème, quelle que soit la donnée de celui-ci. La classe P est l'ensemble de tous les problèmes de reconnaissance polynomiaux. Pour le reste de la classe NP, on n'est pas sûr qu'il n'existe pas un algorithme polynomial pour résoudre chacun de ses problèmes. Ainsi, on sait que P est inclue dans NP mais on n'a pas pu prouver que P n'est pas NP.

Classe NP-Complet

La classe NP-Complet regroupe les problèmes les plus difficiles de la classe NP. Elle contient les problèmes de la classe NP tels que n'importe quel problème de la classe NP leur est polynomiale-ment réductible. Entre eux, les problèmes de la classe NP-Complet sont aussi difficiles [19].

Un problème de décision est dit NP-complet s'il appartient à la classe NP et il résolu, au mieux, en un temps exponentiel.

Remarque :

Un problème d'optimisation est dit NP-Difficile, si le problème de décision associé est NP-complet [11].

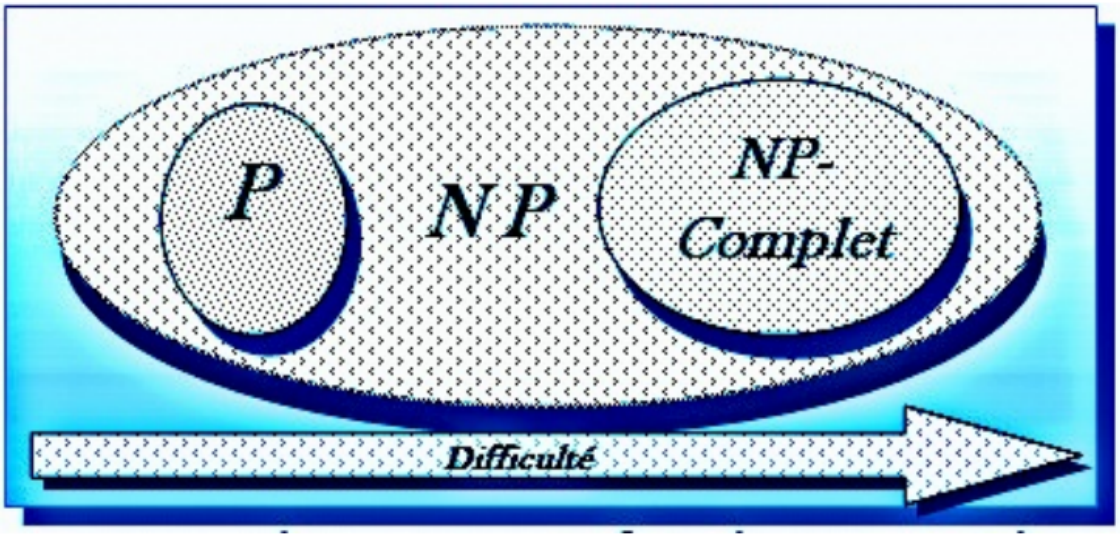


FIGURE 1.6: Schéma représentatif de classes de complexité

1.7.3 Réduction entre problème d'ordonnancement

Nous donnons ci-dessous quelques réductions (transformations polynomiales) entre problème d'ordonnancement. Une tâche de X à Y signifié que la résolution d'un modèle avec le paramètre Y soit plus difficile que celle du même modèle avec le paramètre X, ces relations nous aide à déterminer la complexité des problèmes [7].

