Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Université Dr.Yahia Farès de Médéa

Faculté des Sciences -Département de Mathématiques & Informatique



MÉMOIRE

présenté en vue de l'obtention

du

Diplôme de Master en Mathématiques
Spécialité : Recherche Opérationnelle

Par

MEBROUKI Mohamed Amine et GASTI Mohamed

Thème

Etude de certains problèmes d'ordonnancement avec temps de préparation

Soutenu devant le jury composé de :

M^r	N. IKHLEF ESCHOUF	Professeur	UYF Médéa	Président
M^r	A. BENKACI	Maitre assistant	UYF Médéa	Examinateur
M^r	R. LEMDANI	Maitre de conférences	UYF Médéa	Examinateur
M^{me}	S. KOUIDER AIAD	Maitre assistant	UYF Médéa	Promotrice

Juin 2022

Remerciements

Tous d'abord, nous tiens à remercier Allah clément et miséricordieux de notre avoir donné la force et le courage de mener à terme ce modeste travail. Comme, c'est un plaisir de remercier tous ce qui ont contribué à la réalisation de ce mémoire :

En premier lieu, nous voudrons exprimer nos remerciements les plus sincères à notre promotrice **Mme S.KOUIDER AIAD** , pour nous avoir guidés pour la réalisation de ce mémoire et le soutient scientifique et moral.

Monsieur **N. IKHLEF ECHOUF** Professeur à l'Université Dr.Yahia Farés de Médéa, de nous avoir fait l'honneur de présider le jury de ce mémoire.

Nous tenons à remercier Monsieur**A. BENKACI** Maitre assistant à l'Université Dr.Yahia Farés de Médéa et Monsieur **R. LEMDANI** Maitre de conférences à l'Université Dr.Yahia Farés de Médéa pour nous avoir fait l'honneur d'être membres du jury. Ainsi que pour avoir consacré une partie de leur temps précieux pour lire et corriger ce mémoire.

Nous devons remercier si particulièrement nos très chers parents pour leur soutien et l'encouragement qu'ils nous ont donné afin de continuer nos études

Enfin nous remercions toutes les personnes qui ont aidé de prés ou de loin à la réalisation de ce projet.

Dédicaces

À la plus belle créature qu'Allah a créée sur terre..

Je dédie ce travail :

À mes très chers parents : Mon Père "Said", Ma Mère "Fatima zohra"qui m'ont guidé durant les moments les plus pénibles de ce long chemin, mes chers parents qui ont sacrifié toute leur vie afin de me voir devenir ce que je suis, Maman j'adore ,Papa j'adore.

À ma soeur : "Maroua"

À mes frères : "Yacine", "Hamza Rafik".

À tous ma famille "Mebrouki" et la famille "Harabi".

À mon binôme "Mohamed", mon intime ami et mon frère qui n'existait pas, merci pour tous les beaux moments que nous avons vécus, dieu m'a toujours préserver pour moi.

À mes chère amis : "Seddik", "Kamel", "Youcef", "Abdelhak", "Hassan", "Abdelkader", "Hamza", "Djamel","Oussama", "Rachid", et "Amine".

. Ainsi que tout mes collègues de la promo RO

À tous ceux qui, par un mot, m'ont donné la force de continuer...

Mohamed Amine

Dédicaces

À la plus belle créature qu'Allah a créée sur terre..

Je dédie ce travail :

À mes très chers parents : Mon Père "abdeldjabar", Ma Mère "rokia"qui m'ont guidé durant les moments les plus pénibles de ce long chemin, mes chers parents qui ont sacrifié toute leur vie afin de me voir devenir ce que je suis, Maman j'adore ,Papa j'adore.

À mes soeurs : "Maria", "Maroua", "Merieme".

À mon frère : "Mouatassim billah"

À tous ma famille "Gasti" et la famille "Hamel" et "Fortas"

À mon binôme "Mohamed Amine", mon intime ami et mon frère qui n'existait pas, merci pour tous les beaux moments que nous avons vécus, dieu m'a toujours préserver pour moi.

À mes chère amis : "Ayoub", "Abdelraouf", "abdelnour", "Oussama", "Elhadj", "Moustapha", "Toufik", "Abdelkader", "Hamza", "Rachid" et "Amine"

. Ainsi que tout mes collègues de la promo RO

À tous ceux qui, par un mot, m'ont donné la force de continuer...

Mohamed

Résumé

Le problème d'ordonnancement, qui consiste à ordonnancer des travaux sur une ensemble de machines, a été très étudié dans la littérature et beaucoup de méthodes performants ont été proposées pour sa résolution.

Nous considérons ici le cas ou les machines peuvent avoir à subir un certain temps de préparation entre deux opérations consécutives.

Dans ce travail, nous nous intéressons à étudier quelques problèmes avec temps de préparation et cherchons à les résoudre de manière exacte ou approchée.

Mots clés : Ordonnancement, temps de préparation, complexité, heuristique.

Abstract

The scheduling problem, which consists in scheduling jobs on a set of machines, has been extensively studied in the literature and many efficient methods have been proposed for its resolution.

We consider here the case where machines may have to a certain setup time between two consecutive operations.

In this work, we are interested to study some problems with setup time and try to solve them in an exact or approximate way.

Mots clés :Scheduling, setup time, complexity, heuristics.

ملخص

تمت دراسة مشكلة الجدولة ، والتي تتمثل في جدولة الوظائف على مجموعة من الآلات ، على نطاق واسع في الأدبيات وتم اقتراح العديد من الطرق الفعالة لحلها.
نعتبر هنا الحالة التي قد تضطر فيها الآلات إلى الخضوع لوقت إعداد معين بين عمليتين متتاليتين.
في هذا العمل نهتم بدراسة بعض المشاكل المتعلقة بوقت التحضير ونسعى لحلها بشكل دقيق أو تقريبي.

الكلمات المفتاحية: الجدولة، وقت التحضير، التعقيد ، ارشادية

Table des matières

Résumé **1**

Abstract **2**

Introduction Générale **1**

1 Notions de base **3**

1.1 Introduction 3

1.2 Présentation d'un problème d'ordonnancement 3

1.2.1 Définition de l'ordonnancement 3

1.2.2 Définition d'un problème d'ordonnancement 3

1.3 Éléments d'un problème d'ordonnancement 4

1.3.1 Les tâches 4

1.3.2 Les ressources 5

1.3.3 Les contraintes 5

1.3.4 Les objectifs 6

1.4 Ordonnancement avec temps de préparation 7

1.5 Caractérisation des machines d'un atelier 8

1.5.1 Les machines parallèles (parallel machines) 8

1.5.2 Les machines spécialisées 8

1.6 Classification d'un problème d'ordonnancement 9

1.6.1 Champ α : Ressources 9

1.6.2 Champ β : Contraintes 10

1.6.3 Champ γ : Critère d'optimisation 11

1.7 Représentation des problèmes d'ordonnancement par un Diagramme de Gantt 11

1.8 Analyse de complexité 11

1.8.1 Les principes fondamentaux de complexité 11
1.8.2 Les classes de complexité 12
1.8.3 Réduction entre problèmes d'ordonnancement 13
1.9 Conclusion 14

2 Quelques problèmes avec temps de préparation **15**
2.1 Introduction 15
2.2 Le problème $F2/S_{ij}/C_{max}$ 15
2.2.1 Description du problème 15
2.2.2 Complexité du problème 15
2.2.3 Résolution de problème 15
2.3 Le problème $1/Set-up/C_{max}$ 18
2.3.1 Description du problème 18
2.3.2 Complexité du problème 18
2.3.3 Exemple numérique 18
2.4 Le problème $P2/P_i = P; S_i/C_{max}$ 20
2.4.1 Description du problème 20
2.4.2 Complexité du problème 20
2.5 Le problème $P2/S_i/C_{max}$ 21
2.5.1 Description du problème 21
2.5.2 Complexité du problème 21
2.6 Conclusion 22

