

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET
POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE

Université Mohamed Boudiaf de M'sila

Faculté de Mathématiques et de l'Informatique

Département de Mathématiques



N° d'ordre :

Thèse

Présentée pour l'obtention du diplôme
de Doctorat en sciences

Spécialité :

Mathématiques appliquées

Option :

Mathématiques appliquées

Présentée par :

Lounissi Nabila

Thème

**Efficacité et fiabilité des algorithmes génétiques
pour résoudre un problème d'ordonnancement
à machine unique**

Soutenue le 30/06/2026 devant le jury composé de :

SAADI Khalil	Prof.	Université de M'sila	Président
SELT Omar	Prof.	Université de Djelfa	Rapporteur
HEMMEK Alloua	Prof.	Université de M'sila	Co-encadreur
GAGUI Bachir	MCA	Université de M'sila	Examineur
BEHLOULI Djilali	Prof.	Université USTHB Alger	Examineur
MIHOUBI Milloud	Prof.	Université USTHB Alger	Examineur

Année universitaire : 2025/2026

Dédicaces

À la mémoire de ma chère mère, qu'Allah lui accorde Sa miséricorde. Elle a quitté ce monde, mais elle demeure vivante dans mon cœur. Son souvenir et ses prières continuent d'éclairer chacun de mes pas. Puisse Allah l'accueillir dans Son vaste paradis et faire de ce travail une œuvre qui honore sa mémoire.

À mon cher père, mon soutien et ma source de force, pour son amour, ses encouragements et sa présence précieuse.

À mon cher époux, compagnon de mon chemin, pour sa patience, sa compréhension et son soutien constant tout au long de ce parcours.

À mes chers enfants, la lumière de ma vie et la raison de ma persévérance. Je vous dédie le fruit de cet effort, en espérant qu'il soit pour vous un exemple de patience, de volonté et de réussite.

À toutes les personnes qui m'ont soutenue, encouragée ou accompagnée par une parole bienveillante ou une prière sincère.

Je dédie humblement ce travail.

Remerciements

Avant tout, je remercie Allah, le Tout-Puissant, de m'avoir donné la force, la patience et la volonté nécessaires pour mener à bien ce travail.

Je tiens également à exprimer mes sincères remerciements aux membres du jury, qui m'ont fait l'honneur d'accepter de lire, d'évaluer et de discuter ce travail.

J'adresse mes vifs remerciements au Professeur Saadi Khalil, pour avoir accepté de présider le jury de soutenance.

Je remercie également le Professeur Selt Omar, en sa qualité de rapporteur, pour l'intérêt porté à ce travail, ainsi que le Professeur Hemmak Alloua, en sa qualité de co-encadreur, pour son accompagnement, ses conseils et ses orientations.

Mes remerciements s'adressent aussi aux Professeurs Gagui Bachir, Mihoubi Milloud et Behlouli Djilali, pour avoir accepté de faire partie du jury et pour leurs remarques scientifiques qui contribueront à enrichir ce travail.

Enfin, j'exprime toute ma reconnaissance à mon cher père, à mon époux et à mes enfants pour leur soutien, leur patience et leurs encouragements constants.

Mes pensées vont également à la mémoire de ma chère mère, qu'Allah lui accorde Sa miséricorde.

Liste des notations et symboles utilisés

Dans cette thèse, les principales notations et symboles utilisés sont définis comme suit :

Symbole	Signification
$J = \{1, \dots, n\}$	Ensemble des tâches à ordonnancer
n	Nombre total de tâches
p_i	Durée de traitement de la tâche i
w_i	Poids associé à la tâche i
S_i	Date de début d'exécution de la tâche i
C_i	Date d'achèvement de la tâche i
x_{ij}	Variable binaire de précédence entre les tâches i et j
y_{ik}	Variable binaire liée aux périodes d'indisponibilité
M	Constante suffisamment grande du modèle PLNE
GA	Algorithme génétique
TS	Recherche Tabou
T	Liste Tabou