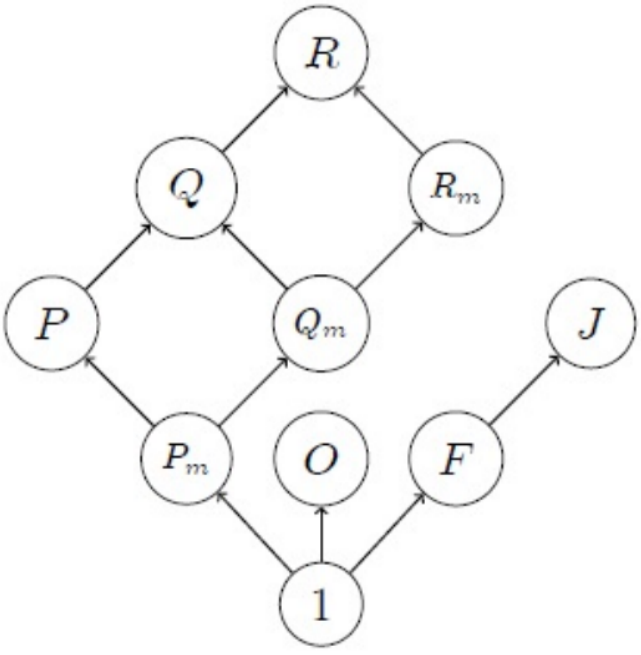


FIGURE 1.7: Réduction entre les machines

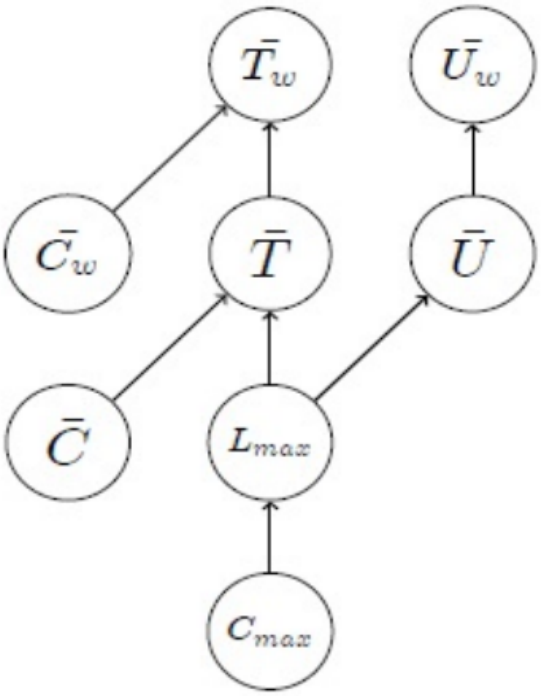


FIGURE 1.8: Réduction entre les critères

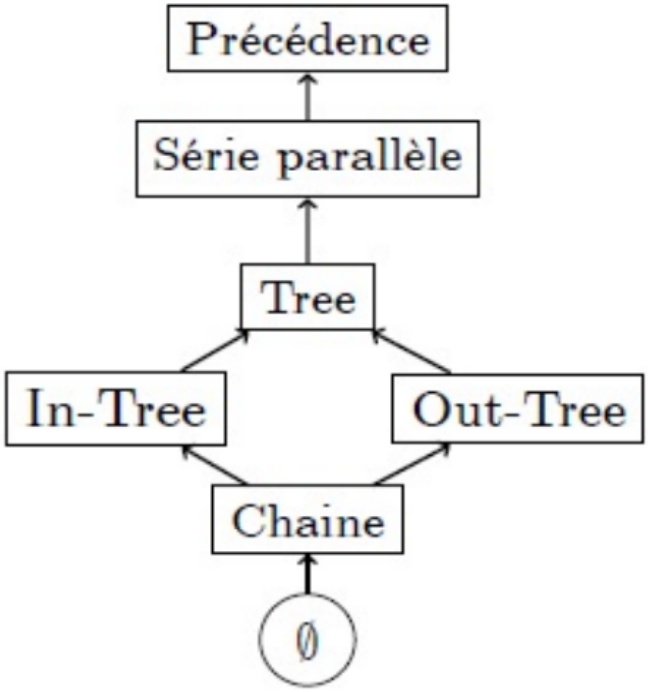


FIGURE 1.9: Relation de précédence

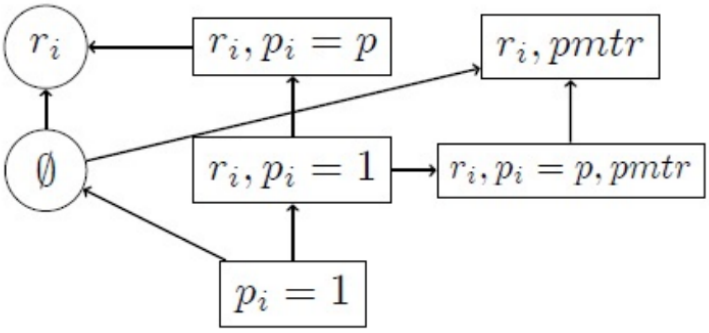


FIGURE 1.10: Quelques Réductions

1.8 Conclusion

Dans ce chapitre, nous avons définit un problème d'ordonnancement, nous avons présenté les éléments nécessaires pour formuler un problème d'ordonnancement à savoir les tâches, les ressources, les contraintes et les objectifs, ainsi que la classification $\alpha/\beta/\gamma$, nous avons introduit la notion de complexité des différents problèmes d'ordonnancement.

CHAPITRE

2

PROBLÈMES DE FLOW-SHOP À DEUX MACHINES

2.1 Introduction

Nous nous concentrons dans ce chapitre au problème de flow-shop. Dans un premier lieu nous étudions ces différents propriété et restrictions. Ensuite nous étudions quelques problèmes faciles et difficiles dans un système flow-shop en deux machines pour terminer ce chapitre avec la résolution du ces problèmes.

2.2 Problème de flow-shop

Le flow shop définit un ensemble de n tâches et m machines. Les contraintes du problème sont de deux types :

— Les contraintes de gamme, toutes les tâches doivent passer sur toutes les machines, de la machine 1 à la machine m ;

— Les contraintes de ressources, une machine ne peut traiter qu'une tâche à la fois.

En générale, on cherche des solutions telles que les tâches ne peuvent pas se doubler, elles passent dans le même ordre sur toutes les machines, c'est ce qu'on appelle le flow shop. Dans le cas le plus simple,

les données du problème sont les temps que chaque tâche passe sur chaque machine, soit une matrice dans $M_{nm}(R^+)$, dont les coefficient ont nommés p_{ji} (temps pour passer par la tâche T_i sur la machine M_j) [**?**].
Les fonctions objectifs sont généralement :

— C_{max}, date de fin de la dernière tâche sur la machine m, soit le temps total passé à exécuter tous les travaux ;

— $\sum_{i=1}^{n} c_i$, somme des dates de fin des tâches sur la machine m.

Plusieurs recherches effectuées sur le problème de flow-shop où l'ordre de passage des tâches sur chacune des machines est identique. Résoudre ce problème revient à trouver une permutation $\pi = (\pi(l), \pi(2), ..., \pi(n))$ minimisant le makespan.

Johnson a prouvé que le problème à deux machines est résoluble en O(n log n). Le problème devient NP-difficile au sens fort pour $m \geq 3$.

2.3 Méthodes exactes pour les problèmes de flow shop

2.3.1 Méthode de branch and bound

L'algorithme Branch and Bound consiste à placer progressivement les tâches sur les ressources en explorant un arbre de recherche décrivant toutes les combinaisons possibles. Il s'agit de trouver la meilleure configuration donnée de manière à élaguer les branches de l'arbre qui conduisent à de mauvaises solutions. L'algorithme branch and bound effectue une recherche complète de l'espace des solutions d'un problème donné, pour trouver la meilleure solution. La démarche de l'algorithme Branch and Bound consiste à :

— Diviser l'espace de recherche en sous espaces

— Chercher une borne minimale en terme de fonction objectif associée à chaque sous espace de recherche

— Éliminer les mauvais sous-espaces

— Reproduire les étapes précédentes jusqu'à l'obtention de l'optimum global [1].

2.3.2 Programmation linéaire

C'est un outil d'optimisation très puissant qui permet de résoudre un grand nombre de modèles linéaires. Son utilisation demande que le problème posé puisse se ramener à l'optimisation d'une fonction de forme linéaire, en respectant un ensemble de contraintes elles aussi linéaires, fonctions des mêmes variables positives ou nulles. Des contraintes très variées peuvent être ajoutées de la même façon. L'étape de modélisation du problème peut s'avérer difficile, et la résolution d'un programme

linéaire complexe peut demander un temps relativement long. Parmi les logiciels de résolution des problèmes de programmation linéaire, on peut citer LP-Solve, Cplex, Mosel-Xpress (qui est aussi un langage de modélisation et de résolution des problèmes d'optimisation) [**?**].

2.4 Méthodes approchées pour le problème de flow shop

2.4.1 Heuristique

L'heuristique est un algorithme de résolution d'un problème complexe, exploitant au mieux la structure du problème considéré et permettant de lui trouver rapidement une réponse numérique ; des analyses comparatives permettant d'évaluer la performance d'heuristique alternatives pour traiter une classe de problèmes ; aucune garantie d'optimalité n'est donnée et on peut, en général, simplement espérer obtenir rapidement une $\ll$ bonne $\gg$ solution [2].

2.4.2 Méta-heuristique

Les méta-heuristiques sont des techniques puissantes et généralement applicable à un grand nombre de problèmes. Formellement, une méta-heuristique fait référence à une stratégie de maître interactif qui guide et modifie les opérations heuristiques subordonnées en combinant intelligemment différents concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou proches de l'optimales. Les performances de ce type de méthodes dépendent de la qualité de la solution obtenue et du temps de calcul nécessaire. Les méta-heuristiques n'offrent généralement pas de garantie d'optimalité, mais leur succès est dû à la capacité de résoudre en pratiques certains problèmes difficiles [**?**].

2.5 Problème F2/ /C_{max}

2.5.1 Description du problème

Traiter un ensemble de tâches indépendantes non interruptibles ayant des temps de traitement arbitraires et des dates de disponibilités nulles sur deux machines spécialisées dans un système flow shop, l'objectif est la minimisation de la date de fin de traitement.

Ce problème est de complexité O(n log n).

2.5.2 Résolution de problème

On cherche à minimiser n tâches sur 2 machines ; chaque tâche selon l'ordre suivant M_1, M_2 et à minimiser le temps maximum de séjour dans l'atelier C_{max}.
Le problème se résout de manière optimale par l'algorithme de Johnson.

2.5.3 Algorithme de Johnson pour les problèmes de flow shop à deux machines

Chaque tâche comporte maintenant deux opérations distinctes effectuées par deux centres de production différents. On suppose que la préemption est interdite. Pour simplifier, on conviendra que ces deux centres de production sont les centres de production A et B. on notera p_{Ai} et p_{Bi} les temps d'exécution de la tâche dans chacun des centres de production.

L'algorithme de Johnson à pour objectif de minimiser la durée de réalisation d'une fille d'attente de n tâches devant tout passer selon le même ordre sur deux machines.