3 Heuristiques proposées pour résoudre le problème $P2/P_i = P; S_i/C_{max}$ **23**
3.1 Introduction 23
3.2 Heuristiques pour résoudre le problème $P2/P_i = P; S_i/C_{max}$ 23
3.2.1 Représentation du problème 23
3.2.2 Les heuristiques proposées 24
3.3 Exemple d'application 28
3.4 Comparaison entre les heuristiques proposées 33
3.5 Conclusion 35

Conclusion Générale **36**

BIBLIOGRAPHIE **37**

Table des figures

1.1 Différents modèles de temps de préparation(S_i étant le temps de montage et R_i le temps de démontage) 7
1.2 Exemple d'un diagramme de Gantt 11
1.3 Schéma représentative de classe de complexité 13
1.4 Hiérarchie de complexité entre différents problèmes 14

2.1 Solution optimale de l'exemple 1 17
2.2 Graphe complet de l'exemple numérique 19
2.3 Solution optimal de l'exemple numérique 20

3.1 Solution générale de l'exemple par l'heuristique H1 29
3.2 Solution générale de l'exemple par l'heuristique H2 31
3.3 Solution générale de l'exemple par l'heuristique H3 33

Liste des algorithmes

1 Algorithme de Johnson 16
2 Algorithme de H1 25
3 Algorithme de H2 26
4 Algorithme de H3 27

Liste des tableaux

2.1 Les données de l'exemple 1 16
2.2 Les temps de traitement modifiés 17
2.3 Les temps de traitement de l'exemple numérique 19
2.4 Les temps de transition de l'exemple numérique 19

3.1 Les données de l'exemple d'application 28

Notations et Abréviations

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

Introduction Générale

La recherche opérationnelle, appelée aussi science du management ou science de la décision, est une discipline relativement récente, elle apparat principalement dans les années 1940, motivée par des applications militaires. elle s'inscrit aujourd'hui pleinement dans le cadre de la compétitivité économique. La recherche opérationnelle a pour but d'apporter une démarche scientifique à la résolution de problèmes de décision complexe issus du monde réel. Sa vocation scientifique est alors de construire des modèles formels d'aide à la décision, en particulier les modèles liés à des problèmes d'optimisation et de proposer des méthodes de résolution efficace de ces modèles. l'efficacité de ces méthodes nécessite une connaissance approfondie des caractéristiques du problème réel, leur évaluation se mesure à la fois sur leur efficacité (complexité, taille des problèmes résolus,...) et sur l'implémentation pratique des solutions obtenues[1].

La théorie d'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcule de dates d'exécution optimales de tâche pour cela, il est très souvent nécessaire d'affecter en même temps, les ressources nécessaires à l'exécution de ces tâches, un problème d'ordonnancement peut être considéré comme un sous-problème de planification dans lequel il s'agit de décider de l'exécution opérationnelle des tâches planifiées.

Les problèmes d'ordonnancement d'ateliers consistent surement pour les entreprises une des difficultés importantes de leurs systèmes de gestion et de pilotage de la production, en effet, c'est ce niveaux que doivent être prises en compte les caractéristiques réelles multiples et complexes des ateliers. Dans ce type d'ordonnancement les ressources sont généralement des machines et chaque travail à ordonnancement concerne un produit ou un lot de produit à fabriquer en respectant les gammes de fabrication.
Dans notre travail, nous nous intéressons à l'ordonnancement avec temps de préparation. Pour cela nous divisons ce travail en trois chapitres, ils sont organisé comme suit :

Dans le premier chapitre nous traitons les notions de base de l'ordonnancement (présentation, formulation et classification d'un problème d'ordonnancement), la complexité et l'ordonnancement avec temps de préparation.
Le deuxième chapitre est consacré à étudier quelques problèmes avec temps de préparation à savoir un problème facile avec sa résolution et des problèmes difficiles avec la preuve de leurs difficulté.
Nous identifions le dernier chapitre pour la représentation du problème $P2/P_i = P; S_i/C_{max}$ qui est NP-difficile (chapitre 2) et nous proposons trois heuristiques pour le résoudre avec un exemple d'application, enfin nous faisons une comparaison entre ces heuristiques.

Nous terminons notre travail par une conclusion générale.

Chapitre 1

Notions de base

1.1 Introduction

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches, de façon plus précise, un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchainement, etc) et de contraintes portant sur la disponibilité des ressources requises. Un ordonnancement est défini par le planning d'exécution des tâches et l'allocation des ressources et vise à satisfaire un ou plusieurs objectifs.

1.2 Présentation d'un problème d'ordonnancement

1.2.1 Définition de l'ordonnancement

Ordonnancer c'est programmer l'exécution d'une réalisation en attribuant les ressources aux tâches et en fixant leur dates d'exécution. La réalisation peut être, par exemple, un programme informatique ou le carnet de commandes d'un atelier alors les tâches sont des processeurs informatiques ou des traitements à appliquer à certaines pièces par des machines. Les ressources sont alors des processeurs de la mémoire ou des machines et du personnel. Etablir un ordonnancement revient à coordonner l'exécution de toutes les tâches, en utilisant au mieux les ressources disponibles[14].

1.2.2 Définition d'un problème d'ordonnancement

Le problème d'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu des contraintes temporelles (délais, contraintes d'enchaî-

nements...) et des contraintes portant sur l'utilisation et la disponibilité des ressources requises[6].

1.3 Éléments 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[10].
Une tâche notée T_i est définie par ses caractéristiques temporelles :
p_i : le temps de traitement de la tâche T_i.
S_i : le temps de préparation de la tâche T_i.
r_i : la date de disponibilité de la tâche T_i.
d_i : la date échue de la tâche T_i.
C_i : la date de fin de traitement de la tâche T_i.
W_i : un poids associé à la tâche T_i.
l_i= C_i-d_i : tardiveté de la tâche T_i(retard algébrique).
t_i= max {l_i,0} : retard de la tâche T_i.
U_i :indicateur de retard de la tâche $T_i = \begin{cases} 1 & \text{si} C_i > d_i \\ 0 & \text{sinon} \end{cases}$
f_i= C_i-r_i : une durée de flow (flow time)
Dans le reste de ce cours, nous avons : L'ensemble des n tâches est noté T = {T_1, T_2,..., T_n} L'ensemble des m machines est noté M = {M_1,M_2,...,M_n} 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 :

Les tâches morcelables (préemptibles) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.

Les 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[6] :

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

Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distinguées : contraintes de ressources et contraintes temporelles[3].

Les contraintes de ressources

Plusieurs types de contraintes peuvent être induites par la nature des ressources. A titre d'exemple, la capacité limitée d'une ressource implique un certain nombre, à ne pas dépasser, de tâches à exécuter sur cette ressource. Les contraintes relatives aux ressources peuvent être disjonctives, induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource, ou cumulatives impliquant la limitation du nombre de tâches à réaliser en parallèle.

Les contraintes temporelles

Elles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. 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 la tâche T_j, des contraintes de dates au plus tôt, liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

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 :$N_T = \sum_{i=1}^{n} u_i$

1.4 Ordonnancement avec temps de préparation

Les temps de préparation des ressources incluent les tâches nécessaires à effectuer sur les machines ou les processus de production avant de commencer la production proprement dite. Ces travaux de préparation peuvent être de différentes natures. A titre d'exemple, on peut citer le débranchement, le démontage, le nettoyage, etc. Le temps de préparation est un temps immobilisant la ressource préalablement ou suite à l'exécution d'une opération. On distingue : le temps de montage indépendant de la séquence (non sequence dépendent setup times). le temps de démontage indépendant de la séquence (non sequence dépendent removal times). le temps de préparation dépendant de la séquence (sequence dépendent setup times). Certaines tâches sont parfois regroupées par caractéristiques communes de préparation. Le temps de préparation par lot ou par famille (batch ou family setup time) est un cas fréquemment rencontré. Chaque tâche est supposée appartenir à une famille ou batch. Aucun temps de préparation n'est nécessaire entre deux tâches de la même famille (ou batch). Ce temps dépendant soit uniquement de la famille de tâches (batch) qui le suit, soit celle qui le précède, soit dépendant de la séquence des familles (batch). Le temps de préparation majeur (noté maj) est le temps de préparation nécessaire entre deux opérations de familles différentes alors qu'un temps de préparation mineur (noté min) est un temps nécessaire entre deux opérations de la même famille[11].