TAB. 1 : Liste des notations et symboles utilisés

Table des matières

Dédicaces	i
Remerciements	ii
Liste des notations et symboles utilisés	iii
1 Fondements théoriques des problèmes d’ordonnancement	5
1.1 Introduction	5
1.2 Concepts fondamentaux de l’ordonnancement	7
1.2.1 Définition d’une tâche	7
1.2.2 Définition de l’ordonnancement	8
1.2.3 Différence entre séquencement et ordonnancement	9
1.2.4 Représentation mathématique d’un problème d’ordonnancement	10
1.2.5 Synthèse des principales notions	10
1.3 Contraintes dans les problèmes d’ordonnancement	11
1.3.1 Contraintes de capacité des ressources	12
1.3.2 Contraintes de précédence	12
1.3.3 Contraintes de non-préemption	12
1.3.4 Contraintes liées aux périodes d’indisponibilité des machines	13
1.3.5 Impact des périodes d’indisponibilité sur la complexité	14
1.4 Classification des problèmes d’ordonnancement	14
1.4.1 Classification selon l’environnement des machines	15
1.4.1.1 Machine unique	15
1.4.1.2 Machines parallèles	16
1.4.1.3 Flow Shop	16
1.4.1.4 Job Shop	16
1.4.1.5 Open Shop	16
1.4.2 Classification selon les contraintes spécifiques du problème	17
1.4.3 Classification selon les critères d’optimisation	18
1.4.3.1 Minimisation du makespan	18
1.4.3.2 Minimisation de la somme des dates d’achèvement	18

1.4.3.3	Minimisation de la somme pondérée des dates d'achèvement	18
1.4.4	Positionnement du problème étudié	18
1.5	Ordonnancement avec périodes d'indisponibilité des machines	19
1.5.1	Classification des périodes d'indisponibilité	19
1.5.1.1	Indisponibilités connues à l'avance	19
1.5.1.2	Indisponibilités imprévues	20
1.5.2	Effet des indisponibilités sur les problèmes classiques	20
1.5.3	Approches exactes pour les problèmes avec indisponibilité	21
1.5.4	Approches heuristiques et métaheuristiques	21
1.6	État de l'art des méthodes de résolution des problèmes d'ordonnancement	21
1.6.1	Travaux fondamentaux sur l'ordonnancement classique	22
1.6.2	Complexité et limites des méthodes exactes	22
1.6.3	Développement des approches heuristiques	23
1.6.4	Développement des métaheuristiques	23
1.6.4.1	Algorithmes génétiques	23
1.6.4.2	Recherche Tabou	24
1.6.5	Synthèse des travaux antérieurs	24
1.6.6	Synthèse des lacunes scientifiques	25
1.7	Positionnement par rapport aux travaux récents	25
1.7.1	Évolution des méthodes exactes	26
1.7.2	Développement des approches métaheuristiques récentes	26
1.7.3	Travaux récents sur les contraintes de disponibilité	27
1.7.4	Positionnement scientifique de la thèse	28
1.8	Conclusion du chapitre	28
2	Méthodes de résolution des problèmes d'ordonnancement	30
2.1	Introduction	30
2.2	Méthodes exactes de résolution	31
2.2.1	Énumération complète	33
2.2.2	Programmation dynamique	34
2.2.3	Méthode Branch-and-Bound	35
2.2.4	Programmation mathématique	36
2.3	Méthodes approchées et heuristiques	38
2.3.1	Méthodes heuristiques	38
2.4	Métaheuristiques	40
2.4.1	Algorithmes génétiques	41
2.4.2	Recherche Tabou	43
2.4.3	Recuit simulé	44
2.4.4	Approches hybrides et méthodes combinées	45

