Ministère de l'enseignement supérieur et de la recherche scientifique

Université Yahia Farès de Médéa

Faculté des Sciences

Département de Mathématiques et de l'Informatique



MÉMOIRE DE MASTER

en mathématiques

Option : Recherche Opérationnelle

Thème

Heuristique pour la minimisation de la variance des dates de fin d'exécution des tâches dans les problèmes d'ordonnancement à une seule machine

présenté

par

MOUSSOUNI Nadjet et OUZZAR Ibtissam

soutenu publiquement, devant le Jury composé de :

N. IKHLEF ESCHOUF	Professeur	U.YF.M	Président
S. KOUIDER AIAD	M.A.A	U.YF.M	Examinatrice
A. BENKACI	M.A.A	U.YF.M	Promoteur

22 – juin – 2022

Dédicace

Je dédie ce modeste travail,

À mes chers parents ma mère et mon père, pour leur patience, leur amour, leur soutien et leur encouragements.

À mes frères et mes sœurs qui m'avez toujours soutenu et encouragé durant ces années d'études.

À toute ma famille pour l'amour et le respect qu'ils m'ont toujours accordé.

À mon amie intime Ibtissam OUZZAR avec qui j'ai partagé de beaux moments. Que DIEU le tout puissant la protège inchallah.

À toutes mes amies qui m'ont toujours encouragé, et à qui je les souhaite plus de succès dans leur vie.

À toute personne qui m'a aidé à franchir un horizon dans la vie…

Nadjat MOUSSOUNI

Dédicace

Je dédie ce modeste travail de fin d'étude,

À ma chère mère, qui a œuvré pour ma réussite de par son amour, son soutien tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

À mon cher père, qui peut être fier et trouver ici le résultat de longues années de sacrifices pour m'aider à avancer dans la vie. Puisse dieu faire en sorte que ce travail porte son fruit : merci pour les valeurs nobles, l'éducation et le soutient permanent venu et de vous.

À mes beaux frères et mes chères sœurs : Hakim et son épouse Fadila, Soufiane, Salima et son époux Mohamed, Souad et son époux Krimo et ses enfants dans tous mes moments d'examens par leur soutien et pour leurs encouragements.

À Nadjat MOUSSOUNI, mon intime amie avec qui j'ai partagé de beaux moments. Que DIEU le tout puissant, la protège inchallah.

À mon encadrant M. BENKACI qui m'a aidé pour réaliser ce travail, dont je lui doit un grand remerciement, grand respect et une parfaite appréciation.

Enfin à tous mes chères amies et à tous ceux qui m'ont aidé de près ou de loin.

Ibtissam OUZZAR

Résumé

Dans ce mémoire, nous nous intéressons au problème de minimisation de la variance des dates de fin d'exécution des tâches dans les problèmes d'ordonnancement à une seule machine (CTV). Ce problème est classé NP-difficile (Kubiak, 1993), et il se formule comme un programme quadratique en nombres binaires (0/1). Pour le résoudre, une heuristique est proposée en élaborant un code de calcul en langage Matlab, permettant de calculer une solution approchée en temps raisonnable. Nous comparons notre heuristique avec l'heuristique EC1.2 qui est due à Eilon et Chowdhury en 1977. Nous reportons aussi des résultats numériques qui sont obtenus en testant notre code de calcul sur des exemples types du problème tirés de la littérature. Ces résultats confirment l'efficacité de notre approche de résolution.

Mots–clés : ordonnancement; minimisation de la variance; heuristique.

Abstract

In this report, we are interested in the problem of minimizing the completion time variance (CTV) in single machine scheduling problems. This problem is NP-hard (Kubiak, 1993), and it is formulated as a quadratic program in binary numbers (0/1). To solve it, a heuristic is proposed by developing a calculation code in Matlab language, allowing to compute an approximate solution in a reasonable time. We compare our heuristic with the EC1.2 heuristic which is due to Eilon and Chowdhury in 1977. We report numerical results which are obtained by testing our computer code on standard examples from the literature. These results confirm the effectiveness of our resolution approach.

Keywords : scheduling; variance minimization; heuristic.

Remerciements

Tout d'abord, nous tenons à remercier, ALLAH le Tout Puissant, pour nous avoir données le courage et la force pour accomplir ce travail de recherche.

Nous tenons à exprimer toute notre gratitude à notre promoteur, Monsieur A. BENKACI, enseignant à la faculté des sciences de l'université de Médéa, pour nous avoir encadrées et guidées tout au long de ce mémoire. Il nous a permis de réaliser ce travail en nous manifestant sa confiance, et en nous encourageant dans notre travail. Il a été toujours à notre disposition.

Nous sommes très honorées par la présence au jury notre enseignant, Monsieur N. IKHLEF-ESCHOUF, Professeur à l'université de Médéa. Nous le remercions très sincèrement, d'avoir accepté d'être président du jury.

Nos vifs remerciements vont à notre enseignante, Madame S. KOUIDER-AIAD, enseignante à l'université de Médéa, d'avoir acceptée d'examiner notre travail, et aussi pour nous avoir fait partager ses connaissances durant la formation de Master.

Enfin, nous remercions toute personne qui nous a aidé par sont soutient et sa confiance afin de finaliser ce travail.

Tables des matières

Résumé i

Abstract ii

Remerciements iii

Tables des matières iv

Liste des figures vi

Liste des tableaux vii

Liste des algorithmes viii

Introduction 9

1 Notions de base sur l'ordonnancement d'atelier 11

1.1 Introduction....11

1.2 Généralités et définitions....11

1.3 Description d'un problème d'ordonnancement....14

1.3.1 Notations....14

1.3.2 Représentation d'une tâche....14

1.3.3 Critères d'optimisation....15

1.3.4 Classification des problèmes d'ordonnancement....16

1.3.5 Représentation des ordonnancements....16

1.4 Modèle mathématique....16

1.5 Conclusion....17

2 Méthodes de résolution d'un problème d'ordonnancement 18

2.1 Introduction....18

2.2 Complexité des problèmes....18

2.3 Méthodes de résolution 20
2.3.1 Méthodes exactes 21
2.3.2 Méthodes approchées 22
2.4 Conclusion 23
3 Minimisation de la variance des dates de fin d'exécution (CTV) 24
3.1 Introduction 24
3.2 Problème de CTV 25
3.2.1 Notations 25
3.2.2 Hypothèses 25
3.2.3 Critère de performance 25
3.3 Importance pratique 26
3.4 Propriétés de dominance 27
3.4.1 Conditions nécessaires d'optimalité 27
3.4.2 Réduction de l'espace de recherche 28
3.4.3 Séquence duale et propriétés 28
3.4.4 Exemple Illustratif 29
3.5 Modèle mathématique 30
3.6 Conclusion 31
4 Heuristique pour la minimisation de la variance 32
4.1 Introduction 32
4.2 Approche de résolution 32
4.2.1 L'espace V-Shape 33
4.2.2 Les heuristiques EC1.1 et EC1.2 33
4.2.3 L'espace V-Shape symétrique et l'heuristique H 35
4.2.4 Exemple illustratif 37
4.3 Comparaison des résultats numériques 38
4.4 Conclusion 42
Conclusion 43
Notations et abréviations 45
Références 46

Liste des figures

Figure 1.1 – Caractéristiques de la tâche j. 15

Figure 2.1 – Classification des problèmes combinatoires. 19

Figure 2.2 – Schéma récapitulatif des méthodes de résolution des problèmes d'ordonnancement d'atelier. 21

Figure 3.1 – Une première séquence V-shaped. 28

Figure 3.2 – Une deuxième séquence V-shaped. 28

Figure 4.1 – Séquence n'est pas V-shaped, (ici $P_j > \max(P_i, P_k)$). 33

Figure 4.2 – Construction de séquence V-shaped par l'heuristique EC1.1. 34

Figure 4.3 – Construction de séquence V-shaped par l'heuristique EC1.2. 34

Figure 4.4 – Permutation des deux tâches se trouvant dans les positions i et j.....35

Figure 4.5 – Représentation de la séquence des tâches dans le cas n est pair. 36

Figure 4.6 – Représentation de la séquence des tâches dans le cas n est impair. 36

Liste des tableaux

Tableau 1.1 – Critères d'optimisation.....15

Tableau 4.1 – Résultats de comparaison des instances dans (Eilon et Chowdhury, 1977). 39

Tableau 4.2 – Résultats de la comparaison des instances dans (Gupta et al., 1990). .40

Tableau 4.3 – Résultats de la comparaison des instances dans (Srirangacharyulu and Srinivasan, 2010). 41

Liste des algorithmes

Algorithme 4.1 – L'heuristique EC1.1 33

Algorithme 4.2 – L'heuristique EC1.2 34

Algorithme 4.3 – L'heuristique H.....37

Introduction

L'ordonnancement occupe une place importante dans l'industrie et les services. Il s'attache à l'allocation des ressources aux tâches sur des périodes données de temps, et dont l'objectif est d'optimiser un ou plusieurs critères. Les ressources et les tâches peuvent prendre différentes formes en pratique. Les ressources peuvent être des machines, des ouvriers ou tout simplement des outils de fabrication dans un atelier de production, elles peuvent être des unités de calcul ou des serveurs dans un environnement informatique. Les tâches sont des opérations sur des produits à fabriquer sur les machines dans un système de production, elles peuvent être des programmes à exécuter ou simplement des requêtes à transmettre sur des serveurs dans un environnement informatique. L'ordonnancement contribue à l'élaboration de meilleures stratégies à adopter pour l'optimisation et l'utilisation des ressources.

L'ordonnancement d'atelier consiste à organiser dans le temps le fonctionnement d'un atelier pour utiliser au mieux les ressources humaines et matérielles disponibles dans le but de produire les quantités désirées dans le temps imparti. Le problème d'ordonnancement à une seule machine consiste à rechercher la séquence optimale de n tâches sur une seule machine. Une solution est décrite par l'ordre de passage des tâches sur la machine.

Dans ce mémoire, nous nous intéressons au problème de la minimisation de la variance des temps de fin d'exécution des tâches sur une seule machine. Ce problème est classé NP-difficile [23], il se formule comme un programme quadratique en nombres binaires (0/1). Pour le résoudre, nous proposons une heuristique en élaborant un code de calcul en langage Matlab permettant de calculer une solution approchée en temps raisonnable et plausible.