Algorithm 1: Algorithme de Johnson [11]

1 **Début**
2 Établir l'ensemble E1 des fabrications dont le temps opératoire sur la première machine est inférieur ou égal au temps opératoire sur la deuxième machine ;
3 Établir l'ensemble E2 des fabrications dont le temps opératoire sur la première machine supérieur au temps opératoire sur la deuxième machine ;
4 Trier l'ensemble E1 dans l'ordre croissant des temps opératoires sur la première machine ;
5 Trier l'ensemble E2 dans l'ordre décroissant des temps opératoires sur la deuxième machine ;
6 L'ordre des fabrications est déterminé par l'ordre de E1 trié puis de E2 trié ;
7 **Fin**

Exemple 1 :

L'exemple ci-dessous illustre l'algorithme de Johnson, soit 4 tâches T_1, T_2, T_3 et T_4 et leurs temps d'exécution respectifs sur les deux machinesM_1, et M_2.

machines/tâches	T_1	T_2	T_3	T_4
M_1	1	3	8	5
M_2	4	2	7	6

On affecte les différents tâches à la liste E1 et E2 tout en respectant l'algorithme de johnson.
On aura :

— La liste E1 va contenir les tâches : T_1,T_4.

— La liste E2 va convertir les tâches : T_2, T_3

L'ordre final sera alors T_1, T_4, T_3, T_2

Le diagramme de Gantt associé à cet exemple est illustré par la figure suivante :

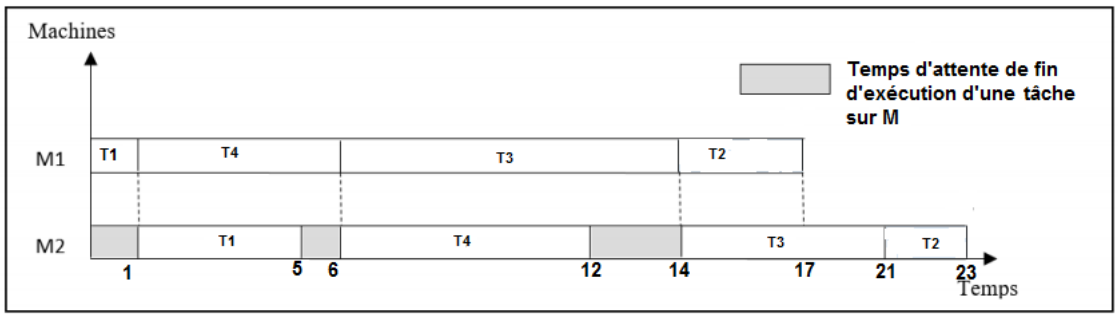


FIGURE 2.1: Solution optimale de l'exemple 1

2.6 Problème F2 /recr(2) /C_{max}

2.6.1 Description du problème :

Traiter un ensemble des tâches indépendantes non interruptibles ayant des temps de traitement arbitraire et des dates de disponibilités nulles avec recirculation des tâches sur la deuxième machine $M_1 \rightarrow M_2 \rightarrow M_2$, sur deux machines spécialisées dans un système flow shop, l'objectif est la minimisation de la date de fin de traitement.

Le problème précédent est illustré dans la figure suivante :

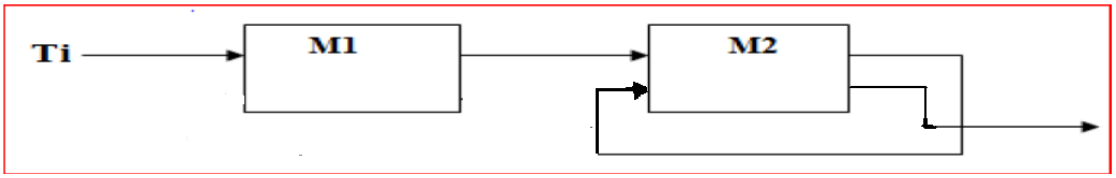


FIGURE 2.2: Flow shop à deux machines avec recirculation des tâches sur la deuxième machine

Meziani Nadjet a prouvée que ce problème est polynomial et propose l'algorithme suivant pour le résoudre :

2.6.2 Algorithme F2/recr(2)/C$_{max}$

Algorithm 2: Algorithme du $F2/recr(2)/C_{max}$ [3]

1 **Début**

2 Transformer le problème considéré au problème $F2//C_{max}$ comme suit : Sur la première machine les temps de traitement ne change pas $p'_{1i} = p_{1i}$; Sur la deuxième machine, les différentes opérations d'une même tâche forment une seul opération tel que $p'_{2i} = p_{2i1} + p_{2i2}$;

3 Résoudre le problème $F2//C_{max}$ avec l'algorithme de Johnson ;

4 Construire la solution du problème initial $F2/recr(2)/C_{max}$ en partageant la durée de traitement de chaque tâche sur la deuxième machine en durées de traitement de ses opérations élémentaires initiales ;

5 **Fin**

Exemple 2 :

Soit 4 tâches T_1, T_2, T_3 et T_4 à ordonnancer sur deux machines dans un système flow shop, les temps d'exécution respectifs sur les deux machines M_1, et M_2 avec recirculation sur la machine M_2 sont donnés dans le tableau suivant :

machines/tâches	T_1	T_2	T_3	T_4
P_{1i}	7	4	3	1
P_{2i1}	4	2	1	5
P_{2i2}	1	2	6	3

L'algorithme donne la solution optimale suivante avec $C_{max} = 31$

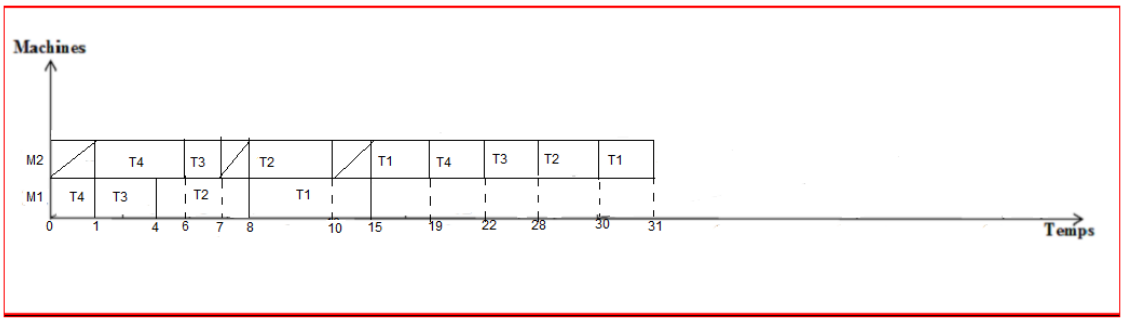


FIGURE 2.3: Solution optimale de l'exemple 2

2.7 Problème F2/coup-opr(1),ai,bi,Li,ci/C$_{max}$

2.7.1 Description du problème :

Ce problème est un problème de flow shop à deux machines où $T = \{T_i, i = \overline{1,n}\}$ est l'ensemble de n tâches à traiter sur les deux machines.

— Chaque tâche est composée d'un couple d'opération O_{1i} sur la première machine et d'une seul opération sur la deuxième machine O_{2i}.

— Chaque couple d'opération de tâches O_{1i} est noté par le triplet (ai, Li, bi) telles que :

ai et bi : sont les durées de traitement de la première et de la deuxième opération du couple d'opération O_{1i}.
Li : est le délai qui s'écoule entre les dates de fin et début de traitement de la première et de la deuxième opération.
ci : est le temps d'exécution de l'opération O_{2i} qui ne peut être traiter que si le traitement du couple d'opération O_{1i} est terminer.

— L'objectif est de minimiser la date de fin de traitement des tâches (makespan).