2.5	Comparaison des méthodes de résolution	46
2.6	Analyse critique des travaux antérieurs	46
2.7	Conclusion	49
3	Modélisation mathématique exacte et résolution du problème d'ordonnancement par des approches métaheuristiques	51
3.1	Introduction	51
3.2	Modélisation mathématique exacte du problème	52
3.2.1	Notations et paramètres	52
3.2.2	Variables de décision	53
3.2.3	Variables binaires de séquençement	53
3.2.4	Fonction objectif	53
3.2.5	Contraintes du modèle	54
3.2.6	Contraintes du modèle	54
3.2.6.1	Contrainte de précédence entre les tâches	54
3.2.6.2	Contraintes de non-chevauchement des tâches	54
3.2.6.3	Contraintes liées aux périodes d'indisponibilité	55
3.2.6.4	Contraintes de domaine des variables	56
3.2.7	Formulation finale du modèle PLNE	56
3.3	Génération de la solution initiale par la règle WSPT	57
3.4	Algorithme génétique adapté au problème étudié	58
3.4.1	Représentation des solutions (chromosome)	58
3.4.2	Initialisation de la population	59
3.4.3	Fonction d'évaluation (fitness)	59
3.4.4	Opérateur de sélection	60
3.4.5	Opérateur de croisement	60
3.4.6	Opérateur de mutation	61
3.4.7	Critère d'arrêt	61
3.4.8	Pseudo-code de l'algorithme génétique	62
3.5	Recherche Tabou adaptée au problème étudié	62
3.5.1	Représentation d'une solution	63
3.5.2	Génération de la solution initiale	63
3.5.3	Structure de voisinage	64
3.5.4	Exploration du voisinage et choix du mouvement	64
3.5.5	Liste Tabou et critère d'aspiration	65
3.5.6	Critère d'arrêt	65
3.5.7	Pseudo-code de la recherche Tabou	66
3.5.8	Validation sur une instance illustrative de 20 tâches	66
3.5.8.1	Résultat obtenu par la Recherche Tabou	67

3.5.8.2	Résultat obtenu par l'Algorithme Génétique	68
3.5.9	Analyse des résultats obtenus	68
3.5.9.1	Qualité des solutions obtenues	68
3.5.9.2	Temps de calcul	70
3.6	Conclusion du chapitre	71
Conclusion générale		73
Résumé		78
Abstract		79
80		المخلص

Table des figures

3.1	Amélioration des solutions obtenues par rapport à la solution WSPT	69
3.2	Évolution du temps de calcul en fonction de la taille des instances	70

Liste des tableaux

1	Liste des notations et symboles utilisés	iii
1.1	Principales notions d'un problème d'ordonnancement	11
1.2	Synthèse des principaux travaux liés à l'ordonnancement	24
2.1	Synthèse critique de travaux et références représentatifs	48
3.1	Caractéristiques de l'instance de 20 tâches	67
3.2	Valeurs moyennes de la fonction objectif et amélioration obtenue par rapport à WSPT	69
3.3	Temps de calcul moyen des algorithmes TS et GA	70

Introduction générale

Les problèmes d’ordonnancement constituent l’un des domaines majeurs de la recherche opérationnelle et de l’optimisation combinatoire. Ils interviennent dans de nombreux systèmes industriels où plusieurs tâches doivent être exécutées sur des ressources limitées tout en respectant un ensemble de contraintes opérationnelles. L’objectif principal consiste à déterminer une organisation temporelle efficace des activités permettant d’améliorer les performances du système considéré [1–3].

De manière générale, un problème d’ordonnancement consiste à déterminer quand et dans quel ordre les tâches doivent être exécutées sur les ressources disponibles. La résolution de ces problèmes nécessite de prendre en compte plusieurs éléments tels que les durées de traitement des tâches, les relations de précédence, les capacités des ressources, les dates d’échéance ainsi que les différentes contraintes pouvant affecter le fonctionnement du système.

La diversité des applications industrielles explique la richesse de ce domaine de recherche. Les problèmes d’ordonnancement apparaissent notamment dans les systèmes de production manufacturière, les opérations de maintenance, les systèmes logistiques et les environnements industriels complexes. Dans ces contextes, une mauvaise organisation des activités peut entraîner une augmentation des temps d’attente, des coûts supplémentaires ou une diminution de l’efficacité globale du système.

Depuis les travaux fondamentaux de Graham et al. [1], la notation standard à trois champs $\alpha|\beta|\gamma$ est largement utilisée pour classer les problèmes d’ordonnancement. Le premier champ décrit l’environnement des machines, le deuxième précise les caractéristiques particulières du problème et le troisième indique le critère d’optimisation considéré. Cette classification constitue un cadre de référence permettant d’analyser la structure des problèmes étudiés ainsi que leur complexité algorithmique.