Ce mémoire est composé de quatre chapitres, et d'une conclusion générale.

Le premier chapitre aborde les principales notions et notations qui sont communes à tous les problèmes d'ordonnancement d'atelier à une seule machine que l'on retrouve dans la littérature scientifique. Les principaux concepts et définitions sont rappelés ainsi que leur développement mathématique qui va nous être utile tout au long de ce mémoire.

Dans le deuxième chapitre, nous rappelons en premier temps, les notions de base en matière de la complexité des problèmes combinatoire de manière générale, ensuite, nous exposons les méthodes de résolution exactes et approchées les plus utilisées dans la pratique, pour la résolution des problèmes d'ordonnancent classés NP-difficile.

Dans le troisième chapitre, nous exposons d'abord, le problème de la minimisation de CTV, puis nous évoquons l'importance pratique du problème dans différents secteurs de la vie quotidienne. Ensuite, les propriétés de dominance qui permettent de réduire l'espace de recherche et de localiser la séquence optimale, ainsi que la notion de la séquence duale, sont rappelées. Au final, nous présentons un programme quadratique à contraintes linéaires en nombres bivalents (0/1), provenant de la littérature. Sa résolution exacte est difficile et les tailles solvables sont assez réduites.

Le quatrième chapitre est consacré à l'exposition de notre contribution à la résolution du problème $1 \parallel CTV$, et à l'implémentation de notre approche de résolution, en proposant une heuristique constructive, notée H. Nous comparons aussi les résultats numériques obtenus en testant notre programme Matlab sur des exemples du problème tirés de la littérature. L'heuristique H est comparée avec l'heuristique EC1.2 qui est due à Eilon et Chowdhury 1977.

Au final, le manuscrit se termine par une conclusion générale, et des perspectives de recherche envisagées.

1

Notions de base sur l'ordonnancement d'atelier

1.1 Introduction

Dans ce chapitre, nous abordons les principales notions et notations communes à tous les problèmes d'ordonnancement d'atelier à une seule machine que l'on retrouve dans la littérature. Les principaux concepts et définitions sont rappelés ainsi que leur développement mathématique qui va nous être utile tout au long de ce mémoire.

1.2 Généralités et définitions

L'ordonnancement est un processus de décision qui apparait dans la plupart des systèmes de production et de transport ainsi que dans la gestion et le suivi de projet. Nous tirons de la littérature les deux définitions ci-dessous de l'ordonnancement :

Définition 1.1. Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution.

Définition 1.2. Une tâche est un travail élémentaire nécessitant un certain nombre d'unités de temps et de ressources. Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leur date de début.

Ces deux définitions nous permettent de déterminer les éléments caractérisant un problème d'ordonnancement.

Un problème d'ordonnancement est composé de façon générale d'un ensemble de tâches soumises à certaines contraintes, et dont l'exécution nécessite des ressources. Résoudre un problème d'ordonnancement, consiste à organiser ces tâches, c'est à dire à déterminer leur dates de démarrage et à leur attribuer des ressources, de telles sorte que les contraintes soient respectées.

Une tâche est un travail (ou job) dont la réalisation nécessite un nombre d'opérations élémentaires. Chaque opération élémentaire nécessite un certain nombre d'unités de temps (sa durée) et d'unités de ressources.

Une ressource est un moyen, technique ou humain, permettant la réalisation des tâches et dont la disponibilité limitée ou non est connue à priori. On distingue deux types de ressources : les ressources renouvelables et les ressources consommables. Une ressource est renouvelable si après avoir été allouée à une tâche, elle redevient disponible pour les autres. Une ressource est consommable si après avoir été allouée à une tâche, elle n'est plus disponible pour les tâches restant à exécuter.

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

Les contraintes temporelles intègrent :

- Les contraintes de temps alloué, issues généralement d'impératifs de gestion et relatives aux dates limites des tâches (délais de livraison, disponibilité des approvisionnements) ou à la durée totale d'un projet ;
- Les contraintes d'antériorité et plus généralement les contraintes de cohérence technologique, qui décrivent le positionnement relatif de certaines tâches par rapport à d'autres ;
- Les contraintes de calendrier liées au respect d'horaires de travail … etc.

Les contraintes de ressources traduisent le fait que les ressources sont disponibles en quantité limitée. On distingue deux types de contraintes de ressources, liées à la

nature disjonctive ou cumulative des ressources. Une ressource disjonctive ne peut être utilisée que par une tâche à la fois. Par contre dans une ressource cumulative les ensembles de tâches non réalisables simultanément sont de cardinalité quelconque.

D'une manière générale, trois types d'objectifs sont essentiels dans la résolution des problèmes d'ordonnancement : l'utilisation efficace des ressources, un délai d'exécution des tâches aussi faible que possible et le respect des dates d'achèvement prescrites à l'avance.

Les problèmes d'ordonnancement sont classés par familles de problèmes selon la structure de l'atelier, on distingue :

- **Atelier à une machine** : l'atelier est composé soit d'une seule machine (un bloc/outil de production) ou l'essentiel de l'atelier peut être représenté par une seule machine (machine goulot par exemple). Pour ce modèle chaque tâche est constituée d'une seule opération qui doit être effectuée sur cette machine.
- **Atelier à plusieurs machines parallèles** : chaque tâche est constituée d'une seule opération qui peut être effectuée par plusieurs machines. Les machines peuvent être identiques ou différentes. Pour ce type d'ateliers, en plus du problème de succession des tâches sur les machines, il faut aussi décider de leur affectation sur les ressources.
- **Atelier FlowShop** : les machines sont disposées en ligne et toutes les tâches ont la même gamme opératoire, c'est-à-dire, chaque tâche doit passer sur chaque machine et cela dans le même ordre pour toutes les tâches.
- **Atelier JobShop** : les machines sont disposées en ligne, mais chaque tâche à sa propre gamme opératoire, c'est-à-dire, chaque tâche dispose de son propre ordre de passage sur les machines.
- **Atelier OpenShop** : ici la numérotation des machines est immatérielle. La gamme de fabrication des tâches n'est pas fixée à l'avance, elle est déterminée lors du traitement des tâches.
- **Atelier Flexible** : l'atelier peut être de type FlowShop ou JobShop, mais chaque étage est composé de plusieurs machines parallèles identiques ou différentes.

1.3 Description d'un problème d'ordonnancement

Les données d'un problème d'ordonnancement sont les tâches et leurs caractéristiques, les contraintes potentielles, les ressources et le(s) critère(s) d'optimisation. Une tâche J_j ne peut commencer son exécution avant une date de disponibilité r_i et doit être achevée avant une date échue d_i. L'intervalle $[r_i, d_i]$ est appelé la fenêtre de temps d'exécution de la tâche J_i. Les tâches sont souvent liées entre elles par des relations d'antériorité, sinon, on dit qu'elles sont indépendantes.

1.3.1 Notations

Nous utilisons les notations présentées dans la plupart des ouvrages et articles sur le sujet.

$J = \{1,2,\dots,n\}$	L'ensemble des tâches ou (jobs)
p_j	Temps opératoire (Processing time)
r_j	Date de disponibilité (Release date)
d_j	Date d'échue (Due date)
S_j	Début d'exécution (Start time)
$W_j = S_j - r_j$	Temps d'attente (Waiting time)
$C_j = S_j + p_j$	Temps de fin (Completion time)
$F_j = C_j - r_j$	Temps du flot (Flow time)
$L_j = C_j - d_j$	Retard algébrique (Lateness)
$T_j = \max(C_j - d_j, 0)$	Retard (Tardiness)
$E_j = \max(d_j - C_j, 0)$	Avance (Earliness)
$U_j = \begin{cases} 1, & C_j > d_j \\ 0, & \text{sinon} \end{cases}$	Indicatrice (Unit penality)

1.3.2 Représentation d'une tâche

Nous représentons une tâche par un rectangle dont la longueur représente la durée ou le temps de traitement. L'axe horizontal représente le temps, et l'axe vertical représente les machines.

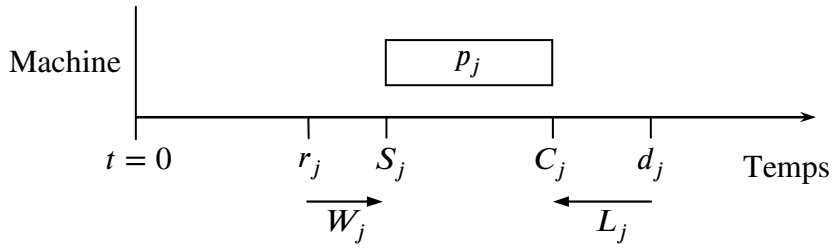


Figure 1.1 – Caractéristiques de la tâche j.

La résolution des problèmes d'ordonnancement nécessite le calcul ou l'évaluation de critères de performance. Ces critères seront décrits dans la section suivante.

1.3.3 Critères d'optimisation

Le choix d'un ordonnancement se fait en fonction d'un ou plusieurs critères que l'on cherche à optimiser. Généralement, le critère considéré est une fonction des dates de fin d'exécution :

$$f(C_1, C_2, \ldots, C_n) \tag{1.1}$$

Les mesures les plus considérées, dans la littérature, sont données dans le tableau ci-dessous.

Tableau 1.1 – Critères d'optimisation.

Critères	Maximum	Moyenne	Moyenne pondérée
Fin de traitement	$C_{\max} = \max_{1 \le i \le n}(C_i)$	$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$	$\bar{C}_w = \sum_{i=1}^{n} w_i C_i$
Décalage	$L_{\max} = \max_{1 \le i \le n}(L_i)$	$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i$	$\bar{L}_w = \sum_{i=1}^{n} w_i L_i$
Retard	$T_{\max} = \max_{1 \le i \le n}(T_i)$	$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$	$\bar{T}_w = \sum_{i=1}^{n} w_i T_i$
Avance	$E_{\max} = \max_{1 \le i \le n}(E_i)$	$\bar{E} = \frac{1}{n}\sum_{i=1}^{n} E_i$	$\bar{E}_w = \sum_{i=1}^{n} w_i E_i$
Nombre de retards		$\bar{U} = \frac{1}{n}\sum_{i=1}^{n} U_i$	$\bar{U}_w = \sum_{i=1}^{n} w_i U_i$

Une caractéristique intéressante pour un critère est sa régularité. Tous les critères vus précédemment, sont des fonctions monotones croissantes en fonction des dates de fin d'exécution C_j ,($j \in J$) elles vérifient :

$$\forall\, j \in J \,:\, C_j \leq C'_j \Longrightarrow\; f(C_1, C_2, \ldots, C_n) \;\leq\; f\left(C'_1, C'_2, \ldots, C'_n\right) \tag{1.2}$$

1.3.4 Classification des problèmes d'ordonnancement

Les classes de problèmes d'ordonnancement sont spécifiées en termes de classification à trois champs $\alpha|\ \beta\ |\ \gamma$ introduite par (Graham,1979) [15]. La première entrée α spécifie l'environnement du processeur, β précise les caractéristiques des tâches (contraintes) et γ désigne le critère d'optimalité (fonction objective).