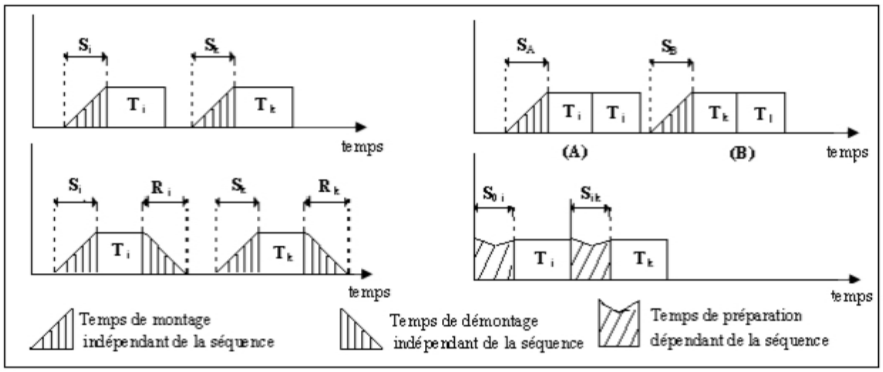


FIGURE 1.1: Différents modèles de temps de préparation(S_i étant le temps de montage et R_i le temps de démontage)

1.5 Caractérisation des machines d'un atelier

Il existe en général 2 types de machines[10] :

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

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

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.

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.

Jop 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.

1.6 Classification d'un problème d'ordonnancement

Pour des raisons de clarté et de compréhension, il est important d'adopter un descripteur qui soit basé sur l'existant et homogénéisant les notations actuelles.

La notion $\alpha/\beta/\gamma$ introduite par Graham et al. est couramment utilisée pour distinguer les différents problèmes d'ordonnancement entre eux et les classifier[7].

1.6.1 Champ α : Ressources

Renseigne les informations relatives aux ressources, à savoir le nombre de machine, type de machine et les types d'ateliers. Il est composé de deux sous champs et désigné par α_1, α_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 tous 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 tous les tâches T_i d'une même machine M_j.

R : Machine parallèle générale : la vitesse d'exécution est différente pour chaque machines 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.

$\beta_3 \in \{\emptyset, r_i\}$ décrit les dates de disponibilité des différentes tâches.

$\beta_3 = \emptyset$: Toutes les dates de disponibilité sont nulles.

$\beta_3 = r_i$: Il existe des dates de disponibilité pour les tâches non identique.

$\beta_4 \in \{\emptyset, p_i = p, \underline{P} \leqslant p_i \leqslant \overline{P}\}$ 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.

$\beta_4 = \underline{P} \leqslant p_i \leqslant \overline{P}$: Les temps de traitement de chaque tâches est compris entre $\underline{P}$ et $\overline{P}$.

$\beta_5 \in \{\emptyset, d_i, \widetilde{d_i}\}$définit 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.

$\beta_5 = \widetilde{d_i}$: Il existe une date limite pour chaque tâche.

β_6 $\{\emptyset, S_i\}$ 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 Champ γ : Critère d'optimisation

Le troisième champ γ décrit le critère d'optimalité. On cherche à minimiser Y avec $Y \in \{C_{max}, \overline{C}, \overline{C_w},\ T_{max}, L_{max}...etc\}$.

1.7 Représentation des problèmes d'ordonnancement par un Diagramme de Gantt

Ce type de diagramme a été mis au point par un américain Henry Gantt. Ceux-ci indiquant, selon une échelle temporelle, l'occupation des machines par les différentes tâches, les temps mort, et les éventuelles indisponibilités des machines dues aux changement entre produits[4].

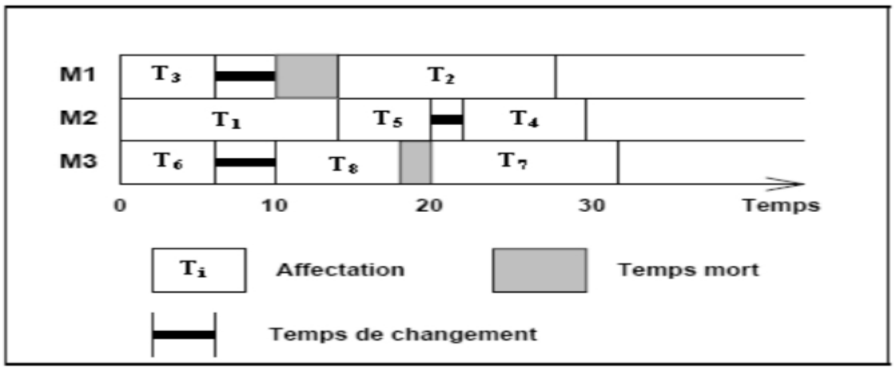


FIGURE 1.2: Exemple d'un diagramme de Gantt

1.8 Analyse de complexité

1.8.1 Les principes fondamentaux de complexité

La complexité

La complexité d'un algorithme est la durée d'exécution de cet algorithme en fonction de la taille des données. Un algorithme est dit efficace ou encore polynomial lorsque le nombre total d'opérations élémentaires (opérations mathématiques, comparaison) effectuées pour aboutir à la solution est borné par un polynôme en la taille du problème en entrée. Donc un problème est dit polynomial s'il existe un algorithme polynomial pour le résoudre. Il est dit pseudo-polynomial si sa complexité est bornée par un polynôme en fonction de la taille de la plus grande instance du problème[10].

1.8.1.1 Problème d'optimisation combinatoire

Un problème d'optimisation combinatoire consiste à déterminer un meilleur élément parmi un nombre fini d'élément mais souvent très élevé .pour ce type de problème, il est nécessaire de mesurer l'efficacité de tout algorithme mis en ouvre .Ainsi, pour comparer la performance de plusieurs algorithmes résolvant un même problème, il existe une mesure basée sur leur temps d'exécution, c'est la complexité en temps[2].

1.8.1.2 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).

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[12].

1.8.2 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[12].

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.

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[13].

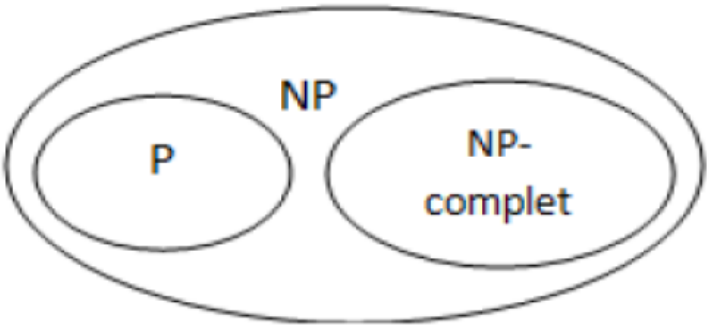


FIGURE 1.3: Schéma représentatif de classe de complexité

1.8.3 Réduction entre problèmes 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[5].

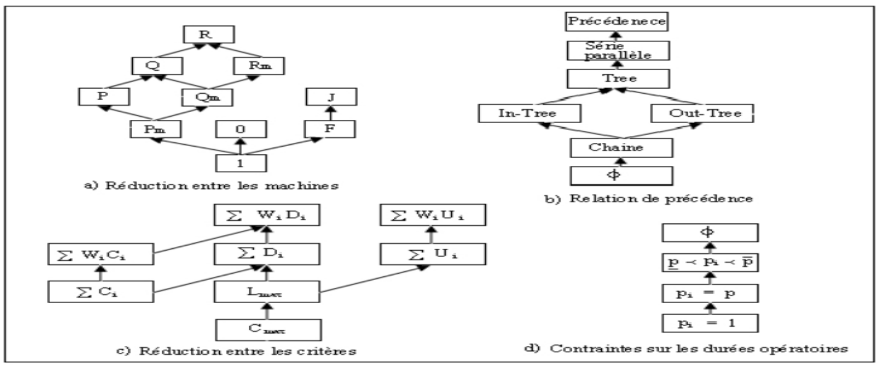


FIGURE 1.4: Hiérarchie de complexité entre différents problèmes

1.9 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 et le problème d'ordonnancement avec temps de préparation , 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

Quelques problèmes avec temps de préparation

2.1 Introduction

Nous concentrons dans ce chapitre sur quelques problèmes avec temps de préparation. Dans un premier lieu nous étudions ces différents propriété et restrictions. Ensuite nous étudions quelques problèmes difficiles avec temps de préparation.