Malgré leur apparente simplicité, certains problèmes d’ordonnancement présentent une complexité combinatoire importante. Dans le cas d’une machine unique traitant n tâches indépendantes, le nombre de séquences possibles peut atteindre :

$$n!$$

Cette croissance factorielle de l’espace de recherche rend l’exploration exhaustive difficile lorsque la taille des instances augmente. De nombreux problèmes classiques d’ordonnancement

appartiennent ainsi à la classe des problèmes NP-difficiles, pour lesquels l'obtention d'une solution optimale en temps polynomial n'est pas connue à ce jour [4, 5].

Dans les environnements industriels réels, cette difficulté est accentuée par l'introduction de contraintes supplémentaires liées aux conditions d'exploitation des équipements. Parmi ces contraintes, les périodes d'indisponibilité des machines occupent une place importante. Elles peuvent correspondre à des opérations de maintenance préventive, des arrêts planifiés ou toute autre période durant laquelle une ressource ne peut pas être utilisée.

Les problèmes d'ordonnancement avec contraintes d'indisponibilité constituent ainsi une extension importante des modèles classiques. Dans ce contexte, la machine n'est plus considérée comme disponible durant tout l'horizon de planification, ce qui modifie les dates d'achèvement des tâches et réduit l'ensemble des ordonnancements réalisables [6, 7].

La présente thèse s'intéresse à un problème d'ordonnancement sur une machine unique soumise à des périodes déterministes d'indisponibilité connues à l'avance. Les tâches considérées sont indépendantes et non préemptives, ce qui signifie qu'une tâche commencée doit être exécutée entièrement sans interruption. Chaque tâche est caractérisée par une durée de traitement p_i et un poids associé w_i représentant son importance relative.

L'objectif étudié consiste à déterminer une séquence réalisable des tâches permettant de minimiser la somme pondérée des dates d'achèvement :

$$\min_{\sigma \in \Sigma} \sum_{i=1}^n w_i C_i(\sigma)$$

où Σ représente l'ensemble des ordonnancements admissibles, σ un ordonnancement donné, w_i le poids associé à la tâche i et $C_i(\sigma)$ sa date d'achèvement dans l'ordonnancement considéré.

Ce critère, appelé *Weighted Completion Time*, permet d'accorder une importance plus élevée aux tâches associées à des poids importants. Dans le problème classique à une machine sans contrainte d'indisponibilité, la règle WSPT (*Weighted Shortest Processing Time*) proposée par Smith [8] fournit une solution optimale en ordonnant les tâches selon un rapport poids/durée approprié. Cependant, cette propriété n'est plus garantie lorsque des périodes d'indisponibilité de la machine sont introduites. Dans cette thèse, le problème étudié peut être positionné selon la notation $\alpha|\beta|\gamma$ de Graham et al. [1] sous la forme :

$$1|h_k| \sum_{i=1}^n w_i C_i$$

où 1 représente une machine unique, h_k désigne la présence de k périodes déterministes d'indisponibilité connues à l'avance, et $\sum_{i=1}^n w_i C_i$ correspond au critère de minimisation de la somme pondérée des dates d'achèvement.

Cette formulation met en évidence la difficulté supplémentaire introduite par les périodes d'indisponibilité, puisque les décisions d'ordonnancement doivent simultanément respecter l'ordre des tâches et les contraintes temporelles imposées par la disponibilité de la machine. Compte

tenu de cette complexité, l'utilisation de méthodes exactes devient rapidement limitée lorsque la taille des instances augmente. Bien que ces méthodes permettent de garantir l'optimalité des solutions obtenues, leur coût computationnel peut devenir prohibitif pour les problèmes de grande dimension.

Dans ce contexte, les méthodes approchées, et particulièrement les métaheuristiques, constituent une alternative intéressante. Elles permettent d'explorer efficacement l'espace des solutions et d'obtenir des solutions de bonne qualité dans des temps de calcul raisonnables.