2.7.2 Résultats de complexité

N.Meziani, A.Oulamara et M.Boudhar dans leurs article [17], ont considéré le problème $F2/coup - opr(1), ai, bi = Li = p, ci/C_{max}$, ils montrent que ce problème est NP-Difficile, en utilisent le problème de réductions avec cardinalités identiques connu NP-difficile.
Les auteurs ont prouvé la difficulté de ce problème à partir des deux lemmes suivants :
Lemme 1 : si le problème PCI (problème de partition avec cardinalité identique) a une solution alors il existe un ordonnancement S avec $C_{max}(s) \leq y$ où y= 4(2n+1)p+1.
Lemme 2 : si le problème d'ordonnancement admet une solution avec $C_{max} \leq y$ alors le problème (PCI) a une solution.

Ils ont prouvé aussi les deux Lemmes précédents et trouvent le résultat dans le théorème suivant :

Théorème 2.7.1 *Le problème $F2/coup - opr(1), ai, bi = Li = p, ci/C_{max}$ est NP-difficile.*

Ils présentent aussi des cas particuliers du problème précédent on peut les résoudre en O(n log n), par des algorithmes proposent par eux.

2.8 Problème F2/recr(1)/C$_{max}$

2.8.1 Description du problème

Traiter un ensemble des tâches indépendantes non interruptibles ayant des temps des traitement arbitraire et des dates de disponibilités nulles avec recirculation des tâches sur la première machine, chaque tâche doit être exécuter selon l'ordre $M_1 \rightarrow M_2 \rightarrow M_1$, sur deux machines spécialisées dans un système flow shop, l'objectif est la minimisation de la date de fin de traitement.

— Une machine ne peut exécuter qu'une seule opération à la fois.

— Une opération ne peut être exécuter par plus d'une machine. Le problème précédent est illustré dans la figure suivante :

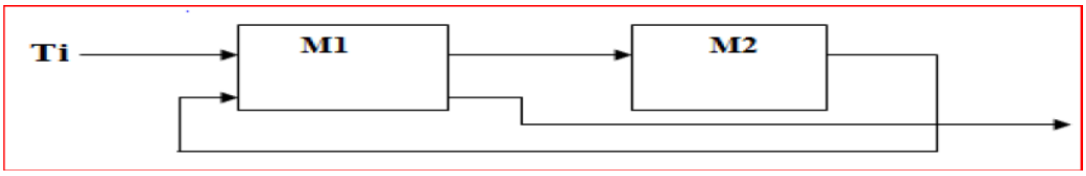


FIGURE 2.4: Flow shop à deux machines avec recirculation des tâches sur la première machine

Pour un ordonnancement quelconque des tâches notons :

— P_{1i1} : la durée d'exécution de la première opération de la tâche T_i sur la première machine.

— P_{1i2} : la durée d'exécution de la deuxième opération de la tâche T_i sur la première machine.

— P_{2i} : la durée d'exécution de la tâche T_i sur la deuxième machine.

— S_{1i1} : la date de début d'exécution de la première opération de la tâche T_i sur la première machine.

— S_{1i2} : la date de début d'exécution de la deuxième opération de la tâche T_i sur la première machine.

— S_{2i} : la date de début d'exécution de la tâche T_i sur la deuxième machine.

Exemple illustratif

Pour illustrer le problème, considérons l'instance suivante : disposons de 6 tâches indépendantesT_1, T_2, T_3, T_4, T_5 et T_6 à traiter sur deux machines. Les durées de traitement de ces 6 tâches sur les deux machines sont données dans le tableau suivant :

Ti	T1	T2	T3	T4	T5	T6
P_{1i1}	2	3	2	1	3	2
P_{2i}	2	3	1	1	4	1
P_{1i2}	1	2	2	1	1	4

Un ordonnancement est donné dans la figure suivante :

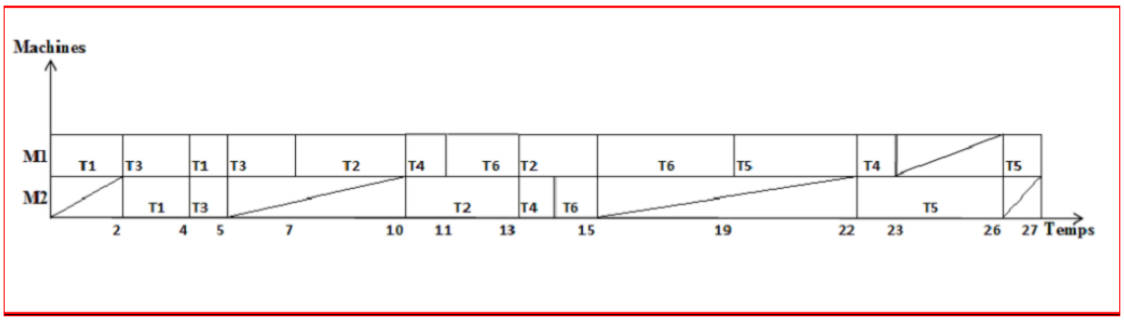


FIGURE 2.5: Solution de l'exemple précédent

2.8.2 Complexité du problème

M.Y.Wang, et al. dans leurs articles [20], considèrent le problème de flow shop à m machines avec recirculation des tâches sur la première machine $(M_1, M_2, ..., M_m, M_1)$ noté $Fm/chain{-}reentrant/C_{max}$ et ils proposent le théorème suivant :

Théorème 2.8.1 *(M.Y.Wang, et al.) [20] le problème $Fm/chain - reentrant/C_{max}$ est NP-difficile pour m=2 et NP-dure pour $m \geq 3$.*

Remarque :
Pour m=2 le Problème $F2/recr(1)/C_{max}$ est équivalent à $Fm/chain - reentrant/C_{max}$ qui est NP-difficile. Alors notre problème est NP-difficile. (Pour plus de détails consulter) [20, 15].

2.9 Problème F2/r$_i$/C_{max}

2.9.1 Description du problème

Traiter un ensemble des tâches indépendantes sans préemption ayant des temps de traitement et des dates de disponibilité arbitraires sur deux machines spécialisées dans un système flow shop. L'objectif est de minimiser la date de fin de traitement des tâches .

2.9.2 Complexité du problème

Théorème 2.9.1 *(Lenstra et al., 1977) Le problème $F2/r_i/C_{max}$ est NP-complet [10].*

Preuve

La réduction est faite à partir du problème de la somme des sous ensembles (SUBSET SUM), c'est un cas particulier du problème de sac-à-dos, qui est NP-complet .
INSTANCE : Un entier V, et k entiers positifs v_i : $i \in \tau = \{1, 2, ..., k\}$.

QUESTION : Existe-t-il un sous-ensemble $S \subset \tau$ sur lequel $\sum_{i \in S} v_i = V$?
au problème de décision de notre problème :$F2/r_i/C_{max}$
INSTANCE : Un entier B, et n tâches $T_i : i \in N = \{1, 2..., n\}$, chacun avec des paramètres $< a_i, b_i, r_i >$, à ordonnancer sur deux machines dans un système flow shop, où a_i, b_i est le temps de traitement sur M_1, M_2 respectivement, et r_i est la date de disponibilité de la tâche T_i.
QUESTION : Existe-t-il un ordonnancement avec $C_{max} \leq B$?
Pour toute instance de Subset Sum, on définit une instance du problème $F2/r_i/C_{max}$ comme suit :

— B := W+1, où W = $\sum_i V i$

— n = k+1

— $< a_i, b_i, r_i >$:= $< v_i, 0, 0 >$, $i \in \tau$

— $< a_n, b_n, r_n >$:= $< 1, W - V, V >$

Pour voir pourquoi les deux problèmes auront toujours la même réponse, il suffit de noter que nous ne pouvons un makespan de B (la réponse "oui") que si T_i est ordonnancé immédiatement à son arrivée, et si les autres tâches sont ordonnançables sur M_1 dans les deux intervalles définis par le positionnement de T_i. (Pour plus de détails voir [**?**])

2.10 Conclusion

Dans ce chapitre nous avons présenté quelques problèmes d'ordonnancement sur deux machines dans un système flow shop comme les problèmes faciles, nous avons présenté des algorithmes pour résoudre ce type de problème ainsi que quelques exemples illustratifs, nous avons présenté aussi des problèmes difficiles et la preuve de leurs complexité.