1.3.5 Représentation des ordonnancements

Le diagramme de Gantt, du nom de son développeur Henry Gantt (1861-1919), est le moyen le plus simple et le plus utilisé pour représenter un ordonnancement, a la fois dans la littérature et dans les entreprises. Dans le cas de problèmes d'atelier, ce diagramme se compose de plusieurs lignes horizontales, chacune d'entre elles désignant une machine. Les opérations exécutées sur une machine donnée sont représentées sous forme de barres ayant des longueurs proportionnelles à leurs temps opératoires. L'axe des abscisses représente le temps, les barres sont positionnées aux dates de début d'exécution des opérations. Ainsi, le diagramme de Gantt permet de montrer l'occupation des machines dans le temps, les séquences de traitement sur chaque machine et les dates de fin des travaux.

Les problèmes d'ordonnancement d'ateliers sont des problèmes combinatoires extrêmement difficiles et il n'existe pas de méthodes universelles permettant de les résoudre efficacement. Une classification de ces problèmes selon leur difficulté est donnée par exemple, dans [4, 5, 12, 34].

1.4 Modèle mathématique

Il existe deux types de modélisation d'un problème d'ordonnancement. La modélisation graphique sous forme de graphe caractérisant la précédence, priorité conflit…, et la représentation analytique sous forme de programme mathématique que nous allons le présenter dans cette section.

La variable de décision :

$$x_{ij} = \begin{cases} 1, & \text{si la tâche } j \text{ occupe la position } i \\ 0, & \text{sinon} \end{cases}$$

Programme mathématique en nombres entiers :

$$\text{Minimiser} \qquad f(C_1, C_2, \dots, C_n) \tag{1.3}$$

$$S_j \geq \sum_{j=1}^{n} r_j x_{ij} \qquad (i = 1, \dots, n) \tag{1.4}$$

$$S_{j+1} \geq C_j \qquad (j = 1, \dots, n-1) \tag{1.5}$$

$$C_i \geq S_i + \sum_{j=1}^{n} p_j x_{ij} \qquad (i = 1, \dots, n) \tag{1.6}$$

$$T_i - E_i \geq C_i - \sum_{j=1}^{n} d_j x_{ij} \qquad (i = 1, \dots, n) \tag{1.7}$$

$$T_j \geq 0\,, \text{ et } E_j \geq 0 \qquad (j = 1, \dots, n) \tag{1.8}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \dots, n) \tag{1.9}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \dots, n) \tag{1.10}$$

$$x_{ij} \in \{0\,, 1\} \qquad (i, j = 1, \dots, n) \tag{1.11}$$

Il s'agit d'un programme quadratique en nombres bivalents (0/1). La taille de l'espace de recherche engendré par la variable de décision est de l'ordre 2^{n^2}.

Remarque 1.1. La résolution exacte de ce programme est difficile et les tailles solvables exactement sont assez réduites.

1.5 Conclusion

Dans ce chapitre, nous avons introduit les principales notions et notations communes à tous les problèmes d'ordonnancement d'atelier à une seule machine que l'on retrouve dans la littérature scientifique. Les principaux concepts et définitions sont rappelés ainsi que leur développement mathématique qui va nous être utile tout au long de ce mémoire.

2

Méthodes de résolution d'un problème d'ordonnancement

2.1 Introduction

Dans ce chapitre, nous rappelons tout d'abord, des notions de base relevant de la complexité des problèmes combinatoire de manière générale. Ensuite, nous exposons les méthodes de résolution exactes et approchées les plus connues pour la résolution des problèmes d'ordonnancent classés NP-difficile.

2.2 Complexité des problèmes

La théorie de la complexité (Cook, 1971) [10] a pour but d'analyser les coûts de résolution, en termes de temps de calcul, des problèmes d'optimisation combinatoire. Elle permet d'établir une classification des problèmes en plusieurs niveaux de difficulté. Parmi les classes de problèmes combinatoires, on distingue les problèmes de décision et les problèmes d'optimisation.

- Un problème d'optimisation est un problème pour lequel on doit chercher une solution admissible optimisant au mieux une fonction objective.
- Un problème de décision est un énoncé auquel la réponse peut être uniquement OUI ou NON. Chaque problème d'optimisation possède un problème de décision correspondant.

Par la suite, nous avons besoin de ces deux définitions :

Définition 2.1. (*Algorithme polynômial*). Un algorithme est dit polynômial si le nombre d'opérations élémentaires, nécessaires pour atteindre le résultat, est majoré par un polynôme de la taille des données n, par exemple $\mathcal{O}(n^{\alpha})$ avec α est une constante positive indépendante de n.

Définition 2.2. (*Algorithme exponentiel*). Un algorithme est dit exponentiel si le nombre d'opérations élémentaires est minoré par une forme exponentielle de la taille des données n, par exemple $\mathcal{O}(a^{n})$ avec $(a > 1)$ est une constante indépendante de n.

La classe des problèmes NP (pour Non déterministe Polynomial) est la classe des problèmes de décision pouvant être résolus par un algorithme polynomial sur une machine de Turing non déterministe (Garey, 1979) [14]. C'est-à-dire, les problèmes de décision pour lesquels il existe un algorithme tel que si on lui fournit une solution dont la réponse est OUI, il est capable de la vérifier en temps polynomial.

Parmi la classe NP, on distingue deux grandes sous-classes : la classe des problèmes polynomiaux (la classe P) et la classe des problèmes NP-complet (la classe NPC).

La classe P est la classe des problèmes de décision solvables par un algorithme polynomial déterministe, c'est-à-dire, les problèmes de décision pour lesquels il existe un algorithme polynomial permettant de répondre à la question par un OUI. C'est la classe des problèmes les plus « faciles » de NP.

La classe NPC est la classe des problèmes dont il n'existe aucun algorithme polynomial pour les résoudre, sous l'hypothèse ($P \neq NP$). Cependant, si on peut résoudre un problème de NPC en un temps polynomial, on obtient par transformation des algorithmes polynomiaux pour tous les problèmes de NPC.

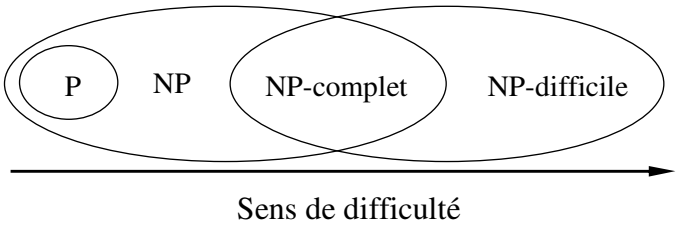


Figure 2.1 – Classification des problèmes combinatoires.

Quand un problème est NP-Complet, il n'est pas possible, à l'heure actuelle, de construire un algorithme polynomial le résolvant. Cependant, dans certains cas, on peut construire un algorithme efficace appelé *schéma d'approximation polynomial* (PTAS) ou *schéma d'approximation totalement polynomial* (FPTAS), le problème étudié est alors dit NP-Complet au sens faible. Dans le cas contraire, il est dit NP-Complet au sens fort.

Un problème d'optimisation est dit NP-Difficile si le problème de décision qui lui correspond est NP-Complet.

Pour résoudre un problème d'ordonnancement de manière efficace, il faut prendre en compte la particularité de chaque problème. C'est pourquoi il y a autant d'approches que de problèmes d'ordonnancement. Toutefois, il existe des approches qui fournissent des solutions approchées de bonne qualité pour un certain nombre de problèmes.

2.3 Méthodes de résolution

Dans cette section, nous passons en revue les deux approches exactes et approchées pour la résolution des problèmes d'ordonnancement.

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique en nombres entiers, programmation dynamique, procédure par séparation et évaluation, théorie des graphes,…). Ces méthodes garantissent en général l'optimalité de la solution fournie, mais les algorithmes dont la complexité n'est pas polynomiale ne peuvent pas être utilisés pour résoudre des problèmes de grande taille, d'où la nécessité de construire des méthodes de résolution approchées, qu'on appelle méthodes heuristiques ou métaheuristique pour ces problèmes NP-difficiles [25].

Dans la suite, nous présentons les méthodes de résolution les plus connues. Nous commençons par les méthodes exactes, ensuite on présentera les méthodes approchées.

Nous présentons ci-dessus, un schéma général des méthodes de résolution d'un problème d'ordonnancement.