2.2 Le problème $F2/S_{ij}/C_{max}$

2.2.1 Description du problème

Traiter un ensemble de tâches indépendantes et sans préemption, ayant des temps de traitement et des temps de préparation arbitraires sur deux machines spécialisées dans un système flow shop. l'objectif est la minimisation de makspan.

2.2.2 Complexité du problème

Théorème 2.1. *Le problème $F2/S_{ij}/C_{max}$ est O(n log n)[15].*

2.2.3 Résolution de problème

Nous cherchons à minimiser n tâches à travers 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 modification des temps de traitement ensuite l'utilisation de l'algorithme de Johnson.
La modification se fait de la manière suivante : $p'_{i1} = s_{i1} + p_{i1} - s_{i2}$;

$p'_{i2} = p_{i2}$

où

s_{ij} :temps de préparation de la tâche T_i pour la j^{me} machine (j=1,2).

p_{ij} :temps de traitement de la tâche T_i sur la j^{me} machine (j=1,2).

Algorithme de Johnson

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[9].

Algorithm 1 Algorithme de Johnson

Début

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 ;

Établir l'ensemble E2 des fabrications dont le temps opératoire sur la première machine supérieur au temps opératoire sur la deuxième machine ;

Trier l'ensemble E1 dans l'ordre croissant des temps opératoires sur la première machine ;

Trier l'ensemble E2 dans l'ordre décroissant des temps opératoires sur la deuxième machine ;

L'ordre des fabrications est déterminé par l'ordre de E1 trié puis de E2 trié ;

Fin

Exemple 1 : Etant donné le tableau suivant

T_i	T_1	T_2	T_3	T_4	T_5	T_6
p_{i1}	3	4	2	5	1	6
p_{i2}	5	1	2	3	5	6
s_{i1}	1	2	1	3	4	5
s_{i2}	2	3	1	2	4	1

TABLE 2.1: Les données de l'exemple 1

Nous calculons les temps de traitement modifiés :

T_i	T_1	T_2	T_3	T_4	T_5	T_6
p'_{i1}	2	3	2	6	1	10
p'_{i2}	5	1	2	3	5	6

TABLE 2.2: Les temps de traitement modifiés

Nous affectons les différentes tâches à la liste E1 et E2 tout en respectant l'algorithme de Johnson.
Nous aurons :

— La liste E1 va contenir les tâches : T_5,T_3 ,T_1,T_6.

— La liste E2 va convertir les tâches : T_2, T_4

L'ordre final sera alors T_5, T_3, T_1, T_6, T_4, T_2

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

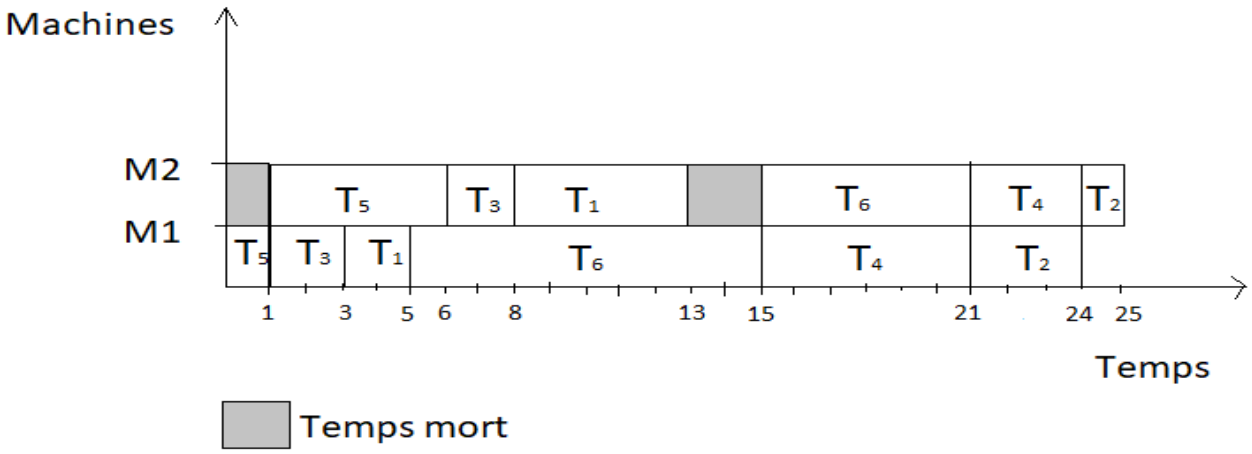


FIGURE 2.1: Solution optimale de l'exemple 1

2.3 Le problème $1/Set-up/C_{max}$

2.3.1 Description du problème

Traiter un ensemble de tâches indépendantes et sans préemption, ayant des temps de traitement et des temps de préparation arbitraires sur une seule machine. l'objectif est la minimisation de makspan.

2.3.2 Complexité du problème

Théorème 2.2. *Le problème* $1/Set-up/C_{max}$ *est NP-difficile (ce résultat est tirée de[10]).*

Preuve Considérons le problème suivant :un voyageur de commerce doit visiter n villes et doit passer une seule fois par chacune d'elles. Généralement,nous commençons par choisir n'importe quelle ville et nous terminons par un retour à cette ville . nous devons minimiser la distance totale parcourue sachant que les distances entre les villes sont connues. La notion de distance peut être remplacée par le temps ou le coût. Le problème de voyageur de commerce est un problème NP-difficile. cela signifie que le temps nécessaire pour trouver une solution optimale augmente de façon exponentielle en fonction des données du problème[8].
Supposons qu'il existe un temps de transition (set-up time) C_{ik}=$C(T_i, T_k)$ entre deux tâches T_i et T_k correspondant au temps de réglage d'ajustement au de changement d'outil.
Le problème $1/Set-up/C_{max}$ est équivalent au problème du voyageur de commerce dans un graphe complet G=(V,E) où V est l'ensemble des tâches et où chaque arête reliant deux tâches T_i et T_k est valée C_{ik}. en effet, il suffit de trouver $\min(C_{1\pi(1)}+C_{1\pi(2)}+...+C_{\pi(n-1)\pi(n)})$ où π est une permutation de nombre 1,2,...,n .
Ce problème est donc NP-difficile au sens fort.