CHAPITRE

3

HEURISTIQUES PROPOSÉES POUR RÉSOUDRE LE PROBLÈME $F2/R_I/C_{MAX}$

3.1 Introduction

Dans ce chapitre nous présentons le problème d'ordonnancement $F2/r_i/C_{max}$ en vue de minimiser la date de fin de traitement, en proposant 3 heuristiques pour ce problème qui est NP-difficile.

3.2 Représentation du problème F2/r$_i$/C_{max}

Toujours avec l'objectif de la minimisation de makespan, supposons que les tâches ne sont pas tous disponibles simultanément, mais ont des dates de disponibilité r_i différentes.
Le problème $F2/r_i/C_{max}$ consiste à ordonnancer un ensemble des Tâches sur deux machines spécialisées dans un système flow shop avec dates de disponibilité.
Ce problème est NP-difficile (voir chapitre 2).

3.3 Heuristiques pour résoudre le problème F2/r$_i$/C_{max}

3.3.1 Définition

Les heuristiques sont des méthodes empiriques qui donnent généralement de bonnes résultats sans pour autant être démontrables. Elles se basent sur des règles simplifies pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable [1].

3.3.2 Les heuristiques proposées

Dans cette session nous proposons des heuristiques lesquelles fournissent rapidement une solution réalisable proche de l'optimum pour le problème d'ordonnancement $F2/r_i/C_{max}$ ou notre but est de trouver l'heuristique de bonne qualité.

Pour cela nous avons les notations suivantes :
t_{1i} : la date de début de traitement de la tâche en $i^{ème}$ position sur la première machine.
C_{1i} : la date de fin de traitement de la tâche en $i^{ème}$ position sur la première machine.
C_{2i} : la date de fin de traitement de la tâche en $i^{ème}$ position sur la deuxième machine.

3.3.2.1 A- Principe de la première heuristique

La première phase de cette heuristique consiste à ranger les tâches T_i par ordre croissant de leurs date de disponibilité.

Algorithm 3: Algorithme de l'heuristique H1

1 **Début**
2 Ranger les tâches par ordre croissant de leurs date de disponibilité dans une liste L ;
3 $t_1 = r_1$;
4 $C_{11} = t_{11} + p_{11}$;
5 **pour i=2 à n faire**
6 $t_{1i} = \max\{C_{1(i-1)}, r_i\}$;
7 $C_{1i} = t_{1i} + p_{1i}$;
8 **Fait**
9 Ordonnancer les tâches dans le même ordre sur la deuxième machine ;
10 $C_{max} = C_{2i}$;
11 **Fin.**

3.3.2.2 B- Principe de la deuxième heuristique

L'idée de cette heuristique est de transformer le problème $F2/r_i/C_{max}$ au problème $F2//C_{max}$, on suppose que les dates de disponibilité des tâches sont nulles, ensuite en résoudre le problème $F2//C_{max}$ avec l'algorithme de Johnson et à la fin on extraire la solution de notre problème.

Algorithm 4: Algorithme de l'heuristique H2

1. **Début**
2. Supposer que toutes les dates de disponibilités des tâches sont nulles $r_i = 0, i = \overline{1,n}$;
3. Utiliser l'algorithme de Johnson pour trouver la solution ;
4. Construire une liste L selon la solution trouvée ;
5. Ordonnancer les tâches suivant la liste L ,en respectant les dates de disponibilité de chaque tâche sur la première machine ;
6. Ordonnancer les tâches dans le même ordre sur la deuxième machine ;
7. $C_{max} = C_{2i}$;
8. **Fin.**

3.3.2.3 C- Principe de la $3^{ème}$ heuristique

La première phase de cette heuristique consiste à trouver les tâches disponible à l'instant $t_i = \max\{C_{(i-1)}, \min\{r_i\}\}$, ensuite ordonnancer la tâche T_i ayant la plus petit temps de traitement sur la première machine.

Algorithm 5: Algorithme de l'heuristique H3

1. **Début**
2. $C_{10} = 0$;
3. **Pour k=1 à n faire**
4. $T' = \{ T_i \in T / r_i \leq t_{1k}, t_{1k} = \max\{C_{1(k-1)}, \min\{r_i\}\}$;
5. Choisir $T_i \in T'$ ayant $\min\{p_{1i}\}$;
6. Ordonnancer T_i à l'instant t_{1k} ;
7. $T = T/T_i$;
8. $C_{1k} = t_{1k} + p_{1i}$;
9. **Fait**
10. Ordonnancer les tâches dans le même ordre sur la deuxième machine ;
11. $C_{max} = C_{2k}$;
12. **Fin.**

3.3.3 Exemple d'application

Dans cette session nous désirons traiter 12 tâches indépendantes sur deux machines dans un système flow shop, dont les temps de traitement et les dates de disponibilité sont donnés dans le tableau suivant :

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
r_i	1	18	7	2	5	8	4	12	6	20	17	9
P_{1i}	3	2	4	6	5	6	1	2	5	4	7	8
P_{2i}	1	4	3	5	3	1	7	5	8	1	2	2

Nous appliquons cet exemple sur les trois heuristiques proposées précédemment.

3.3.3.1 A- Application de la $1^{ère}$ heuristique

Nous appliquons notre heuristique H1 : Ranger les tâches par ordre croissant de leurs dates de disponibilité on trouve la liste L suivante :

L=$\{T_1, T_4, T_7, T_5, T_9, T_3, T_6, T_{12}, T_8, T_{11}, T_2, T_{10}\}$

- **Itération 0** :

  $t_{11} = r_1 = 1 \longrightarrow C_{11} = t_{11} + p_{11} = 4$

  Itération 01 :

  $t_{12} = \max\{C_{11}, r_2\} = \max\{4, 2\} = 4 \longrightarrow C_{12} = t_{12} + p_{12} = 10$

- **Itération 02** :

  $t_{13} = \max\{10, 4\} = 10 \longrightarrow C_{13} = t_{13} + p_{13} = 11$

- **Itération 03** :

  $t_{14} = \max\{11, 5\} = 11 \longrightarrow C_{14} = t_{14} + p_{14} = 16$

- **Itération 04** :

  $t_{15} = \max\{16, 6\} = 16 \longrightarrow C_{15} = t_{15} + p_{15} = 21$

- **Itération 05** :

  $t_{16} = \max\{21, 7\} = 21 \longrightarrow C_{16} = t_{16} + p_{16} = 25$

- **Itération 06** :

  $t_{17} = \max\{25, 8\} = 25 \longrightarrow C_{17} = t_{17} + p_{17} = 31$

- **Itération 07** :

  $t_{18} = \max\{31, 9\} = 31 \longrightarrow C_{18} = t_{18} + p_{18} = 39$

- **Itération 08** :

  $t_{19} = \max\{39, 12\} = 39 \longrightarrow C_{19} = t_{19} + p_{19} = 41$

- **Itération 09** :

  $t_{110} = \max\{41, 17\} = 41 \longrightarrow C_{110} = t_{110} + p_{110} = 48$

- **Itération 10** :

  $t_{111} = \max\{48, 18\} = 48 \longrightarrow C_{111} = t_{111} + p_{111} = 50$

- **Itération 11** :

  $t_{112} = \max\{50, 20\} = 50 \longrightarrow C_{112} = t_{112} + p_{112} = 54$

nous ordonnançons les tâches dans le même ordre sur la deuxième machine, nous avons trouvé la solution présentée par la figure 3.1 avec $C_{max} = C_{212} = 55$.