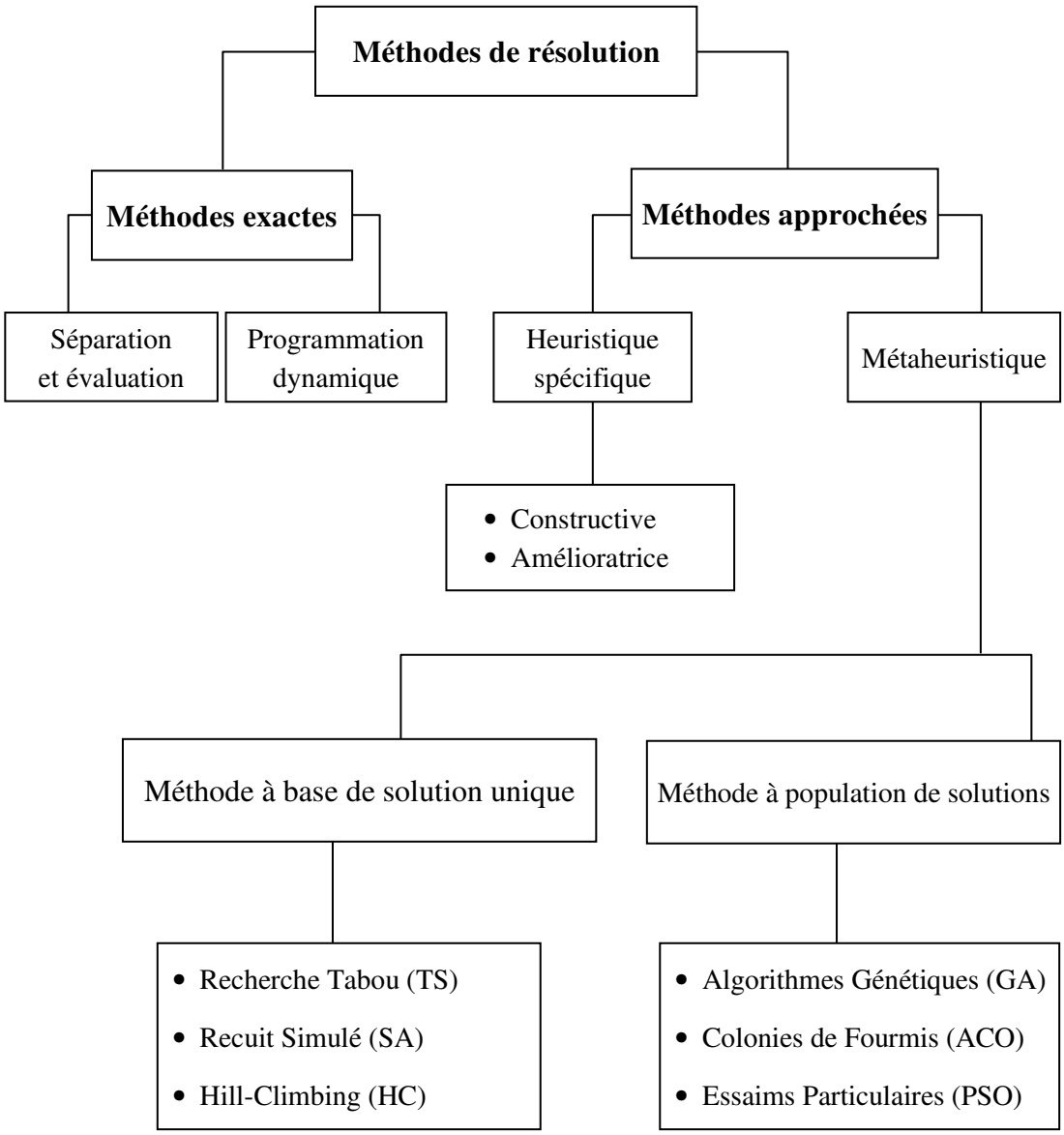


Figure 2.2 – Schéma récapitulatif des méthodes de résolution des problèmes d'ordonnancement d'atelier.

2.3.1 Méthodes exactes

Généralement, il n'est pas possible de construire un algorithme polynomial résolvant un problème d'optimisation NP-difficile. Toutefois, on peut mettre au point des méthodes efficaces pour une grande partie des énoncés. Pour certains problèmes, on appliquera une Procédure par Séparation et Évaluation (PSE), pour d'autres, la programmation dynamique (PD) ou encore la programmation linéaire en nombres entiers (PLNE). Une référence dans le domaine est celle développée par [5].

2.3.2 Méthodes approchées

Il existe deux types de méthodes approchées, à savoir les heuristiques et les métaheuristiques.

2.3.2.1 Heuristique

Le principe général d'une méthode heuristique est de trouver en un temps polynomial une solution réalisable la meilleure possible. Elle dépende de la nature du problème étudié.

2.3.2.2 Les méthodes constructives

Le principe de ces méthodes est de construire une solution à partir des données initiales. Ce sont des méthodes itératives, où à chaque itération, on complète une solution partielle.

Les méthodes constructives se distinguent par leur rapidité et leur grande simplicité. Le principal défaut de ces méthodes réside dans la qualité mauvaise des solutions obtenues. Le fait de vouloir opérer à tout prix le meilleur choix à chaque étape est une stratégie dont les effets peuvent être catastrophiques à long terme.

2.3.2.3 Méthodes amélioratrices

Le principe de ces méthodes n'est plus de construire un ordonnancement a partir des données initiales du problème, mais de modifier le résultat d'une solution admissible en vue d'améliorer la valeur de la fonction objective. La plupart de ces méthodes utilisent la notion de voisinage de solution, on parle aussi de méthodes par voisinage.

À partir d'une solution initiale générée aléatoirement ou avec une heuristique, une méthode par voisinage ou encore méthode de recherche locale réalise un processus itératif qui consiste à remplacer la solution courante par l'un de ses voisins en tenant compte de la fonction objectif. La méthode s'arrête et retourne la meilleure solution trouvée. Le critère d'arrêt peut être un nombre d'itérations atteint ou une valeur de la fonction objective atteinte. Ces méthodes diffèrent essentiellement par le système de voisinage utilisé et la stratégie de parcours du système de voisinage.

2.3.2.4 Métaheuristique

Une métaheuristique est un ensemble de concepts universels, applicables à une large classe de problèmes d'optimisation combinatoire. On distingue deux types : les méthodes à base d'individu, telles que le Recuit Simulé (Simulated Annealing SA) et la Recherche Tabou (Tabu Search, TS), et les méthodes à population telles que l'Algorithme Génétique (Genetic Algorithm, GA) et la Colonie de fourmis (Ant Colony Optimisation, ACO). Pour le plus amples d'informations, le lecteur peut consulter [1, 13, 28, 29, 31].

Remarque 2.1 Les méthodes exactes ne sont efficaces que pour les instances de problèmes de petite taille.

2.4 Conclusion

Dans ce chapitre, nous avons rappelé tout d'abord, des notions de base en matière de la complexité des problèmes combinatoire de manière générale. Ensuite nous avons présenté des méthodes de résolution exactes et approchées les plus utilisées dans la littérature. Une adaptation de ces outils pour la résolution des problèmes d'ordonnancement d'atelier classés NP-difficile est indispensable.

3

Minimisation de la variance des dates de fin d'exécution (CTV)

3.1 Introduction

Le problème de CTV a été proposé pour la première fois par Merten et Muller en 1972 [30], il consiste à minimiser la variance des dates de fin d'exécution des tâches dans un ordonnancement à une seule machine. Il a fait l'objet depuis quelques années d'une attention croissante pour deux raisons, outre son intérêt théorique, il a de nombreuses applications dans le domaine de la production industrielle, de la logistique et de l'informatique. Ce problème NP-difficile (Kubiak, 1993) [23] a été largement étudié et résolu par un grand nombre de chercheurs dans le cas général par des heuristiques constructives et amélioratrices, ainsi que par des métaheuristiques qui vont des méthodes à solution unique jusqu'aux méthodes à population de solutions.

Dans ce chapitre, nous exposons d'abord, le problème de la minimisation de CTV, puis nous évoquons l'importance pratique du problème dans différents secteurs de la vie quotidienne. Ensuite, les propriétés de dominance les plus pertinentes qui permettent de réduire l'espace de recherche et de localiser la séquence optimale, sont rappelées. Au final, nous présentons un programme quadratique à contraintes linéaires en nombres bivalents (0/1), découlant de la littérature. Sa résolution exacte est difficile et les tailles solvables sont assez réduites.

3.2 Problème de CTV

Dans cette section, nous présentons la position du problème étudié.

3.2.1 Notations

Dans ce qui suit, nous allons adopter les notations suivantes :

$J = \{1,2,\dots,n\}$	L'ensemble des tâches indépendantes ou (jobs)
p_j	Temps opératoire de la tâche j.
π	Séquence arbitraire
π^*	Séquence optimale
Π	L'ensemble de toutes les séquences,$\lvert\Pi\rvert = n!$
$C_j(\pi) = \sum_{i=1}^{j} p_i(\pi)$	Date de fin d'exécution de la tâche située dans la position j, sous la séquence π.
$M = \sum_{i=1}^{n} p_i$	Makespan

3.2.2 Hypothèses

En plus, nous adoptons les hypothèses suivantes :

1. Les temps d'exécution des tâches sont connus et déterministes, et tous les tâches sont disponibles à l'instant ($t = 0$).
2. La machine peut traiter au plus une tâche à la fois, et les temps d'inactivité et la préemption des tâches ne sont pas autorisés.
3. Les temps d'installation sont inclus dans les temps d'exécution.
4. Il n'y a pas de relations de précédence entre les tâches, alors le problème est classé 1 || CTV selon la notation introduite par Graham en 1979, voir [15].

3.2.3 Critère de performance

Nous séquençons n tâches indépendantes de temps d'exécution $p_1, p_2, \dots, p_n$, sur une seule machine. L'objectif est de trouver la séquence des tâches π^* qui minimise la variance des dates de fin d'exécution, définie par :

$$\mathrm{CTV}(\pi) = \frac{1}{n}\sum_{j=1}^{n}\left(C_j(\pi) - \bar{C}(\pi)\right)^2 \tag{3.1}$$

où, $\bar{C}(\pi)$ est la moyenne des dates de fin d'exécution dans la séquence π.

La variance des dates de fin d'exécution est un critère quadratique irrégulier c'est-à-dire, sa valeur peut diminuer lorsqu'un ou plusieurs temps d'exécution augmente.

La valeur du critère CTV peut s'écrire ainsi :

$$\mathrm{CTV}(\pi) = \frac{1}{n}\sum_{j=1}^{n}\left(T_j^2(\pi) + E_j^2(\pi)\right) \tag{3.2}$$

où, $T_j(\pi) := \max(C_j(\pi) - \bar{C}(\pi), 0)$ représente le retard de la tâche j, et $E_j(\pi) := \max(\bar{C}(\pi) - C_j(\pi), 0)$ représente l'avance de la tâche j.

Ce critère pénalise à la fois, les tâches qui se terminent en retard et en avance de manière symétrique. En d'autre terme, elles ont la même pondération, comme l'indique l'équation (3.2).