Parmi ces approches, l'algorithme génétique et la recherche Tabou ont montré leur efficacité dans plusieurs problèmes d'optimisation combinatoire, notamment dans le domaine de l'ordonnancement. L'algorithme génétique repose sur une exploration globale fondée sur une population de solutions, tandis que la recherche Tabou exploite une recherche locale guidée par une mémoire adaptative [9–11]. Compte tenu des difficultés liées à la résolution de ce type de problème, notamment en présence de contraintes d'indisponibilité des machines, le recours à des méthodes capables d'explorer efficacement l'espace des solutions devient nécessaire. Les méthodes exactes permettent de garantir l'optimalité des solutions obtenues lorsque la résolution est possible, mais leur complexité augmente rapidement avec la taille des instances.

Dans ce contexte, les méthodes métaheuristiques représentent une approche adaptée pour traiter les problèmes d'ordonnancement de grande dimension. Elles ne garantissent pas nécessairement l'obtention de la solution optimale, mais elles permettent d'obtenir des solutions de bonne qualité dans des temps de calcul raisonnables.

Parmi les métaheuristiques les plus utilisées en optimisation combinatoire, l'algorithme génétique et la recherche Tabou présentent des mécanismes de recherche complémentaires. L'algorithme génétique exploite une population de solutions afin d'assurer une exploration globale de l'espace de recherche, tandis que la recherche Tabou repose sur une exploration locale guidée par une mémoire adaptative permettant d'éviter les cycles et d'améliorer progressivement les solutions obtenues [9–11].

Ainsi, la problématique centrale de cette thèse peut être formulée comme suit :

Dans quelle mesure l'algorithme génétique et la recherche Tabou permettent-ils de résoudre efficacement un problème d'ordonnancement sur une machine unique soumise à des périodes déterministes d'indisponibilité, et quelle approche offre le meilleur compromis entre la qualité des solutions obtenues et le temps de calcul nécessaire ?

Afin de répondre à cette problématique, cette thèse poursuit plusieurs objectifs principaux :

- présenter les fondements théoriques des problèmes d'ordonnancement, leurs classifications, leurs caractéristiques et les notions nécessaires à leur modélisation ;
- analyser les principales méthodes de résolution existantes en distinguant les méthodes exactes, les approches heuristiques et les métaheuristiques ;

- formaliser le problème d’ordonnancement étudié avec contraintes d’indisponibilité de la machine ;
- développer et adapter deux approches métaheuristiques, à savoir l’algorithme génétique et la recherche Tabou, pour résoudre le problème considéré ;
- comparer expérimentalement les performances des deux méthodes selon la qualité des solutions obtenues et le temps de calcul.

Cette thèse est organisée en trois chapitres complémentaires permettant de présenter progressivement les fondements théoriques, les méthodes de résolution et l’approche proposée pour le problème étudié.

Le premier chapitre, intitulé *Fondements théoriques des problèmes d’ordonnancement*, est consacré à la présentation des concepts fondamentaux relatifs aux problèmes d’ordonnancement. Il introduit les notions de tâche, de machine, de ressource, de contrainte et de critère d’optimisation. Ce chapitre présente également les principales configurations d’ordonnancement ainsi que la classification des problèmes selon la notation $\alpha|\beta|\gamma$ de Graham et al. Il analyse ensuite la complexité algorithmique des problèmes étudiés et introduit plus particulièrement les problèmes d’ordonnancement avec périodes d’indisponibilité des machines.

Le deuxième chapitre, intitulé *Méthodes de résolution des problèmes d’ordonnancement*, présente les principales approches utilisées pour résoudre les problèmes d’optimisation combinatoire. Il expose les méthodes exactes, notamment l’énumération complète, la programmation dynamique, la méthode Branch-and-Bound et la programmation mathématique. Il présente ensuite les méthodes approchées, en distinguant les heuristiques, les métaheuristiques et les approches hybrides. Une analyse comparative de ces méthodes permet de justifier le choix des approches retenues dans cette thèse.

Le troisième chapitre, intitulé « Modélisation mathématique exacte et résolution du problème d’ordonnancement par des approches métaheuristiques », est consacré à la présentation de l’approche proposée.