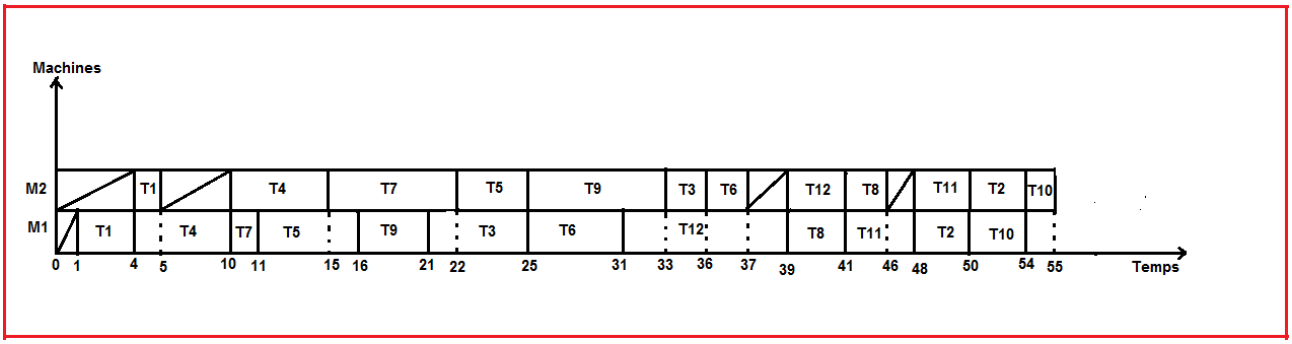


FIGURE 3.1: Solution générale de l'exemple par l'heuristique H1

3.3.3.2 B- Application de la $2^{ème}$ heuristique

On affecte les différents tâches à la liste E1 et E2 tout en respectant l'algorithme de Johnson. On aura :

— La liste E1 va contenir les tâches : T_2, T_7, T_8, T_9.

— La liste E2 va convertir les tâches : $T_1, T_3, T_4, T_5, T_6, T_{10}, T_{11}, T_{12}$.

L'ordre final sera alors L= $\{T_7, T_2, T_8, T_9, T_4, T_3, T_5, T_{11}, T_{12}, T_1, T_6, T_{10}\}$.

— **Itération 0** :

$t_{11} = r_1 = 4 \longrightarrow C_{11} = t_{11} + p_{11} = 4 + 1 = 5$

— **Itération 01** :

$t_{12} = \max\{C_{11}, r_2\} = \max\{5, 18\} = 18 \longrightarrow C_{12} = t_{12} + p_{12} = 18 + 2 = 20$

— **Itération 02** :

$t_{13} = \max\{C_{12}, r_3\} = \max\{20, 12\} = 20 \longrightarrow C_{13} = t_{13} + p_{13} = 20 + 2 = 22.$

— **Itération 03** :

$t_{14} = \max\{C_{13}, r_4\} = \max\{22, 6\} = 22 \longrightarrow C_{14} = t_{14} + p_{14} = 22 + 5 = 27.$

— **Itération 04** :

$t_{15} = \max\{C_{14}, r_5\} = \max\{27, 2\} = 27 \longrightarrow C_{15} = t_{15} + p_{15} = 27 + 6 = 33.$

— **Itération 05** :

$t_{16} = \max\{C_{15}, r_6\} = \max\{33, 7\} = 33 \longrightarrow C_{16} = t_{16} + p_{16} = 33 + 4 = 37.$

— **Itération 06** :

$t_{17} = \max\{C_{16}, r_7\} = \max\{37, 5\} = 37 \longrightarrow C_{17} = t_{17} + p_{17} = 37 + 5 = 42.$

— **Itération 07** :

$t_{18} = \max\{C_{17}, r_8\} = \max\{42, 17\} = 42 \longrightarrow C_{18} = t_{18} + p_{18} = 42 + 7 = 49.$

— **Itération 08** :

$t_{19} = \max\{C_{18}, r_9\} = \max\{49, 9\} = 49 \longrightarrow C_{19} = t_{19} + p_{19} = 49 + 8 = 57.$

— **Itération 09** :

$t_{110} = \max\{C_{19}, r_{10}\} = \max\{57, 1\} = 57 \longrightarrow C_{110} = t_{110} + p_{110} = 57 + 3 = 60.$

— **Itération 10** :

$t_{111} = \max\{C_{110}, r_{11}\} = \max\{60, 8\} = 60 \longrightarrow C_{111} = t_{111} + p_{111} = 60 + 6 = 66.$

— **Itération 11** :

$t_{112} = \max\{C_{111}, r_{10}\} = \max\{66, 20\} = 66 \longrightarrow C_{112} = t_{112} + p_{112} = 66 + 4 = 70.$

nous ordonnançons les tâches dans le même ordre sur la deuxième machine, nous avons trouvé la solution présentée par la figure 3.2 avec $C_{max} = C_{212} = 71$.

Le diagramme de Gantt associé à cet exemple est illustré par la figure suivante :

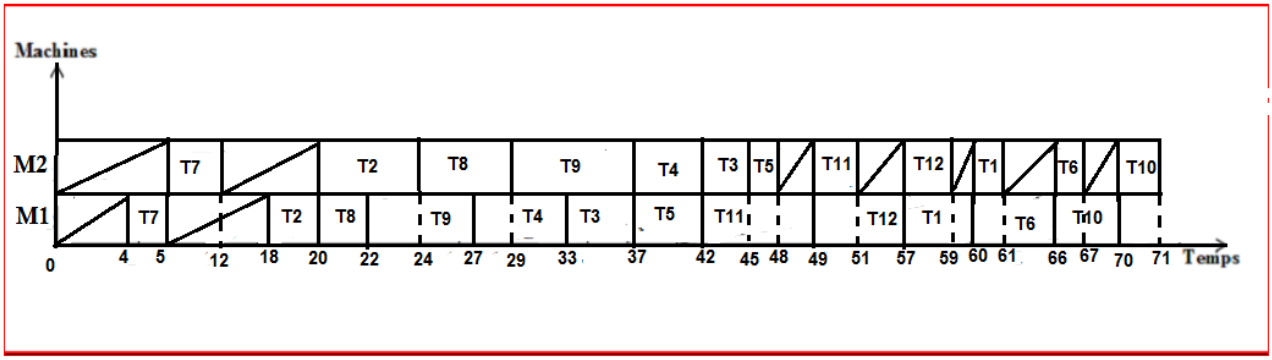


FIGURE 3.2: Solution générale de l'exemple par l'heuristique H2

3.3.3.3 C- Application de la $3^{ème}$ heuristique

Nous appliquons notre heuristique H3 :

— **Itération 0** :

$t_{11} = \min\{r_i\} = 1, T' = \{T_1\}, F = \{T_1\}, T = T/T_1, c_{11} = t_{11} + p_{11} = 1 + 3 = 4.$

— **Itération 01** :

$t_{12} = \max\{c_{11}, \min\{r_i\}\} = \max\{4, 2\} = 4, T' = \{T_4, T_7\}, F = \{T_1, T_7\}, T = T/T_7,$

$C_{12} = t_{12} + p_{12} = 4 + 1 = 5.$

— **Itération 02** :

$t_{13} = \max\{c_{12}, \min\{r_i\}\} = \max\{5, 2\} = 5, T' = \{T_4, T_5\}, F = \{T_1, T_7, T_5\}, T = T/T_5, ;$

$C_{13} = t_{13} + p_{13} = 5 + 5 = 10.$

— **Itération 03** :

$t_{14} = \max\{c_{13}, \min\{r_i\}\} = \max\{10, 2\} = 10, T' = \{T_4, T_3, T_6, T_9, T_{12}\}, F = \{T_1, T_7T_5, T_3\}$,
$T = T/T_3, C_{14} = t_{14} + p_{14} = 10 + 4 = 14$.