3.3 Importance pratique

Le problème est spécialement important pour tout système industriel où toutes les tâches nécessitent un traitement uniforme, c'est-à-dire les tâches se terminent presque en même temps dans le système. nous retrouvons le problème dans la planification de production, la gestion des fichiers, les systèmes de calcul sur les grandes bases de données, et la transmission de groupes des données dans le réseau internet. La minimisation du critère de performance CTV pénalise à la fois le retard et l'avance des tâches de manière symétrique, ce qui se concorde avec la philosophie de production baptisée Juste-à-Temps (JIT) adoptée actuellement par la plupart des institutions industrielles afin d'assurer et de stabiliser la Qualité de Service (QoS) dans leur système de production.

Maintenant, Nous établissons quelques propriétés importantes de solutions optimales au problème formulé ci-dessus. Le concept de séquences V-shaped a été introduit pour la première fois par (Eilon et Chowdhury, 1977) [11] dans le contexte du

problème de minimisation de la variance des temps d'attente WTV (Waiting time variance), voir [43].

3.4 Propriétés de dominance

Dans cette section, nous présentons une propriété importante qui va jouer un rôle fondamental dans la résolution des problèmes d'ordonnancement NP-difficile. Avant ça, nous rappelons les deux définitions suivantes :

Définition 3.1 (*Propriété de dominance*) Toute propriété ou une condition nécessaire d'optimalité qui permet de réduire énormément l'espace de recherche en localisant la solution optimale dans un sous-ensemble inclus dans l'espace de recherche de départ.

Définition 3.2 (*Ordonnancement V-shape*) Un ordonnancement est dit V-shape si les tâches qui précèdent la tâche minimale sont ordonnées LPT, et les tâches qui succèdent la tâche minimale sont ordonnées SPT.

Du fait de sa complexité, le problème CTV a suscité beaucoup d'intérêt de la part de la communauté scientifique. De nombreux résultats théoriques ont été obtenus. Dans cette section, on ne cite que les plus intéressants.

3.4.1 Conditions nécessaires d'optimalité

Nous présentons des résultats classiques permettant de caractériser la solution optimale pour le problème $1 \parallel \text{CTV}$. Sans perte de généralité, nous supposons que $p_1 \leq p_2 \leq \cdots \leq p_n$.

Théorème 3.1 (Shrage, 1975) [36] Dans une séquence optimale, la première tâche la plus longue, est planifiée en premier.

Théorème 3.2 (Eilon and Chowdhury, 1977) [11] La solution optimale qui minimise la variance du temps de fin d'exécution est en forme V-shaped.

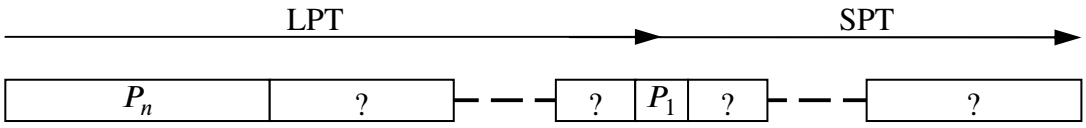


Figure 3.1 – Une première séquence V-shaped.

Théorème 3.3 (Hall and Kubiak,1991)[19]. Dans une séquence optimale, la première tâche la plus longue est planifiée en premier, la deuxième la plus longue est la dernière et la troisième la plus longue est la deuxième.

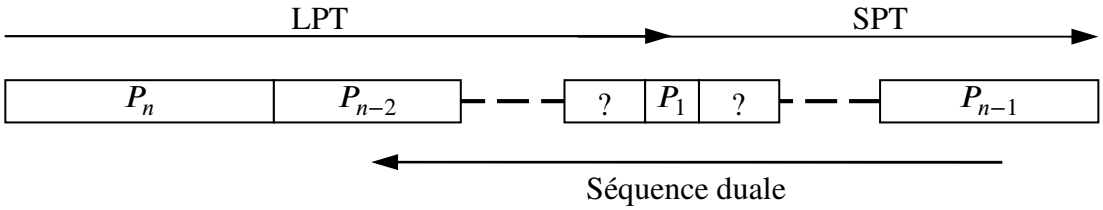


Figure 3.2 – Une deuxième séquence V-shaped.

3.4.2 Réduction de l'espace de recherche

- Il y a 2^{n-1} façons pour construire une séquence V-shape.
- Les conditions nécessaires précédentes, réduisent énormément, la taille de l'espace de recherche de $n!$ à 2^{n-1}.
- Cette propriété de dominance réduit considérablement l'espace de recherche et sert de base au développement d'algorithmes efficaces.

3.4.3 Séquence duale et propriétés

Dans cette sous section, nous présentons une définition importante à savoir la séquence duale ainsi que ses propriétés intrinsèques.

Définition 3.3 (*Séquence duale*) Soit $P = (p_1, p_2, \dots, p_n)$, tel que $p_1 \leq p_2 \leq \cdots \leq p_n$. Une séquence $\pi \in \Pi$ est dite séquence duale, notée $\pi^d \in \Pi$, si et seulement si, pour tout couple de positions (i, j) tel que $i + j = n + 2$, on a,

$$p_i(\pi^d) = p_j(\pi), \qquad \pi \in \Pi \tag{3.3}$$

Par conséquent, les propriétés suivantes sont valables :

$$C_1\left(\pi^d\right) = C_1(\pi) = p_n, \qquad \forall\, \pi \in \Pi \tag{3.4}$$

$$C_n\left(\pi^d\right) = C_n(\pi) = M, \qquad \forall\, \pi \in \Pi \tag{3.5}$$

Théorème 3.4 (Manna DK, Prasad VR, 1999) [11]. Soit $\pi \in \Pi$. Les séquences π et π^d procurent la même variance :

$$\mathrm{CTV}(\pi) = \mathrm{CTV}\left(\pi^d\right), \qquad \forall\, \pi \in \Pi \tag{3.6}$$

en plus, leurs moyennes sont symétrique par apport $(M + p_n)/2$:

$$\mathrm{CTM}(\pi) + \mathrm{CTM}\left(\pi^d\right) = M + p_n, \quad \forall\, \pi \in \Pi \tag{3.7}$$

3.4.4 Exemple Illustratif

Dans cette sous section, nous présentons l'évaluation de CTM et CTV sous différente séquence π, sur l'exemple : $P = (1, 5, 6, 9, 19)$.

Séquence SPT

π : SPT	1	2	3	4	5	**CTM**	**CTV**
$p_i(\pi)$	1	5	6	9	19		
$C_i(\pi)$	1	6	12	21	40	**16**	**188.4**

Séquence LPT

π : LPT	5	4	3	2	1	**CTM**	**CTV**
$p_i(\pi)$	19	9	6	5	1		
$C_i(\pi)$	19	28	34	39	40	**32**	**60.4**

Séquence V-shape

π	5	3	2	1	4	**CTM**	**CTV**
$p_i(\pi)$	19	6	5	1	9		
$C_i(\pi)$	19	25	30	31	40	**29**	**48.4**

Séquence duale de π

π^d	5	4	1	2	3	**CTM**	**CTV**
$p_i(\pi)$	19	9	1	5	6		
$C_i(\pi)$	19	28	29	34	40	**30**	**48.4**

3.5 Modèle mathématique

Dans cette section, nous exposons un programme quadratique en nombres binaires qui est bien connu dans la littérature, procurant des solutions optimales.

Sans perte de généralité, nous supposons que les temps d'exécution sont arrangés comme suit : $p_1 \leq p_2 \leq \cdots \leq p_n$, le problème peut se formuler ainsi :

$$\text{Minimiser} \quad \text{CTV} = \frac{1}{n}\sum_{j=1}^{n}\left(C_j - \bar{C}\right)^2 \tag{3.8}$$

$$\text{Sujet à} \quad C_1 = p_n \tag{3.9}$$

$$C_i = C_{i-1} + \sum_{j=1}^{n} p_j x_{ij} \qquad (i = 2, \ldots, n) \tag{3.10}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n) \tag{3.11}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n) \tag{3.12}$$

$$x_{ij} \in \{0\,,1\} \qquad (i, j = 1, \ldots, n) \tag{3.13}$$

Il s'agit d'un programme quadratique en nombres bivalents (0/1). La taille de l'espace de recherche engendré par la variable de décision est de l'ordre 2^{n^2}. La résolution exacte de ce programme est difficile et les tailles solvables exactement sont assez réduites.

La valeur du critère de performance (3.3) du modèle exacte (ME) n'est pas entière en générale. Dans la pratique, il est fortement recommandé que la valeur du critère à minimiser soit entière.

Formule classique de la variance (avec la moyenne).

$$n\,\text{CTV} = \sum_{j=1}^{n}\left(C_j - \bar{C}\right)^2 \tag{3.14}$$

Dans la pratique, nous minimisons la quantité entière ci-dessous, en utilisant une formule alternative de la variance (sans la moyenne).

$$2n^2\text{CTV} = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(C_i - C_j\right)^2 \tag{3.15}$$

3.6 Conclusion

Dans ce chapitre, nous avons exposé le problème de la minimisation de CTV, puis nous avons évoqué l'importance pratique du problème dans différents secteurs de la vie quotidienne. Ensuite, les propriétés de dominance les plus pertinentes qui permettent de réduire l'espace de recherche et de localiser la séquence optimale, sont rappelées. Au final, nous avons présenté un programme quadratique à contraintes linéaires en nombres bivalents (0/1) découlant de la littérature. Sa résolution exacte est difficile et les tailles solvables sont assez réduites.

4

Heuristique pour la minimisation de la variance

4.1 Introduction

Dans ce chapitre, nous exposons notre heuristique, notée H, ainsi que son implémentation pour la résolution du problème 1 || CTV. Nous comparons aussi des résultats numériques obtenus en testant notre programme MATLAB sur des exemples type tirés de la littérature en appliquant notre algorithme et l'heuristique EC1.2 développée par (Eilon et Chowdhury, 1977).

4.2 Approche de résolution

L'idée consiste à se contenter de chercher la séquence optimale dans un espace de recherche réduit de V-shape, que nous le notons $\overline{V}$. Cet espace est appelé V-shape symétrique qui est inclut dans l'espace de recherche V-shape initial V. Ici $\overline{V} \subset V$.