Enfin, une conclusion générale synthétise les principaux résultats obtenus, présente les contributions de cette recherche et propose quelques perspectives pour de futurs travaux.

Chapitre 1

Fondements théoriques des problèmes d'ordonnancement

1.1 Introduction

Les problèmes d'ordonnancement constituent l'un des domaines majeurs de la recherche en optimisation combinatoire et en recherche opérationnelle. Ils interviennent dans de nombreux systèmes réels tels que les ateliers de production, les systèmes manufacturiers, les réseaux informatiques, les transports ainsi que les environnements industriels complexes.

De manière générale, un problème d'ordonnancement consiste à déterminer l'affectation temporelle d'un ensemble de tâches sur un ensemble de ressources tout en respectant différentes contraintes opérationnelles et en optimisant un ou plusieurs critères de performance. La difficulté principale réside dans la nécessité de prendre simultanément en considération l'ordre d'exécution des tâches, la disponibilité limitée des ressources et les objectifs économiques associés au fonctionnement du système [1–3].

Dans les environnements industriels modernes, les modèles classiques d'ordonnancement ne permettent pas toujours de représenter fidèlement les conditions réelles d'exploitation. En effet, les équipements peuvent être soumis à différentes contraintes telles que les opérations de maintenance préventive, les arrêts planifiés, les périodes de panne ou les limitations de disponibilité des ressources. Ces contraintes introduisent des interruptions dans le fonctionnement normal des machines et rendent la prise de décision plus complexe.

Parmi ces extensions, les problèmes d'ordonnancement avec périodes d'indisponibilité des machines constituent une classe particulière de problèmes où la disponibilité des ressources n'est plus considérée comme permanente. Durant ces périodes, aucune tâche ne peut être exécutée sur la machine concernée. Lorsque les tâches sont non préemptives, une tâche commencée doit être exécutée entièrement sans interruption. Elle doit donc être terminée avant le début d'une période d'indisponibilité ou commencer après la remise en disponibilité de la machine.

Le problème étudié dans cette thèse concerne l'ordonnancement de tâches indépendantes

et non préemptives sur une machine unique soumise à des périodes d'indisponibilité connues à l'avance. Chaque tâche est caractérisée par une durée de traitement p_i et un poids associé w_i . L'objectif consiste à déterminer une séquence réalisable permettant de minimiser la somme pondérée des dates d'achèvement :

$$\min \sum_{i=1}^n w_i C_i$$

où C_i représente la date d'achèvement de la tâche i et w_i son poids associé.

Ce critère d'optimisation, connu sous le nom de *Weighted Completion Time*, permet d'accorder une importance plus élevée aux tâches possédant des poids importants. Il est largement étudié dans la littérature de l'ordonnancement, notamment depuis les travaux fondamentaux de Smith [8], qui ont montré l'efficacité de la règle WSPT pour le problème classique à une machine sans contraintes supplémentaires.

Cependant, l'introduction des périodes d'indisponibilité modifie fortement la structure du problème. La règle WSPT, bien qu'optimale dans le problème classique :

$$1 || \sum_{i=1}^n w_i C_i$$

ne garantit plus l'optimalité lorsque la machine présente des périodes d'arrêt planifiées. Le problème devient alors plus difficile en raison de la modification des dates d'achèvement et de la réduction de l'ensemble des séquences réalisables.

Selon la notation à trois champs $\alpha|\beta|\gamma$ proposée par Graham et al. [1], le problème étudié appartient à la classe des problèmes d'ordonnancement sur une machine unique intégrant des contraintes d'indisponibilité de la machine.

Dans cette thèse, une représentation adaptée de la notation $\alpha|\beta|\gamma$ est utilisée afin d'expliciter la présence de périodes déterministes d'indisponibilité de la machine connues à l'avance. Le problème étudié est ainsi représenté par :

$$1|h_k| \sum_{i=1}^n w_i C_i$$

où :

- 1 représente un environnement constitué d'une machine unique ;
- h_k désigne k périodes d'indisponibilité de la machine connues à l'avance ;
- $\sum_{i=1}^n w_i C_i$ représente le critère de minimisation de la somme pondérée des dates d'achèvement.