— **Itération 04** :

$t_{15} = \max\{C_{14}, \min\{r_i\}\} = \max\{14, 2\} = 14, T' = \{T_4, T_6T_9, T_8, T_{12}\}$, $F = \{T_1, T_7T_5, T_3, T_8\}$,
$T = T/T_8, C_{15} = t_{15} + p_{15} = 16$.

— **Itération 05** :

$t_{16} = \max\{C_{15}, \min\{r_i\}\} = \max\{16, 2\} = 16, T' = \{T_4, T_6T_9, T_{12}\}$, $F = \{T_1, T_7T_5, T_3, T_8, T_9\}$,
$T = T/T_9, C_{16} = t_{16} + p_{16} = 16 + 5 + 21$.

— **Itération 06** :

$t_{17} = \max\{C_{16}, \min\{r_i\}\} = \max\{21, 2\} = 21, T' = \{T_4, T_6T_2, T_{11}, T_{12}\}, F = \{T_1, T_7T_5, T_3, T_8, T_9, T_2\}$,
$T = T/T_2, C_{17} = t_{17} + p_{17} = 21 + 2 = 23$.

— **Itération 07** :

$t_{18} = \max\{C_{17}, \min\{r_i\}\} = \max\{23, 2\} = 23, T' = \{T_4, T_6, T_{10}, T_{11}, T_{12}\}$,
$F = \{T_1, T_7T_5, T_3, T_8, T_9, T_2, T_4\}, T = T/T_4, C_{18} = t_{18} + p_{18} = 23 + 6 = 29$.

— **Itération 08** :

$t_{19} = \max\{C_{18}, \min\{r_i\}\} = \max\{29, 8\} = 29, T' = \{, T_6, T_{10}, T_{11}, T_{12}\}$,
$F = \{T_1, T_7T_5, T_3, T_8, T_9, T_2, T_4, T_6\}, T = T/T_6, C_{19} = t_{19} + p_{19} = 29 + 6 = 35$.

— **Itération 09** :

$t_{110} = \max\{C_{19}, \min\{r_i\}\} = \max\{35, 9\} = 35, T' = \{T_{10}, T_{11}, T_{12}\}$,
$F = \{T_1, T_7T_5, T_3, T_8, T_9, T_2, T_4, T_6, T_{10}\}, T = T/T_{10}, C_{110} = t_{110} + p_{110} = 35 + 4 = 39$.

— **Itération 10** :

$t_{111} = \max\{C_{110}, \min\{r_i\}\} = \max\{39, 9\} = 39, T' = \{T_{11}, T_{12}\}$,
$F = \{T_1, T_7T_5, T_3, T_8, T_9, T_2, T_4, T_6, T_{10}, T_{11}\}, T = T/T_{11}, C_{111} = t_{111} + p_{111} = 39 + 7 = 46$.

— **Itération 11** :

$t_{112} = \max\{c_{111}, \min\{r_i\}\} = \max\{46, 17\} = 46, T' = \{T_{12}\}$,
$F = \{T_1, T_7T_5, T_3, T_8, T_9, T_2, T_4, T_6, T_{10}, T_{11}, T_{12}\}, T = T/T_{12}, C_{112} = t_{112} + p_{112} = 46 + 8 = 54$.

nous ordonnançons les tâches dans le même ordre sur la deuxième machine, nous avons trouvé la solution présentée par la figure 3.3 avec $C_{max} = C_{212} = 56$.

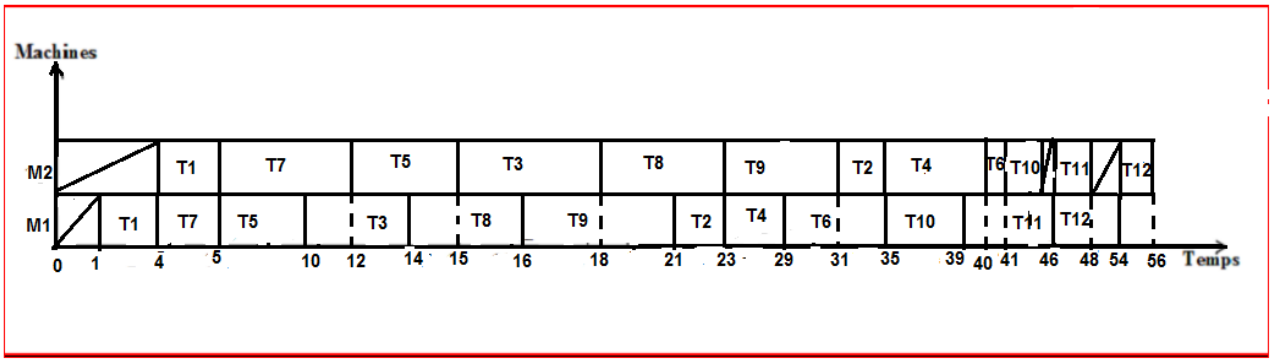


FIGURE 3.3: Solution générale de l'exemple par l'heuristique H3

3.3.3.4 Comparaison entre les heuristiques proposées

Pour comparer les heuristiques proposées, nous avons calculer le ration d'approximation R avec :

$$R = \frac{C_{max}(H)}{Q} \tag{3.1}$$

$C_{max}(H)$: C'est la longueur d'ordonnancement obtenue à partir de l'heuristique proposée H.
Q : est la borne inférieur proposée du problème $F2/r_i/C_{max}$ avec :

$$Q = \min\{r_i\} + \sum_{i=1}^{n} \{p_{1i}\} + \min\{p_{2n}\}. \tag{3.2}$$

Chaque une des heuristiques a un principe bien déterminer qui se base soit sur l'ordonnancement par liste ou bien l'ordonnancement par tâche, donc la longueur d'ordonnancement de notre problème dépend de la liste de priorité, les dates de disponibilité et des temps de traitement des tâches.

Lorsque R est proche de 1 alors l'approximation est de bonne qualité.
Calculons alors la borne inférieure de notre exemple :

$$Q = \min\{r_i\} + \sum_{i=1}^{n} \{p_{1i}\} + \min\{p_{2n}\} = 1 + 53 + 1 = 55.$$

Pour l'heuristique H1

On a $C_{max} = 55$ et donc $R = \frac{C_{max}(H1)}{Q} = \frac{55}{55} = 1$.
Alors cette heuristique est de bonne qualité et optimale pour cet exemple car R = 1.