Nous adoptons la notation suivante :

$$\mathrm{CTV}(A) \coloneqq \min_{\pi \in A} \mathrm{CTV}(\pi) \tag{4.1}$$

Il est clair que,

$$\mathrm{CTV}(\overline{V}) \geq \mathrm{CTV}(V), \qquad \overline{V} \subset V \tag{4.2}$$

La taille de l'espace de recherche V-shape s'écrit en fonction de n : $|V| = 2^{n-1}$. Par la suite, nous allons montrer que la taille de l'espace de recherche V-shape symétrique s'écrit $\left|\overline{V}\right| = 2^{n/2} = (1.414)^n$, ce qui constitue une réduction considérable en matière de l'espace de recherche de départ.

4.2.1 L'espace V-Shape

Eilon et chowdhury ont remarqué que dans une séquence de tâches quelconque, s'il existe 3 tâches consécutives (i, j, k) (voir figure 4.1) dont la tâche centrale vérifie $P_j > \max(P_i, P_k)$, alors, la valeur de CTV peut être toujours améliorée. Par conséquent ils déduisent que la séquence optimale doit être V-shape.

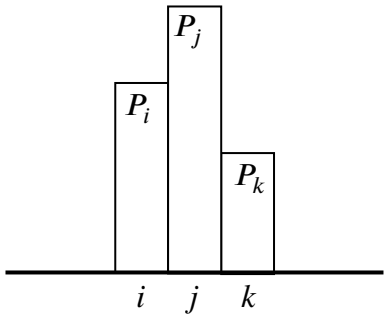


Figure 4.1 – Séquence n'est pas V-shaped, (ici $P_j > \max(P_i, P_k)$).

4.2.2 Les heuristiques EC1.1 et EC1.2

Dans la littérature, il existe beaucoup d'heuristiques efficaces qui permettent de procurer une séquence V-shape minimisant la variance CTV. Dans cette section, nous présentons deux heuristiques constructives : à savoir EC1.1 et EC1.2. Par la suite, sans perte de généralité nous supposons que $P_1 \leq P_2 \leq \cdots \leq P_n$.

Algorithme 4.1 – L'heuristique EC1.1

étape (1) Supprimer la tâche dont le temps d'exécution est le plus long du pool des tâches et placer-la avant la fin de la séquence de tâches, et la fin de la séquence de tâches contient une autre tâche,

étape (2) Retirer la tâche dont le temps d'exécution est le plus long du pool de travaux, placer-la après le début de la séquence de tâches, et l'entête de la séquence de tâches contient une tâche supplémentaire,

étape (3) Répéter les étapes : 1 et 2 jusqu'à ce que le pool de tâches soit vide.

Algorithme 4.2 – L'heuristique EC1.2

étape (1) Générer une séquence V-shape en appliquant l'heuristique EC1.1,

étape (2) Permuter la 2^{eme} tâche P_{n-2} avec la dernière P_{n-1}.

Les figures 4.2 et 4.3 représentent les séquences engendrées par les heuristiques EC1.1 et EC1.2, respectivement. Pour le plus ample d'informations sur les heuristiques rappelées ci-dessus, nous invitons le lecteur à consulter l'article [11].

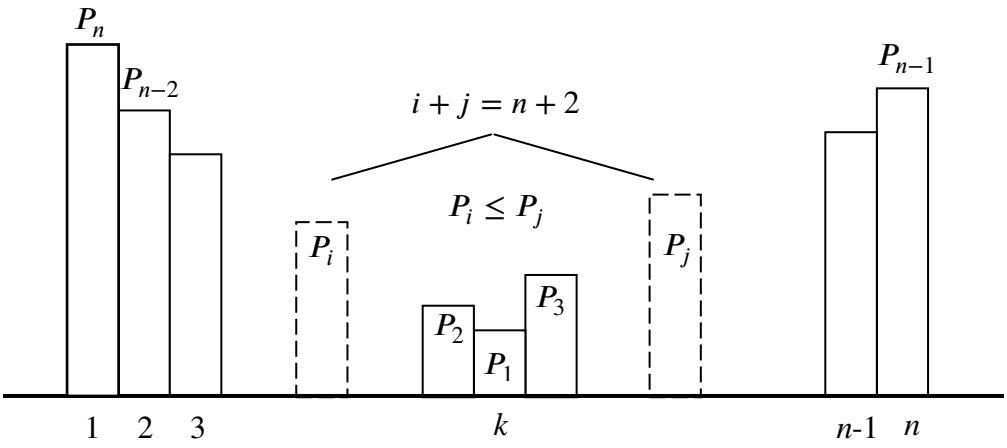


Figure 4.2 – Construction de séquence V-shaped par l'heuristique EC1.1.

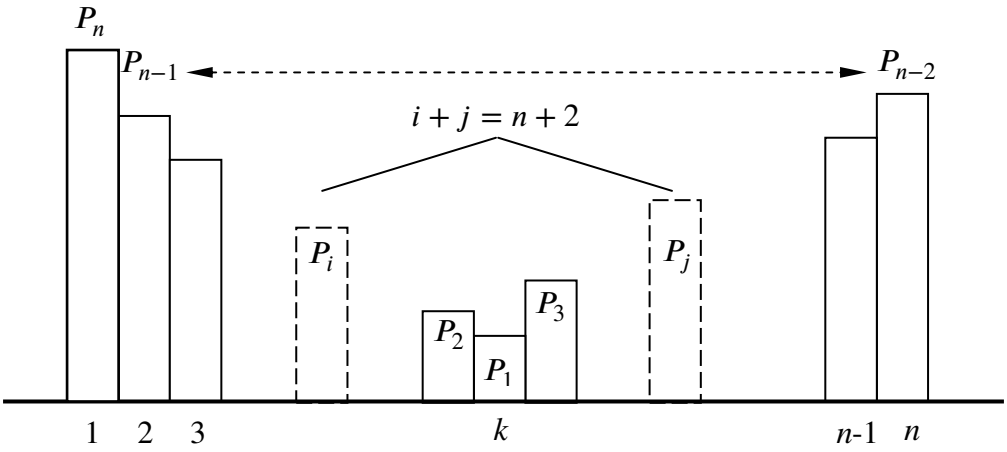


Figure 4.3 – Construction de séquence V-shaped par l'heuristique EC1.2.

4.2.3 L'espace V-Shape symétrique et l'heuristique H

Dans cette sous section, nous présentons, tout d'abord la notion de V-shape symétrique, puis nous exposons notre heuristique constructive H, pour trouver une séquence approchée au problème 1ll CTV .

4.2.3.1 Génération de l'espace V-shape symétrique

L'espace V-shape symétrique noté $\bar{V}$, est un sous-espace V-shape V, $\bar{V} \subset V$. Il peut être générer comme suit,

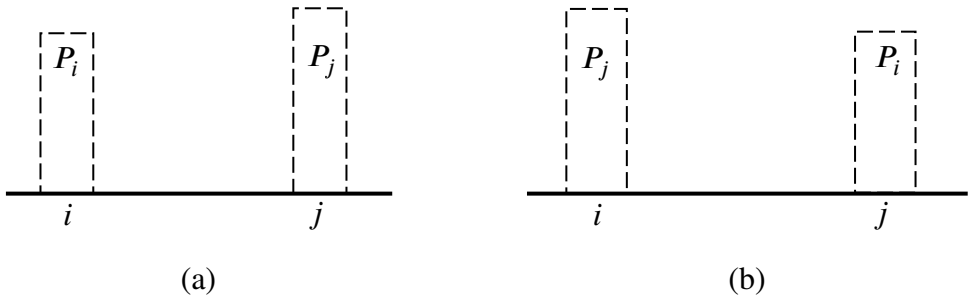
- Générer une séquence V-shape en appliquant l'heuristique EC1.1.
- Permuter les tâches occupant les positions (i, j) telles que $i + j = n + 2$.

4.2.3.2 La taille de l'espace V-shape symétrique

L'ensemble de toutes les positions (permutations), est caractérisé par

$$E := \left\{(i, j) \in \{1,2, \ldots, n\}^2 \,:\, i \leq j, \text{ et } i + j = n + 2\right\} \tag{4.3}$$

La taille de cet espace de recherche est de l'ordre $|E| = 2^{n/2} \approx 1.414^n$.



(a) (b)

Figure 4.4 – Permutation des deux tâches se trouvant dans les positions i et j.

On distingue deux cas : le cas où, le nombre de tâches n est pair et impair, (voir figure 4.5, et 4.6).

Cas 1 : $n = 2k$, (n est pair).

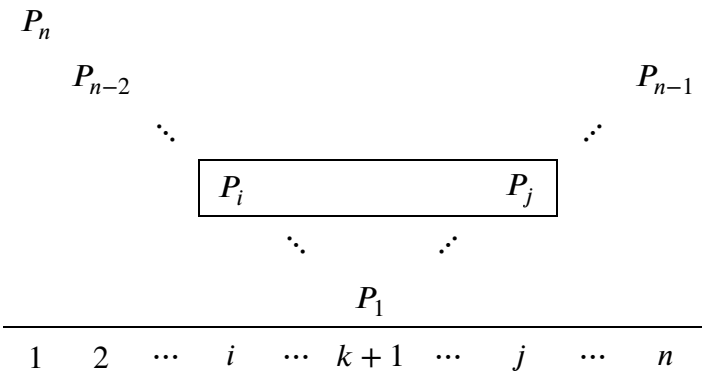


Figure 4.5 – Représentation de la séquence des tâches dans le cas n est pair.

Cas 2 : $n = 2k + 1$, (n est impair).

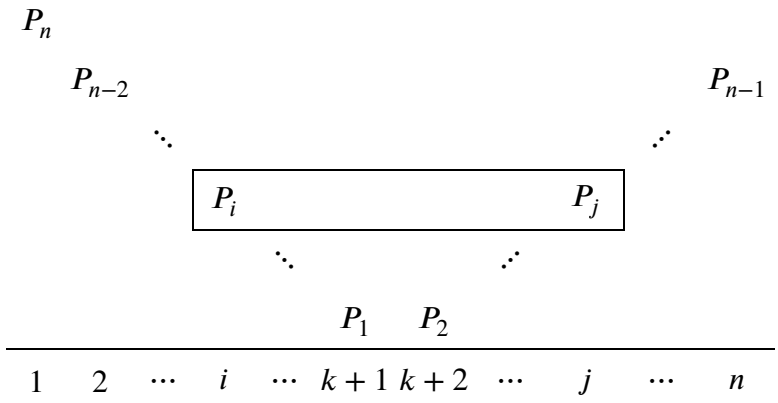


Figure 4.6 – Représentation de la séquence des tâches dans le cas n est impair.