Cette formulation permet de distinguer le problème considéré du problème classique :

$$1 || \sum_{i=1}^n w_i C_i$$

dans lequel la machine est supposée disponible durant tout l’horizon de planification. L’introduction de périodes d’indisponibilité modifie la structure temporelle du problème, car les tâches non préemptives ne peuvent pas chevaucher les intervalles durant lesquels la machine est indisponible [6, 7].

D’un point de vue algorithmique, les problèmes d’ordonnancement appartiennent à une classe de problèmes d’optimisation combinatoire dont la résolution exacte devient rapidement difficile lorsque la taille des instances augmente.

Pour un ensemble de n tâches indépendantes exécutées sur une machine unique, le nombre de séquences possibles correspond au nombre de permutations :

$$n!$$

Cette croissance factorielle de l’espace de recherche rend l’exploration exhaustive rapidement impraticable pour les instances de grande taille [4, 5].

Ainsi, même si les méthodes exactes telles que la programmation linéaire en nombres entiers, la programmation dynamique ou les méthodes de séparation et évaluation (*Branch and Bound*) permettent d’obtenir des solutions optimales, leur utilisation reste limitée aux instances de taille réduite ou moyenne.

Face à ces limitations, les méthodes heuristiques et métaheuristiques se sont imposées comme des outils efficaces permettant d’obtenir des solutions de haute qualité dans des temps de calcul raisonnables [11, 12].

Dans ce travail, deux approches métaheuristiques complémentaires sont étudiées : l’algorithme génétique (*Genetic Algorithm, GA*) et la recherche Tabou (*Tabu Search, TS*). La première repose sur une exploration globale basée sur une population de solutions, tandis que la seconde utilise une recherche locale guidée par une mémoire adaptative [9, 10].

L’objectif de ce chapitre est de présenter les fondements théoriques liés aux problèmes d’ordonnancement, de définir les concepts nécessaires à la compréhension du problème étudié, d’analyser sa complexité et de présenter les principales approches de résolution existantes. Cette analyse permettra de justifier le choix des méthodes développées dans les chapitres suivants.

1.2 Concepts fondamentaux de l’ordonnancement

1.2.1 Définition d’une tâche

Dans un problème d’ordonnancement, une tâche représente une opération élémentaire qui doit être exécutée par une ressource donnée pendant une durée déterminée. Elle constitue l’unité

fondamentale de décision autour de laquelle s'organise l'ensemble du processus d'ordonnement.

Selon Brucker [2], chaque tâche peut être caractérisée par un ensemble de paramètres permettant de définir ses propriétés temporelles et son importance dans le système étudié. Les paramètres les plus couramment utilisés sont la durée de traitement, la date de disponibilité, la date limite et le poids associé à la tâche.

Dans le cadre de cette thèse, les tâches considérées sont indépendantes et non préemptives. L'indépendance signifie qu'aucune relation de précédence n'existe entre les différentes tâches, ce qui permet de déterminer librement leur ordre d'exécution. La non-préemption signifie qu'une tâche, une fois son exécution commencée, doit être poursuivie jusqu'à son achèvement sans interruption. Dans le contexte étudié, elle doit en outre être exécutée entièrement en dehors des périodes d'indisponibilité de la machine.

Dans le cas particulier étudié dans cette thèse, chaque tâche J_i est définie principalement par les paramètres suivants :

$$J_i = (p_i, w_i)$$

où :

- p_i représente la durée de traitement de la tâche i ;
- w_i représente le poids ou l'importance associée à cette tâche.

À ces paramètres s'ajoutent les variables temporelles associées à l'ordonnement :

- S_i : date de début d'exécution de la tâche i ;
- C_i : date d'achèvement de la tâche i .

La relation fondamentale entre ces variables est donnée par :

$$C_i = S_i + p_i$$

Cette relation exprime que la date d'achèvement dépend directement de la date de début et de la durée nécessaire à l'exécution complète de la tâche.

1.2.2 Définition de l'ordonnement

Un ordonnancement représente une affectation temporelle des tâches aux ressources disponibles tout en respectant l'ensemble des contraintes imposées par le système.