Pour l'heuristique H2

On a $C_{max} = 71$ et donc $R = \frac{C_{max}(H2)}{Q} = \frac{71}{55} = 1.2909$.
Alors cette heuristique est de mauvaise qualité car R = 1.2909 est loin de 1.

Pour l'heuristique H3

On a $C_{max} = 56$ et donc $R = \frac{C_{max}(H3)}{Q} = \frac{56}{55} = 1.0181$.
Alors cette heuristique est de bonne qualité car R = 1.0181 est proche de 1.

A partir des résultats obtenues, nous remarquons que la $1^{ère}$ heuristique donne la plus petite valeur de la longueur d'ordonnancement, telle que le quotient R égale à 1 donc elle est optimale. ainsi nous avons constaté que la $3^{ème}$ heuristique donne de meilleur solution que la $2^{ème}$ heuristique.

Remarque 3.3.1 *Nous avons traité les trois heuristiques sur un seul exemple, nous avons remarqué que l'heuristique H1 donne une solution optimale que les heuristiques H2 et H3 pour cet exemple, mais pas forcement pour d'autres exemples.*

3.4 Conclusion

Nous traitons dans ce chapitre le problème d'ordonnancement $F2/r_i/C_{max}$, nous avons commencé par la définition de ce problème ainsi nous proposons trois heuristiques pour trouver une solution approchée de ce dernier.

CONCLUSION GÉNÉRALE

Les problèmes d'ordonnancement sont présentés dans tous les secteurs de l'économie et constituent une fonction important en gestion de production. Un problème d'ordonnancement consiste à allouer dans le temps des tâches à des ressources qui existent en quantité limité, tout en satisfaisant un ensemble de contraintes.

Dans ce mémoire nous avons étudié un cas particulier de problème d'atelier, qui est l'ordonnancement FLOW-SHOP dont on a évalué un critère qui est la minimisation de la durée de fin d'exécution C_{max} de toutes les tâches dans un atelier.

Dans le premier chapitre nous avons mené une recherche bibiographique portant sur des problèmes d'ordonnancement, leurs notions fondamentales et les éléments nécessaires pour caractériser un problème d'ordonnancement sont aussi évoqués : les tâches, les ressources, les contraintes et les objectifs, La deuxième point nous avons abordé la classification, le diagramme de Gantt comme un outil de représentation de ces problèmes, ainsi que nous discutons sur quelques concepts de la théorie de complexité.

Nous nous somme intéressé dans la deuxième partie à étudier quelques problèmes d'ordonnancement sur deux machines qui peuvent être classés comme des problèmes faciles et difficiles, nous avons commencer par les problèmes faciles, nous avons exposé quelques problèmes et nous avons donné les algorithmes pour leurs résolution, en suite en terminer par les problèmes difficiles et leurs complexité .

Dans la dernière partie, nous avons proposé trois heuristiques du problème $F2/r_i/C_{max}$ qui est NP-difficile, tout d'abord nous avons rappeler sa définition, nous avons remarqué que l'heuristique H1 donne une solution optimale, que l'heuristique H3 est meilleure que H2 pour notre exemple, mais pas forcement pour d'autres exemples.

BIBLIOGRAPHIE

[1] **H.AJIRI**, *«Les problèmes d'ordonnancement d'atelier : M-Machine identique en parallèle »*, Mémoire présenté pour l'obtention Du diplôme de Master Académique, Université Mohammed Boudiaf M'sila, 2017 /2018.

[2] **F.AMOUBOUDI & K.CHEBBAH**, *«Etude comparative de quelques heuristiques pour la résolution du problème d'ordonnancement de type Flow shop »*, Mémoire de Master, Université Abderrahmane Mira de Béjaia, 2015/2016.

[3] **K.AMROUCHE**, *« Flow shop à deux machines avec recirculation»*, Mémoire de Master, Université des sciences et de la technologie Houari Boumedienne, 24/10/ 2010.

[4] **M.BELKHOUS & S.BOURAHLA**, *«Modélisation d'un processus de fabrication et résolution d'un problème d'ordonnancement de type Flow-Shop. Cas «ENIEM».»* , Université de Mouloud Mammeri, Tizi-Ouzou, 2016/2017.

[5] **M.BENSADIA & A.HAMANI** , *«Problème d'ordonnancement avec dates échues sur une seule machine»* , Mémoire de Master, Université de Médéa, 2020.

[6] **A.BEY MASTAPHA & K. BOUKENDAKDJI**, *«Problème de minimisation de la longueur d'ordonnancement sur machines parallèles identiques »* , Mémoire de Master, Université de Médéa, 2018/2019.

[7] **H.BOUKEF BEN OTHMAN** , *«sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutique optimisation par algorithmes génétiques et essaims particulaires »* , Thèse de Doctorat, Université de Tunis el manar école nationale d'ingénieurs de tunis, 3 juillet 2009.

[8] **A.CHERGUI & A.DAHMANI & ABDELLAH ADDA ABBOU** , *«Ordonnancement d'un flow-shop par métaheuristique hybride»* , Mémoire de Master, Université Abou Bekr Belkaid – Tlemcen, 2016/2017.

[9] **J.P.FONTAINE**, *«L'ordonnancement des opérations est stratégique»*, www.arima.com, consulté le 14/02/2019.

[10] **E.HAMILTON & G.VAIRAKTARAKIS**, « *Flow shop Scheduling Theoretical Results, Algorithms, and Applications*», Springer Science+Business Media New York, 2013.

[11] **G.JAVEL** , « *Organisation et gestion de la production*», 2000.

[12] **A.KARRAY**, «*Contribution à l'ordonnancement d'atelier agroalimentaires utilisant des méthode d'optimisation hybrides*» ,Thèse de Doctorat ,Ecole centrale de lille, Université de Tunis el manar école nationale d'ingénieurs de Tunis ,5 juillet 2011.

[13] **S.KOUIDER AIAD**, «*Note de cours, Formulation des problèmes d'ordonnancement*»,pour Master 2 Recherche opérationnelle, Université de Médéa, 2021.

[14] **I.LARIBI**, «*Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables*» , Thèse Présentée pour l'obtention du grade de DOCTEUR, Université de Tlemcen, 12 décembre 2018.

[15] **J.K.LENSTRA & A.H.G.RINOOY KAN**, «*Complexity of machine scheduling problems*», Annals of Discrete Mathimatics, 1997.

[16] **M. E. A.MEZIANE**, « *Optimisation par phases pour les problèmes d'ordonnancement des ateliers de type job-shop totalement flexibles*», Mémoire de Magister, Université d'Oran, Algérie 2011.

[17] **N.MEZIANI & A.OULAMARA & M.BOUDHAR**, « *NP-complétude du problème du flow shop à deux machines avec des opération couplées*», $X^{ème}$ conférence internationale :conception et production intégrées, Tanger,maroc.hal-01260752, 12/ 2015.

[18] **I. H. OSMAN & G.LAPORTE** , « *Metaheuristics*», A bibliography, 1996.

[19] **W.TRABELSI** , « *Ordonnancement des systèmes de production flexibles soumis à différents types de contraintes de blocage*», Thèse de Doctorat, Ecole doctorale IAEM Lorraine, 2012.

[20] **M.Y.WANG & S.P.SETHI & S.L.Van De Velde**, « *Minimizing Makespan in a classe of reentrant shops* », Oper.Res, 1997.

[21] **Wikipédia** , «*Ordonnancement d'atelier* » , fr.m.wikipedia.org, consulté le 12.04.2021.

[22] **Wikipédia**, «*Flow shop*», fr.wikipedia.org, consulté 12.04.2021.