4.2.3.3 L'heuristique H

L'heuristique que nous allons présenter ici, est constructive. Elle s'appuie sur deux points essentiels à savoir :

- Construction d'une séquence dans le sous-espace V-shape symétrique. Ceci peut se faire en appliquant l'heuristique EC1.1.
- Utilisation de la notion de la séquence duale. Ceci peut se faire en permutant alternativement, la tâche P_i avec la tâche P_j, $(i, j) \in E$.

Soit π la séquence obtenue en appliquant EC1.1. L'heuristique H génère une nouvelle séquence combinant 50% de π et 50% de π^d.

Algorithme 4.3 – L'heuristique H

étape (3) Générer une séquence V-shape en appliquant l'heuristique EC1.1

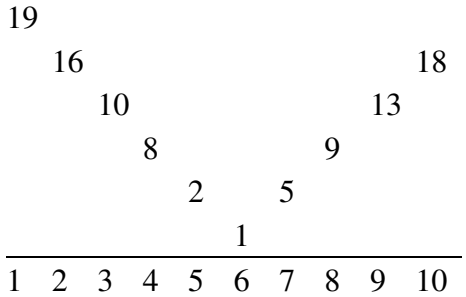
étape (4) Permuter alternativement, la tâche P_i avec la tâche P_j, $(i, j) \in E$.

4.2.4 Exemple illustratif

Nous reprenons l'exemple N°5 du tableau 4.1 : $P = (1\ 2\ 5\ 8\ 9\ 10\ 13\ 16\ 18\ 19)$, Le calcul de la séquence approchée en appliquant l'heuristique H se fait comme suit,

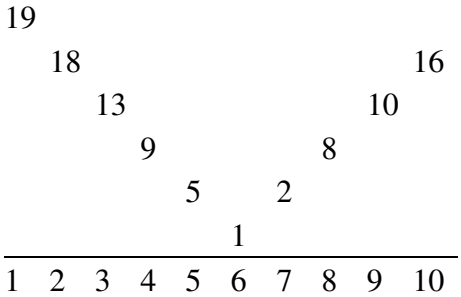
Soit π la séquence engendrée par EC1.1, et la séquence duale est π^d. L'heuristique H procure une nouvelle séquence qui combine 50% de π et 50% de π^d.

$\pi = (\mathbf{10}, 8\ 6\ 4\ 2\ 1\ 3\ 5\ 7\ 9)$
$\mathrm{CTV}(\pi) = 488.36$



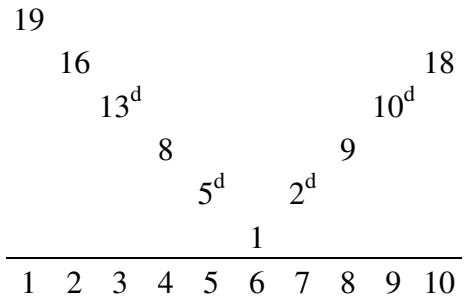
Séquence engendrée par EC1.1.

$\pi^d = (\mathbf{10},\ 9\ 7\ 5\ 3\ 1\ 2\ 4\ 6\ 8)$
$\mathrm{CTV}(\pi^d) = 488.36$



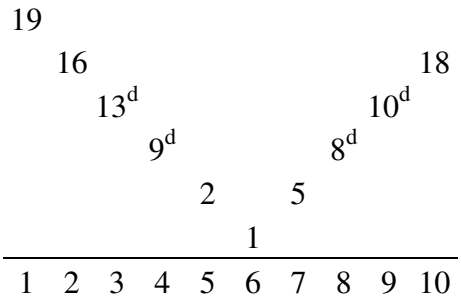
Séquence duale

$\pi^H = (\mathbf{10}, 8\ \mathbf{7}\ 4\ \mathbf{3}\ 1\ \mathbf{2}\ 5\ \mathbf{6}\ 9)$
$\mathrm{CTV}(\pi^H) = 486.56$



Séquence engendrée par H.

$\hat{\pi} = (\mathbf{10}, 8\ \mathbf{7}\ 5\ 2\ 1\ 3\ 4\ \mathbf{6}\ 9)$
$\mathrm{CTV}(\hat{\pi}) = 486.40$



Séquence exacte

4.3 Comparaison des résultats numériques

Dans cette section, nous exposons en premier temps, les exemples test du problème 1 || CTV, ensuite nous présentons des constations et commentaires sur les résultats numériques obtenus.

Les résultats numériques illustrés dans le tableau 4.1, le tableau 4.2, et le tableau 4.3 présentent la comparaison entre les trois valeurs de CTV, obtenues en appliquant l'heuristique EC1.2, notre algorithme H, et du modèle exacte pour ($n \leq 15$), en testons des données qui sont tirées de la littérature à savoir :

Tableau 4.1 : Eilon et Chowdhury en 1977.

Tableau 4.2 : Gupta et al. en 1990.

Tableau 4.3 : Srirangacharyulu and Srinivasan en 2010.

Durant notre étude, nous constatons les points suivant :

- Bien que la valeur de CTV($\overline{V}$) est toujours plus grande ou égale à CTV(V), il existe des cas, ou CTV($\overline{V}$) = CTV(V), notamment pour les exemples de grande taille de l'ordre ($n \geq 20$).
- La supériorité incontestable de l'heuristique H sur heuristique EC1.2 en matière des valeurs calculées (précision de la variance).
- Le modèle exacte n'arrive pas à calculer des séquences optimales dont la taille ($n \geq 20$). En revanche, avec les deux heuristiques EC1.2 et H on peut résoudre les exemples de taille $n \gg 20$, grâce à leur complexité polynômiale.

Tableau 4.1 – Résultats de comparaison des instances dans (Eilon et Chowdhury, 1977).

N°	*n*	Temps d'exécution	CTV			
			EC1.1	**EC1.2**	**H**	**Exact**
1	6	2 6 9 12 19 21	220.81	[a]218.47	[a]218.47	218.47
2	7	2 3 6 9 21 65 82	941.39	923.43	924.98	918.29
3	8	2 4 6 7 8 9 10 16	187.61	187.36	[a]187.23	187.23
4	8	1 2 8 9 10 12 13 16	255.36	254.86	254.48	254.23
5	10	1 2 5 8 9 10 13 16 18 19	488.36	487.24	486.56	486.40
6	10	2 3 6 9 21 23 34 65 82 92	3634.01	3593.21	[b]3584.09	3584.00
7	10	5 7 8 9 10 13 21 25 41 100	1353.64	1336.36	[a]1336.24	1336.24

[a] désigne que la valeur obtenue représente un minimum dans l'espace V-shape (minimum global).

[b] désigne que la valeur obtenue représente un minimum dans le sous-espace V-shape symétrique (minimum local).

Tableau 4.2 – Résultats de la comparaison des instances dans (Gupta et al., 1990).

N°	n	Temps d'exécution	CTV			
			EC1.1	EC1.2	H	Exact
8	10	6 12 17 22 33 40 52 61 70 88	7070.44	7047.76	[a]7027.96	7027.96
9	10	12 14 22 26 37 50 78 79 98 99	12400.80	12314.16	12271.04	12269.76
10	10	11 34 37 48 68 72 73 79 82 94	20918.56	20913.04	[b]20907.44	20903.36
11	10	10 34 39 41 46 48 52 70 76 96	14104.29	14097.81	[a]14094.01	14094.01
12	10	10 42 44 48 52 62 65 75 77 98	18888.56	18887.04	[a]18884.80	18884.80
13	15	3 11 14 20 24 25 26 29 32 39 47 80 83 90 98	16143.32	16113.57	16061.69	16052.20
14	15	6 16 20 24 27 35 41 45 59 63 64 72 77 91 97	27129.63	27103.00	[b]27083.69	27082.79
15	15	5 13 18 26 40 43 50 55 57 58 62 63 77 91 97	29827.05	29814.36	[b]29805.49	29805.39
16	15	7 15 30 42 61 64 68 73 79 82 83 92 95 97 98	57748.96	57745.05	57718.25	57713.76
17	15	7 16 26 32 40 42 51 52 59 67 77 82 84 85 96	36024.65	36023.13	[b]36005.36	36005.18
18	20	8 9 17 26 32 44 46 49 57 58 59 61 71 73 75 77 86 88 97 99	70906.40	70887.59	70856.34	/
19	20	2 9 13 21 30 36 43 47 56 70 74 77 75 77 83 87 88 90 92 97	75167.55	75164.33	75112.85	/
20	20	7 9 20 27 28 29 38 40 46 48 53 63 69 71 74 77 79 89 94 97	58966.27	58958.23	58916.05	/
21	20	2 4 14 22 25 35 44 46 47 55 56 60 63 66 69 72 73 74 77 82	54611.81	54608.84	54593.01	/
22	20	6 10 14 24 27 31 36 37 40 41 47 48 63 65 70 78 83 8992 94	50087.15	50080.19	50016.83	/

Tableau 4.3 – Résultats de la comparaison des instances dans (Srirangacharyulu and Srinivasan, 2010).

N°	*n*	Temps d'exécution	CTV			
			EC1.1	EC1.2	H	Exact
23	15	5 18 19 40 46 55 59 62 63 65 65 73 76 77 83	38987.63	38985.42	38924.86	38922.65
24	15	7 12 14 16 17 29 38 44 51 56 61 63 65 86 86	20153.71	20120.11	20112.73	20102.38
25	15	8 11 29 36 37 38 47 49 51 52 61 65 69 92 93	29238.49	29218.86	[b]29217.57	29217.09
26	15	7 20 21 21 33 40 41 48 53 67 83 87 92 96 99	32604.78	32595.29	[b]32558.86	32551.32
27	20	1 9 13 22 36 38 40 43 45 51 56 61 70 79 85 86 91 91 91 97	64399.43	64399.43	64344.33	/
28	20	1 8 12 12 13 19 38 40 45 52 55 60 65 71 80 81 93 95 96 97	51831.66	51828.79	51743.00	/
29	25	5 10 11 21 29 32 33 37 38 44 47 53 57 64 69 74 78 7986 86 88 88 91 92 95	107618.55	107617.33	107559.76	/
30	25	7 12 14 16 16 17 17 22 29 38 44 44 50 51 56 56 58 6163 65 74 85 86 86 94	67388.08	67388.08	67365.75	/
31	25	1 9 9 10 21 30 32 33 40 49 49 52 53 61 62 68 68 70 7480 81 81 85 98 100	91102.47	91080.88	91028.65	/

[a] désigne que la valeur obtenue représente un minimum dans l'espace V-shape (minimum global).