2.3.3 Exemple numérique

Soit à ordonnancer quatre tâches sur une seule machine, les données sont dans les tableaux suivants :

T_i	T_1	T_2	T_3	T_4
p_i	3	2	1	2

TABLE 2.3: Les temps de traitement de l'exemple numérique

C_{ij}	T_1	T_2	T_3	T_4
T_1	0	3	2	2
T_2	3	0	4	1
T_3	2	4	0	4
T_4	2	1	4	0

TABLE 2.4: Les temps de transition de l'exemple numérique

Nous construisons le graphe complet suivant :

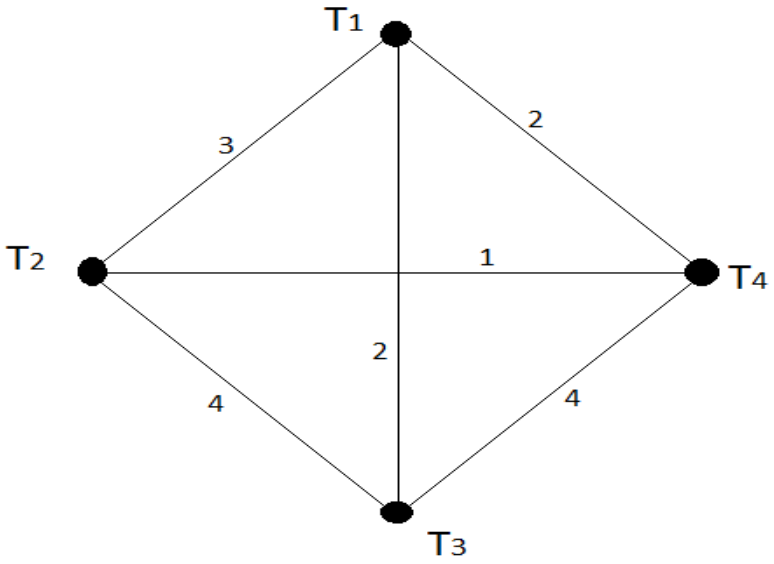


FIGURE 2.2: Graphe complet de l'exemple numérique

La tournée optimale de coût minimal est :(T_1,T_3,T_2,T_4,T_1)

Nous supprimons l'arrête qui a le plus grand coût donc l'arrête (T_2,T_3)

Nous trouvons deux solutions optimales :(T_2,T_4,T_1,T_3) et(T_3,T_1,T_4,T_2) avec $C_{max} = 13$.

Le diagramme de Gantt de la première solution est :

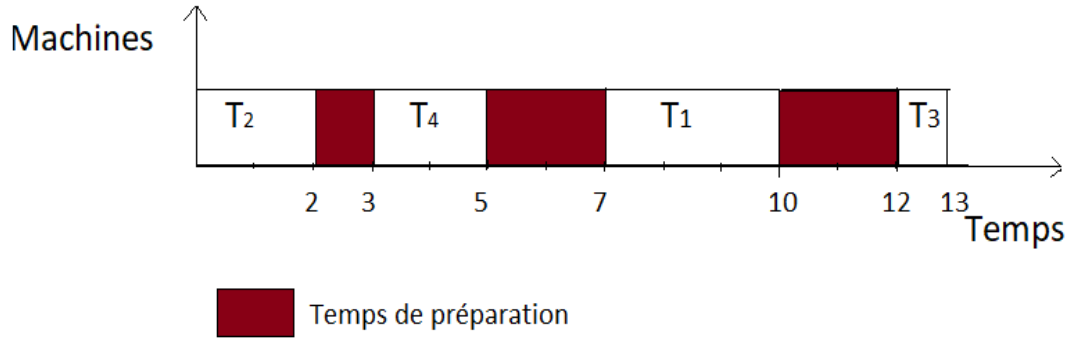


FIGURE 2.3: Solution optimal de l'exemple numérique

2.4 Le problème $P2/P_i = P; S_i/C_{max}$

2.4.1 Description du problème

Il s'agit d'ordonnancer un ensemble de n tâches $T_1, T_2, ..., T_n$ sur 2 machines parallèles identiques M1,M2 pour optimiser le makespan qui correspond à la date de fin de traitement de l'ensemble des tâches. Chaque tâche nécessite avant son traitement une préparation ayant une durée non nulle.

2.4.2 Complexité du problème

Théorème 2.3. *Le problème $P2/P_i = P; S_i/C_{max}$ est NP-difficile[11] .*

Preuve Considérons le problème de décision NP-complet suivant connu sous le nom de 2-partition : " étant donnés n entiers positifs a1,a2, an. Existe-t-il un sous-ensemble A1 $\subseteq A(A = a_1, a_2, ..., an) telque : \sum_{a_i \in A1} a_i = \sum_{a_i \in A \setminus A1} a_i$?
Montrons que ce problème se réduit polynomialement au problème de décision suivant : étant données n tâches T1, T2, ..., Tn indépendantes et non morcelables avec des temps de traitement $p_i = p = 0$ le temps nécessaire pour la préparation d'une tâche est $S_i = ai$, et $Y = \frac{1}{2}\sum_{i=1}^{n} a_i$. Existe-t-il un ordonnancement de ces n tâches sur deux machines de durée inférieure ou égale à Y ?
Ce problème appartient à la classe NP, car on peut vérifier en un temps polynomial qu'une affectation des tâches aux machines vérifie toutes les contraintes.
Il est clair que la réduction précédente est polynomiale (O(n)). Nous prouvons que le

problème 2-partition a une solution si et seulement si le problème d'ordonnancement a une solution. Si le problème 2-partition a une solution, alors, il existe un sous-ensemble $A1 \subseteq A$ avec la propriété désirée : $\sum_{a_i \in A1} a_i = \sum_{a_i \in A \setminus A1} a_i$. Nous construisons une solution pour le problème d'ordonnancement comme suit : nous préparons les tâches qui appartiennent à l'ensemble A1 sur la machine M1 et les tâches restantes, nous les préparons sur la machine M2. Donc la durée de l'ordonnancement est égale à Y = $1/2 \sum_{i=1}^{n} a_i$.
Si le problème d'ordonnancement a une solution de durée inférieure ou égale à Y, alors, chaque tâche n'est préparée que sur l'une des deux machines. Donc le problème 2-partition a une solution, en posant A1 comme étant l'ensemble de tâches préparées sur la machine M1 et $A \setminus A1$ l'ensemble de tâches préparées sur la machine M2 [11].

2.5 Le problème $P2/S_i/C_{max}$

2.5.1 Description du problème

Traiter un ensemble de tâches indépendantes non interruptibles ayant des temps de traitement arbitraires avec temps de préparation sur deux machines parallèles identiques, l'objectif est la minimisation de la date de fin de traitement.

2.5.2 Complexité du problème

Théorème 2.4. *Le problème $P2/S_i/C_{max}$ est NP-difficile[11] .*

Preuve Soit le problème de décision suivant de 2-partition : " étant données n entiers positifs $a_1, a_2, ..., a_n$. Existe-t-il un sous-ensemble $A1 \subseteq A$ $(A = a_1, a_2, ..., a_n)$ tel que :$\sum_{a_i \in A1} a_i = \sum_{a_i \in A \setminus A1} a_i$?" .
Montrons que ce problème se réduit polynomialement au problème à deux machines parallèles identiques, étant données n tâches $T_1, T_2, ..., T_n$ indépendantes et non morcelables avec des temps de préparation $S_i = s$ =0 et chaque tâche Ti a un temps de traitement $p_i = a_i$ et un nombre Y =$1/2 \sum_{i=1}^{n} a_i$.
Existe-t-il un ordonnancement de ces n tâches sur deux machines de durée inférieure ou égale à Y ?
Ce problème appartient à la classe NP, car on peut vérifier en un temps polynomial qu'une affectation des tâches aux machines vérifie toutes les contraintes. Il est clair que la réduction précédente est polynomiale (O(n)). Nous prouvons que le problème 2-partition

a une solution si et seulement si le problème d'ordonnancement a une solution. Supposons que le problème 2-partition a une solution, donc il existe un sous-ensemble $A1 \subseteq A$ avec $\sum_{a_i \in A1} a_i = \sum_{a_i \in A \setminus A1} a_i$. Alors, nous pouvons construire un ordonnancement pour le problème $P2/S_i = s/Cmax$, en traitant les $\mid A1 \mid$ tâches sur la machine M1 et les $\mid A \setminus A1 \mid$ tâches restantes, nous les traitons sur la machine M2. Donc, la durée de l'ordonnancement est égale à Y.

Supposons maintenant que le problème d'ordonnancement a une solution de durée inférieure ou égale à Y, alors,chaque tâches n'est affectée qu'à l'une des deux machines, tel que A1 est l'ensemble des tâches affectées à la machine M1 et $A \setminus A1$ l'ensemble des tâches affectées à la machine M2. Donc le problème 2-partition a une solution .