[b] désigne que la valeur obtenue représente un minimum dans le sous-espace V-shape symétrique (minimum local).

4.4 Conclusion

Dans ce chapitre, nous avons présenté notre heuristique, notée H, ainsi que son implémentation pour la résolution du problème de CTV. Nous avons reporté et comparé aussi des résultats numériques obtenus en testant notre programme MATLAB, sur des exemples du problème, tirés de la littérature. Durant notre étude, nous avons constaté que de nombreux tests numériques effectués, ont permis d'avérer une supériorité flagrante de l'heuristique H sur EC1.2 pour la résolution des exemples du problème. Les résultats numériques obtenus affirment l'efficacité de cette approche de résolution.

Conclusion

L'optimisation combinatoire occupe une place très importante dans la vie quotidienne. Son importance se justifie d'une part, par la grande difficulté des problèmes d'optimisation et d'autre part, par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes dans différents secteurs de l'industrie telle que la logistique, la gestion,…., etc.

Dans ce mémoire, nous nous sommes intéressés au problème de la minimisation de la variance des dates de fin d'exécution des tâches sur une seule machine. Ce problème est classé NP-difficile [23], il est formulé comme un programme quadratique en nombres binaires (0/1). Pour le résoudre, nous avons proposé une heuristique en élaborant un code de calcul en langage Matlab permettant de calculer une solution approchée en temps raisonnable.

Les résultats que nous avons obtenus, ont montré que notre heuristique s'est montrée numériquement plus compétitive que l'heuristique EC1.2 grâce à son efficacité et sa précision, notamment pour les exemples de grande taille. Ces résultats conduisent à envisager quelques perspectives s'articulant autour les points suivants :

- Implémenter une métaheuristique permettant de résoudre les exemples du problème de grande taille ($n \geq 20$), et de comparer les résultats obtenus avec ceux qui sont obtenus en résolvant le modèle mathématique.
- Expliquer, pourquoi dans le cas des exemples de grande taille ($n \geq 20$), L'heuristique H donne des séquences plus proches de l'optimum

Nous espérons que notre but est atteint, et le travail que nous avons réalisé dans ce mémoire, nous a permis d'approfondir et de raffiner nos connaissances théoriques et pratiques dans le domaine de l'ordonnancement d'atelier et d'acquérir des connaissances dans le domaine informatique.

Notations et abréviations

Symboles :

x	Vecteur de taille n : ($x_1, x_2, \dots, x_n$).
$x \geq 0$	Vecteur positif : $x \in \mathbb{R}_+^n$
$\sum_{i=1}^{n} x_i$	La somme : $x_1 + x_2 + \cdots + x_n$.
$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$	La somme double : $(x_{11} + \cdots + x_{1n}) + (x_{21} + \cdots + x_{2n}) + \cdots + (x_{m1} + \cdots + x_{mn})$
$\sum_{i=1}^{n} x_i\, y_i$	Le produit scalaire du vecteur x et y.
$\max z$; $\min z$	Maximiser la valeur z ; Minimiser la valeur z.

Abréviations :

PTAS	Schéma d'approximation polynomial
FPTAS	Schéma d'approximation totalement polynomial
SPT	Shortest Processing Time
LPT	Longest Processing Time
CTM	La moyenne des temps de fin d'exécution
CTV	La variance des temps de fin d'exécution
EC1.1	Heuristique
EC1.2	Heuristique
H	Heuristique proposée

Références

[1] Al-Turki, U., Fedjki, C., & Andijani, A. (2001). Tabu search for a class of single machine scheduling problems. Computer and Operations Research, 28, 1223-1230.

[2] Baker, K. R., & Trietsch, D. (2009). Principles of sequencing and sheduling. New Jersey: Wiley

[3] Baker, K. R., & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties. a review, Oper. Res, 38, 22-36.

[4] Brucker, P. (2006). Scheduling Algorithms. Germany: Springer.

[5] Brucker, P., & Knust, S. (2012). Complex Scheduling. Germany: Springer.

[6] Cai, X. (1993). A Solvable case of the variance minimisation problem. Applied Mathematics Lettres, 6 (6), 97-100.

[7] Cai, X. (1996). V-shape property for job sequences that minimize the expected completion time variance. European Journal of Operational Research, 91, 118-123.

[8] Chaudhry, I., & Drake, P. (2008). Minimizing flow-time variance in a single-machine system using genetic algorithms. International Journal of Advanced Manufacturing Technology, 39 (3-4), 355-366.

[9] Chen, Y., Farley, T., & Ye, N. (2003). QoS requirements of network applications on the Internet. Information, Knowledge, Systems Management, 4 (1).

[10] Cook, S. A. (1971). The complexity of theorem proving procedures. In Association of Computing Machinery (pp. 151–158). New York: Proceedings of the third annual ACM symposium on the theory of computing.

[11] Eilon, S., & Chowdhury, I. (1977). Minimizing waiting time variance in the single machine problem. Management Science, 23, 567-575.

[12] French, S. (1982). Scheduling : An Introduction to the Mathematics of the Job Shop. New York: Wiley.

[13] Gajpal, Y., & Rajendran, C. (2006). An Ant-Colony Optimization algorithm for minimizing the completion-time variance of jobs in flow shops. International Journal of Production Economics, 101, 259-272.

[14] Garey, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide of the theory of NP-completeness. San Fransisco: W.H. freeman and Company.

[15] Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. Annals of Discrete Mathematics, 5, 287-326.

[16] Gupta, M. C., Gupta, Y. P., & Bector, C. R. (1990). Minimizing the Flow-time Variance in Single-machine. Operational Research Society, 41 (8), 767-779.

[17] Gupta, M. C. (1993). Minimizing flow time variance in a single machine system using genetic algorithms. European Journal of Operational Research, 70, 289-303.

[18] Hall, N. G. (1986). Single and Multiple Processor Models for Minimizing Completion Time Variance. Naval Research Logistics Quarterly, 33.

[19] Hall, N. G., & Kubiak, W. (1991). Proof of a conjecture of Schrage about the completion time variance problem. Operations Research Letters, 10 (8), 467-472.

[20] Kanet, J. J. (1981). Minimizing variation of flow times in single machine systems. Science Management, 27 (12), 1453-1459.

[21] Kanet, J. J. (1981). Minimizing the average deviation of job completion times about a common due date. Naval Research Logistics Quarterly, 28, 643–651.

[22] Kubiak, W., Cheng, J., & Kovalyov, M. Y. (2002). Fast fully polynomial approximation schemes for minimizing completion time variance. European Journal of Operational Research, 137, 303-309.

[23] Kubiak, W. (1993). Completion time variance minimization on a single machine is difficult. Operations Research Letters, 14, 49-59.

[24] Kubiak, W. (1995). New results on the completion time variance minimization. Discrete Applied Mathematics , 58, 157-168.

[25] Lenstra, J., Rinnooy Kan, A., & Brucker, P. (1977). Complexity of machine scheduling problems (Vol. 1). Annals of Discrete Mathematics.

[26] Li, X., Ye, N., Liu, T., & Sun, Y. (2007). Job scheduling to minimize the weighted waiting time variance of jobs. Computers & Industrial Engineering, 52, 41-56.

[27] Li, X. (2010). Minimizing job completion time variance for service stability on identical parallel machines. Computers & Industrial Engineering, 58, 729-738.

[28] Mahapatra, S., Dash, R. R., & Pradhan, S. K. (2015). A Heuristic Job Scheduling Algorithm for Minimizing the Waiting Time Variance. 978-1-4799-8433-6/15/$31.00©2015 IEEE .

[29] Mehta, P., Pandit, P., Philip, D., & Sharma, P. (2012). An efficient local search for minimizing completion time variance in permutation flow shops. Computers & Operations Research, 39, 1000-1009.

[30] Merten, A. G., & Muller, M. E. (1972). Variance minimization in single machine sequencing problem. Management Science, 18 (9), 518-528.

[31] M'Hallah, R. (2007). Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. Computers and Operations Research (34), 3126-3142.

[32] Nessah, R., & Chengbin, C. (2010). A lower bound for weighted completion time variance. European Journal of Operational Research, 207, 1221-1226.

[33] Ng, C., Cai, X., & Cheng, T. C. (1996). A tight lower bound for the completion time variance problem. European Journal of Operational Research, 92, 211-213.

[34] Pinedo, M. (2008). Scheduling theory, algorithms and systems. New York: Springer.

[35] Pinedo, M. L. (2009). Planning and Scheduling in Manufacturing and Services. New York: Springer.

[36] Schrage, L. (1975). Minimizing the time in system variance for a finite job set. Management Science , 21 (5), 540-543.

[37] Srinivasan, G., & Srirangacharyulu, B. (2012). Minimising variance of job completion times in a single machine. Int. J. Operational Research, 13 (1), 110-127.

[38] Srirangacharyulu, B., & Srinivasan, G. (2010). Completion time variance minimization in single machine and multi-machine systems. Computers & Operations Research, 37, 62-71.

[39] Srirangacharyulu, B., & Srinivasann, G. (2013). An exact algorithm to minimize mean squared deviation of job completion times about a common due date. European Journal of Operational Research , 231, 547-556.

[40] Ventura, J. A., & Radhakrishnan, S. (2003). Single machine scheduling with symmetric earliness and tardiness penalties. European Journal of Operational Research (144), 598-612.

[41] Viswanathkumar, G., & Srinivasan, G. (2003). A branch and bound algorithm to minimize completion time variance on a single processor. Computers & Operations Research , 30 (8), 1135-1150.

[42] Xu, X. (2011). Minimizing weighted waiting time variance on a single processor. Computers & Industrial Engineering, 61, 1233-1239.

[43] Ye, N., Li, X., Farley, T., & Xu, X. (2007). Job scheduling methods for reducing waiting time variance. Computers & Operations Research, 34, 3069-3083.

[44] Zad, A. S., & Dashti, F. (2012). A New Method for Job Scheduling According to Minimize Variance and Mean Time. International Conference on Communication Systems and Network Technologies.