2.6 Conclusion

Dans ce chapitre nous avons présenté quelques problèmes d'ordonnancement avec temps de préparation comme les problèmes faciles, nous avons présenté des algorithmes pour résoudre ce type de problème ainsi qu'un exemple d'application, nous avons présenté aussi des problèmes difficiles et la preuve de leurs complexité.

Chapitre 3

Heuristiques proposées pour résoudre le problème $P2/P_i = P; S_i/C_{max}$

3.1 Introduction

Dans ce chapitre nous présentons le problème $P2/p_i = p; S_i/C_{max}$ ainsi que nous proposons quelques heuristiques pour résoudre ce problème qui est NP-Difficile .

3.2 Heuristiques pour résoudre le problème $P2/P_i = P; S_i/C_{max}$

3.2.1 Représentation du problème

Toujours avec l'objectif de la minimisation de makespan, supposons que les tâches ne sont pas tous disponibles simultanément, mais ont des temps de préparation s_i différents.
Le problème $P2/p_i = p; S_i/C_{max}$ consiste à ordonnancer un ensemble des tâches indépendantes ayant des temps de traitement identiques égaux à p, des temps de préparation aléatoires et des dates de disponibilité nulles sur deux machines parallèles identiques.
Ce problème est NP-difficile (voir chapitre 2).

3.2.1.1 Définition

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons 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[6].

3.2.2 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 $P2/P_i = P; S_i/C_{max}$ ou notre but est de trouver l'heuristique de bonne qualité.

Pour cela nous avons les notations suivantes :
C_{1l} : La date de fin de traitement de la tâche T_i en l^{me} position sur la première machine.
C_{2k} : La date de fin de traitement de la tâche T_i en k^{me} position sur la deuxième machine.
t_i : La date de début de traitement de la tâche Ti

A- Principe de la première heuristique

Etape 01

Cette heuristique consiste à ranger les tâches T_i par ordre croissant de leurs temps de préparation dans une liste L .

Etape 02

Ordonnancer la première tâche de la liste L sur la première machine libre

Algorithm 2 Algorithme de H1

Début
Ranger les tâches par ordre croissant de leurs temps de préparation dans une liste L ;
$t_1 = s_1$; $C_{11} = t_1 + p$; $l = 1$;
$t_2 = s_2$; $C_{21} = t_2 + p$; $k = 1$;
pour i=3 à n faire
$t_i = \min\{C_{2k}, C_{1l}\} + s_i$;
si $C_{2k} < C_{1l}$ **alors**
Début
$k = k + 1$;
$C_{2k} = t_i + p$;
fin
sinon
Début
$l = l + 1$;
$C_{1l} = t_i + p$;
fin ;
Fait ;
$c_{max} = max\{C_{1l}, C_{2k}\}$;
Fin.

B- Principe de la deuxième heuristique

Etape 01

Cette heuristique consiste à ranger les tâches T_i par ordre décroissant de leurs temps de préparation dans une liste L.

Etape 02

Ordonnancer la première tâche de la liste L sur la première machine libre

Algorithm 3 Algorithme de H2

Début
Ranger les tâches par ordre décroissant de leurs temps de préparation dans une liste L ;
$t_1 = s_1$; $C_{11} = t_1 + p$; $l = 1$;
$t_2 = s_2$; $C_{21} = t_2 + p$; $k = 1$;
pour i=3 à n faire
$t_i = \min\{C_{2k}, C_{1l}\} + s_i$;
si $C_{2k} < C_{1l}$ **alors**
Début
$k = k + 1$;
$C_{2k} = t_i + p$;
fin
sinon
Début
$l = l + 1$;
$C_{1l} = t_i + p$;
fin ;
Fait ;
$c_{max} = max\{C_{1l}, C_{2k}\}$;
Fin.

C- Principe de la 3^{me} heuristique

Etape 01

La première étape de cette heuristique consiste à ranger les tâches T_i par ordre croissant de leurs temps de préparation dans une liste L, ensuite déviser cette liste sur 2 listes L_1 et L_2 telle que :

L_1 : contient les tâches de la première tâche de la liste L jusqu'au milieu

L_2 : contient les tâches de la dernière tâche de la liste L jusqu'au milieu

si n est pair : diviser la liste L sur 2 liste de même cardinale c'est à dire :
$|\text{L1}| = \frac{n}{2}$
$|\text{L2}| = \frac{n}{2}$
si n est impair : affecter l'élément au milieu à la liste L_1 c'est à dire : $|\text{L1}| = \lfloor\frac{n}{2}\rfloor + 1; |\text{L2}| = \lfloor\frac{n}{2}\rfloor$
Grouper les deux listes L_1 et L_2 dans une troisième liste L_3 de la manière suivante : prendre une tâche de la liste L_1 et une tâche de la liste L_2 jusqu'à le fin des deux listes.

Etape 02

Ordonnancer les tâches de la liste L_3 sur le premier machine libre

Algorithm 4 Algorithme de H3

Début
Ranger les tâches par ordre croissant de leurs temps de préparation dans une liste L ;
Diviser la liste L sur deux listes L_1 et L_2 ;
Grouper les listes L_1 et L_2 dans une liste L_3 ;
$t_1 = s_1$; $C_{11} = t_1 + p$; $l = 1$;
$t_2 = s_2$; $C_{21} = t_2 + p$; $k = 1$;
pour i=3 à n faire
$t_i = \min\{C_{2k}, C_{1l}\} + s_i$;
si $C_{2k} < C_{1l}$ **alors**
Début
$k = k + 1$;
$C_{2k} = t_i + p$;
fin
sinon
Début
$l = l + 1$;
$C_{1l} = t_i + p$;
fin ;
Fait ;
$c_{max} = max\{C_{1l}, C_{2k}\}$;
Fin.

3.3 Exemple d'application

Dans cette session nous désirons traiter 8 tâches indépendantes sur deux machines parallèle identiques, dont les temps de traitement sont tout identique et égaux à 2 et les temps de préparation sont donnés dans le tableau suivant :

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
s_i	1	4	7	9	10	6	2	3

TABLE 3.1: Les données de l'exemple d'application

Nous appliquons cet exemple sur les trois heuristiques proposées précédemment.

A- Application de la 1^{re} heuristique

Nous appliquons notre heuristique H1 : Ranger les tâches par ordre croissant de leurs temps de préparation Nous trouvons la liste L suivante :

L=$\{T_1, T_7, T_8, T_2, T_6, T_3, T_4, T_5\}$

Itération 0 :

$t_1 = s_1 = 1 \longrightarrow C_{11} = t_1 + p = 1 + 2 = 3; l = 1;$

$t_2 = s_2 = 2 \longrightarrow C_{21} = t_2 + p = 2 + 2 = 4; k = 1;$

Itération 01 :

$t_3 = \min\{C_{21}, C_{11}\} + s_3$

$t_3 = \min\{4, 3\} + 3; l = 1 + 1 = 2; k = 1;$

$t_3 = 6 \longrightarrow c_{12} = t_3 + p = 6 + 2 = 8$

Itération 02 :

$t_4 = \min\{C_{21}, C_{12}\} + s_4$

$t_4 = \min\{4, 8\} + 4; l = 2; k = 1 + 1 = 2;$

$t_4 = 8 \longrightarrow c_{22} = t_4 + p = 8 + 2 = 10$

Itération 03 :

$t_5 = \min\{C_{22}, C_{12}\} + s_5$

$t_5 = \min\{10, 8\}+6; l = 2+1 = 3; k = 2;$

$t_5 = 14 \longrightarrow c_{13} = t_5 + p = 14 + 2 = 16$

Itération 04 :

$t_6 = \min\{C_{22}, C_{13}\}+s_6$

$t_6 = \min\{10, 16\}+7; l = 3; k = 2+1 = 3;$

$t_6 = 17 \longrightarrow c_{23} = t_6 + p = 17 + 2 = 19$

Itération 05 :

$t_7 = \min\{C_{23}, C_{13}\}+s_7$

$t_7 = \min\{19, 16\}+9; l = 3+1 = 4; k = 3;$

$t_7 = 25 \longrightarrow c_{14} = t_7 + p = 25 + 2 = 27$

Itération 06 :

$t_8 = \min\{C_{23}, C_{14}\}+s_8$

$t_8 = \min\{19, 27\}+10; l = 4; k = 3+1 = 4;$

$t_8 = 29 \longrightarrow c_{24} = t_8 + p = 29 + 2 = 31$

$C_{max} = 31.$

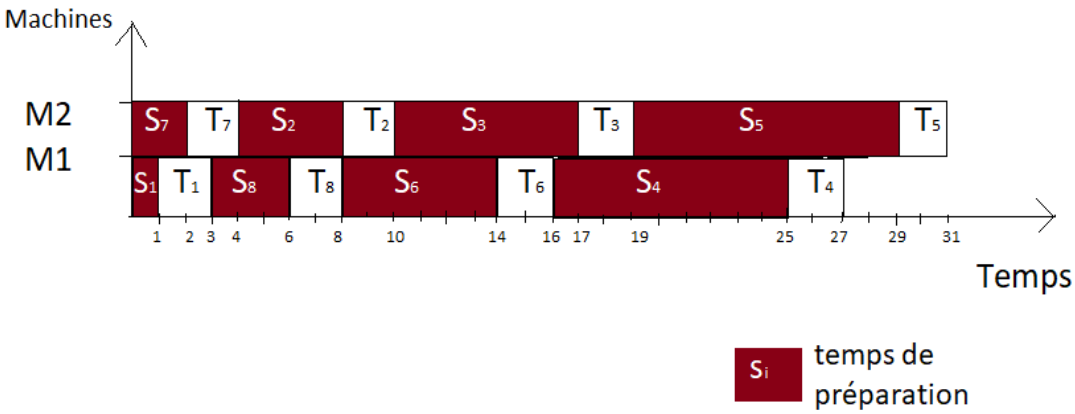


FIGURE 3.1: Solution générale de l'exemple par l'heuristique H1

B- Application de la 2^{me} heuristique

Nous appliquons notre heuristique H2 : Ranger les tâches par ordre décroissant de leurs temps de préparation Nous trouvons la liste L suivante :

L=$\{T_5, T_4, T_3, T_6, T_2, T_8, T_7, T_1\}$

Itération 0 :

$t_1 = s_1 = 10 \longrightarrow C_{11} = t_1 + p = 10 + 2 = 12; l = 1;$

$t_2 = s_2 = 9 \longrightarrow C_{21} = t_2 + p = 9 + 2 = 11; k = 1;$

Itération 01 :

$t_3 = \min\{C_{21}, C_{11}\} + s_3$

$t_3 = \min\{11, 12\} + 7; l = 1; k = 1 + 1 = 2;$

$t_3 = 18 \longrightarrow c_{22} = t_3 + p = 18 + 2 = 20$

Itération 02 :

$t_4 = \min\{C_{22}, C_{11}\} + s_4$

$t_4 = \min\{20, 12\} + 6; l = 1 + 1 = 2; k = 2;$

$t_4 = 18 \longrightarrow c_{12} = t_4 + p = 18 + 2 = 20$

Itération 03 :

$t_5 = \min\{C_{22}, C_{12}\} + s_5$

$t_5 = \min\{20, 20\} + 4; l = 2 + 1 = 3; k = 2;$

$t_5 = 24 \longrightarrow c_{13} = t_5 + p = 24 + 2 = 26$

Itération 04 :

$t_6 = \min\{C_{22}, C_{13}\} + s_6$

$t_6 = \min\{20, 26\} + 3; l = 3; k = 2 + 1 = 3;$

$t_6 = 23 \longrightarrow c_{23} = t_6 + p = 23 + 2 = 25$

Itération 05 :

$t_7 = \min\{C_{23}, C_{13}\} + s_7$

$t_7 = \min\{25, 26\} + 2; l = 3; k = 3 + 1 = 4;$

$t_7 = 27 \longrightarrow c_{24} = t_7 + p = 27 + 2 = 29$

Itération 06 :

$t_8 = \min\{C_{24}, C_{13}\} + s_8$

$t_8 = \min\{29, 26\} + 1; l = 3 + 1 = 4; k = 4;$

$t_8 = 27 \longrightarrow c_{14} = t_8 + p = 27 + 2 = 29$

$C_{max} = 29.$

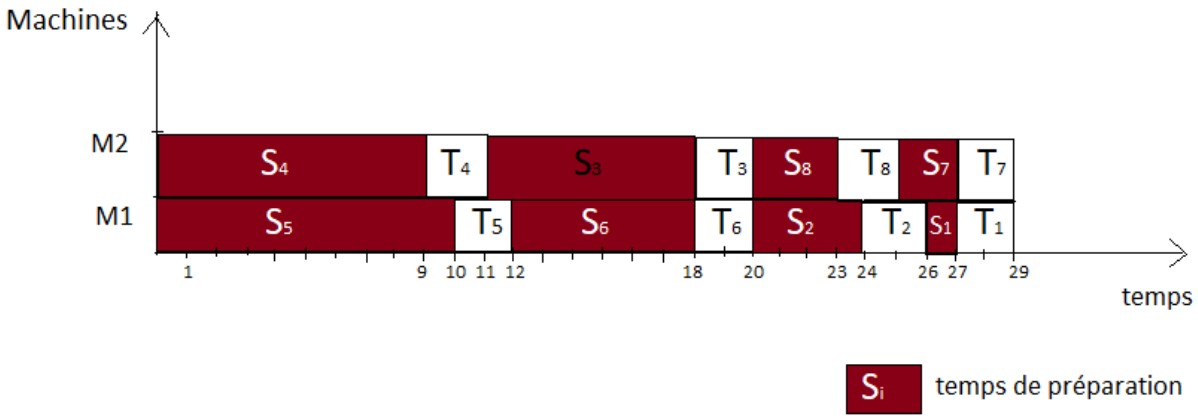


FIGURE 3.2: Solution générale de l'exemple par l'heuristique H2

C- Application de la 3^{me} heuristique

Nous appliquons notre heuristique H3 :

$L = \{T_1, T_7, T_8, T_2, T_6, T_3, T_4, T_5\}$; L_1=$\{T_1, T_7, T_8, T_2\}$;

L_2=$\{T_5, T_4, T_3, T_6\}$;

$L_3 = \{T_1, T_5, T_7, T_4, T_8, T_3, T_2, T_6\}$;

Itération 0 :

$t_1 = s_1 = 1 \longrightarrow C_{11} = t_1 + p = 1 + 2 = 3; l = 1;$

$t_2 = s_2 = 10 \longrightarrow C_{21} = t_2 + p = 10 + 2 = 12; k = 1;$

Itération 01 :

$t_3 = \min\{C_{21}, C_{11}\} + s_3$

$t_3 = \min\{12, 3\} + 2; l = 1 + 1 = 2; k = 1;$

$t_3 = 5 \longrightarrow c_{12} = t_3 + p = 5 + 2 = 7$

Itération 02 :

$t_4 = \min\{C_{21}, C_{12}\} + s_4$

$t_4 = \min\{12, 7\} + 9; l = 2 + 1 = 3; k = 1;$

$t_4 = 16 \longrightarrow c_{13} = t_4 + p = 16 + 2 = 18$

Itération 03 :

$t_5 = \min\{C_{21}, C_{12}\} + s_5$

$t_5 = \min\{12, 18\} + 3; l = 3; k = 1 + 1 = 2;$

$t_5 = 15 \longrightarrow c_{22} = t_5 + p = 15 + 2 = 17$

Itération 04 :

$t_6 = \min\{C_{22}, C_{13}\} + s_6$

$t_6 = \min\{17, 18\} + 7; l = 3; k = 2 + 1 = 3;$

$t_6 = 24 \longrightarrow c_{23} = t_6 + p = 24 + 2 = 26$

Itération 05 :

$t_7 = \min\{C_{23}, C_{13}\} + s_7$

$t_7 = \min\{26, 18\} + 4; l = 3 + 1 = 4; k = 3;$

$t_7 = 22 \longrightarrow c_{14} = t_7 + p = 22 + 2 = 24$

Itération 06 :

$t_8 = \min\{C_{23}, C_{14}\} + s_8$

$t_8 = \min\{26, 24\} + 6; l = 4 + 1 = 5; k = 3;$

$t_8 = 30 \longrightarrow c_{18} = t_8 + p = 30 + 2 = 32$

$C_{max} = 32.$

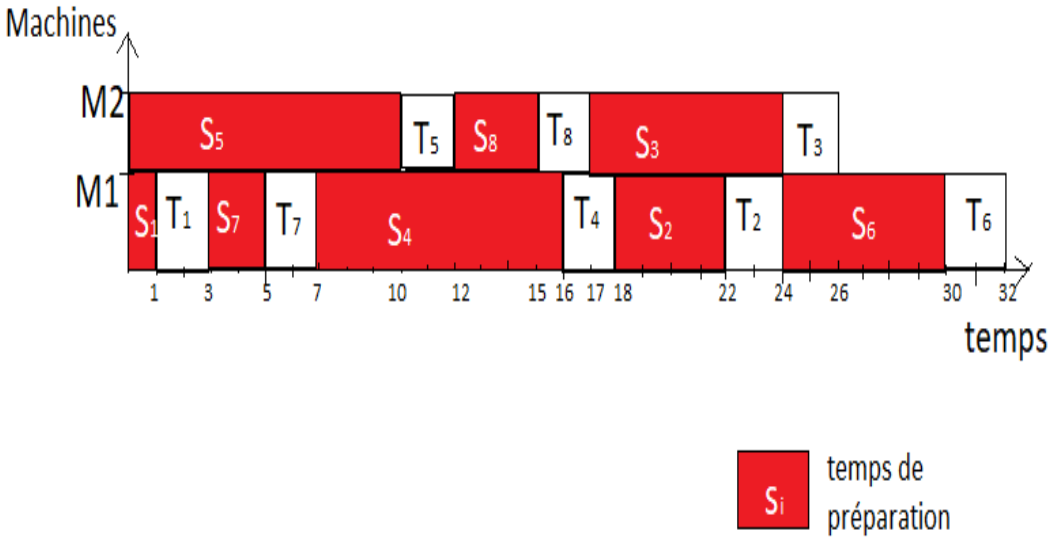


FIGURE 3.3: Solution générale de l'exemple par l'heuristique H3

3.4 Comparaison entre les heuristiques proposées

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

$$R = \frac{C_{max}(H)}{B}$$

— $C_{max}(H)$: Date de fin de traitement de toutes les tâches obtenue par l'heuristique proposée H.

— B : est la borne inférieure du problème $P2/P_i = P; S_i/C_{max}$ avec :

$$B = \frac{\sum_{i=1}^{n} p_i + \sum_{i=1}^{n} s_i}{m}$$

Ou $p_i = p$ (temps de traitement identiques pour toutes les tâches)
m=2 (nombre de machines égale à 2).
Alors nous trouvons :

$$B = \frac{np + \sum_{i=1}^{n} s_i}{2}$$

Lorsque R est proche de 1 alors l'approximation est de bonne qualité.

Calculons alors la borne inférieur de notre exemple :

$$B = \frac{np + \sum_{i=1}^{n} s_i}{2}$$

$$B = \frac{28 + 42}{2} = 29$$

Pour l'heuristique $H1$

On a $C_{max} = 31$ et donc $R = \frac{C_{max}(H_1)}{B} = \frac{31}{29} = 1.0689$

Alors cette heuristique est de bonne qualité car $R = 1.0689$ est proche de 1.

Pour l'heuristique $H2$

On a $C_{max} = 29$ et donc $R = \frac{C_{max}(H_2)}{B} = \frac{29}{29} = 1$

Alors cette heuristique est de bonne qualité car $R = 1$.

Pour l'heuristique $H3$

On a $C_{max} = 32$ et donc $R = \frac{C_{max}(H_3)}{B} = \frac{32}{29} = 1.1034$

Alors cette heuristique est de bonne qualité car $R = 1.1034$ est proche de 1.

A partir des résultats obtenues, nous remarquons que l'heuristique $H2$ donne la solution optimale car la valeur de C_{max} est égale à la borne inférieure Q, donc le ratio d'approximation R est égale 1, par contre les heuristiques $H1$ et $H3$ donnent des valeurs plus grand de C_{max}.

Alors pour cet exemple, l'heuristique $H2$ est plus performant des heuristiques $H1$ et $H3$ alors que $H1$ est meilleur que $H3$.

Remarque

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

3.5 Conclusion

Nous traitons dans ce chapitre le problème d'ordonnancement $P2/P_i = P; S_i/C_{max}$, nous avons commencé par la définition de ce problème ensuite nous avons proposées trois heuristiques pour trouver une solution approchée de ce dernier .

Conclusion Générale

Les problèmes d'ordonnancement sont présents 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.

Les problèmes d'ordonnancement avec temps de préparation est l'un des problèmes d'ordonnancement intensivement étudiés. Dans ce travail, nous avons traité ces problèmes, les types de ces derniers selon leurs complexités et d'application des heuristiques proposées pour parvenir à une solution acceptable.

La première partie de ce mémoire est une présentation générale 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, Le 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 sommes intéressé dans la deuxième partie à étudier quelques problèmes d'ordonnancement avec temps de préparation qui sont classés comme des problèmes faciles ou difficiles, nous avons commencé par un problème facile et nous avons donné un algorithme pour sa résolution, ensuite nous avons exposé quelques problèmes difficile, ainsi que nous avons utilisé le problème de 2-partition pour prouver la difficulté de ces problèmes.

Dans la troisième partie, nous avons proposé trois heuristiques pour résoudre le problème $P2/p_i = p; S_i/C_{max}$ qui est NP-difficile , tout d'abord nous avons donné sa définition, la fin de cette partie est consacré a appliquer ces trois heuristiques sur un exemple donné pour trouver une solution approchée.

Bibliographie

[1] **C. M. S.Ametys,**«*RECHERCHE OPERATIONNELLE.*», 2020.

[2] **A.BENKACI**,Note de cours «*Problèmes d'optimisation combinatoire* » Pour M2 Ro université de médéa ,2016.

[3] **M.BELKHOUS**, « *mémoire de fin d'étude du diplôme master, professionnel en mathématique à la gestion*», 2016/2017.

[4] **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.

[5] **J.BLAZEWIEZ, K.H ECKER,ESCHE, G.SCHMIDT and J.WEGLARZ**, «*Sheduling computer and manufacturing processes*»,Springer-Velag.Berlin,2001.

[6] **H.BOUKEF BEN OTHMAN** , «Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques Optimisation par algorithmes génétiques et essaims particulaires», thèse de Doctorat, UNIVERSITÉ DE TUNIS EL MANAR, 2009

[7] **P.BRUCKER**, «*Scheduling algorithms*» , Fifth edition spring, 2006.

[8] **A.Gharib,and J. Benhra**«*Tuning manuel de l'algorithme d'optimisation par essaims de particules appliqué au problème de voyageur de commerce*», Xème Conférence Internationale : Conception et Production Intégrées, 2015.

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

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

[11] **W.LABBI.** «*Ordonnancent sur machines prallèles identique avec temps de préparation* mémoire de Magister en mathématique spécialité recherche opérationnelle U.S.T.H.B, 2009.

[12] **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.

[13] **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.

[14] **C.SOLON**, «*Réduction de problème combinatoire et optimisation par colonie de fournis*»,2006.

[15] **T.Yoshida and K. Hitom**, «*Optimal two-stage production scheduling with setup times separated, AIIE Trans 11 (1979*»,261-263.