Mathématiquement, un ordonnancement peut être considéré comme une fonction qui associe à chaque tâche une position dans une séquence ainsi qu'un intervalle d'exécution sur une ressource donnée.

Pour un ensemble de tâches :

$$J = \{J_1, J_2, \dots, J_n\}$$

un ordonnancement consiste à déterminer une permutation :

$$X = (J_{\pi(1)}, J_{\pi(2)}, \dots, J_{\pi(n)})$$

où π représente l'ordre d'exécution choisi.

Un ordonnancement est dit réalisable lorsque toutes les contraintes du problème sont satisfaites :

- chaque tâche est exécutée exactement une fois ;
- aucune tâche ne chevauche une autre sur une même ressource ;
- les contraintes temporelles sont respectées ;
- les périodes d'indisponibilité éventuelles sont prises en compte.

Dans le cas étudié, la ressource est une machine unique. Ainsi, un ordonnancement réalisable correspond directement à une permutation des tâches sur cette machine.

1.2.3 Différence entre séquençement et ordonnancement

Bien que les deux notions soient souvent utilisées de manière similaire, le séquençement et l'ordonnancement correspondent à deux niveaux différents de décision dans un problème d'optimisation.

Le séquençement consiste uniquement à déterminer l'ordre relatif d'exécution des tâches. Par exemple :

$$(J_3, J_1, J_5, J_2)$$

indique que la tâche J_3 doit être exécutée avant J_1 , puis J_5 et enfin J_2 .

Cependant, cette représentation ne fournit aucune information concernant les dates réelles d'exécution des tâches.

L'ordonnancement complète le séquençement en associant à chaque tâche une date de début et une date d'achèvement. Il permet ainsi de construire un calendrier d'exécution complet tout en respectant les contraintes du problème.

Ainsi :

Séquençement $\rightarrow$ détermination de l'ordre des tâches

Ordonnancement $\rightarrow$ ordre des tâches + calendrier temporel

Cette distinction est particulièrement importante dans les problèmes d'ordonnancement avec périodes d'indisponibilité des machines, car une même séquence de tâches peut conduire à différents calendriers d'exécution selon la position des périodes d'arrêt.

1.2.4 Représentation mathématique d'un problème d'ordonnancement

Un problème d'ordonnancement classique peut être représenté par un ensemble de tâches :

$$J = \{J_1, J_2, \dots, J_n\}$$

exécutées sur un ensemble de ressources :

$$M = \{M_1, M_2, \dots, M_m\}$$

Chaque tâche possède une durée de traitement :

$$p_i$$

et nécessite la détermination des éléments suivants :

- son ordre d'exécution ;
- sa date de début ;
- sa date d'achèvement.

Dans le problème étudié dans cette thèse, le nombre de machines est égal à un :

$$m = 1$$

L'objectif consiste alors à déterminer une permutation réalisable des tâches minimisant la fonction objectif :

$$\min Z = \sum_{i=1}^n w_i C_i$$

sous les contraintes liées au séquençement des tâches ainsi qu'aux périodes d'indisponibilité de la machine.

1.2.5 Synthèse des principales notions

Le tableau suivant résume les principales notions utilisées dans cette thèse.

TAB. 1.1 : Principales notions d'un problème d'ordonnancement

Notion	Définition
Tâche J_i	Unité élémentaire de travail nécessitant une durée d'exécution déterminée.
Machine	Ressource utilisée pour réaliser les tâches.
Durée p_i	Temps nécessaire pour terminer la tâche i .
Poids w_i	Importance relative attribuée à la tâche dans la fonction objectif.
Séquence	Ordre relatif dans lequel les tâches doivent être exécutées.
Ordonnancement	Détermination simultanée de l'ordre des tâches et de leurs dates d'exécution.
Date d'achèvement C_i	Instant auquel la tâche i termine son traitement.
Indisponibilité	Intervalle durant lequel une machine ne peut pas exécuter de tâche.

1.3 Contraintes dans les problèmes d'ordonnancement

Les contraintes représentent l'ensemble des conditions qui limitent les possibilités d'exécution des tâches dans un système d'ordonnancement. Elles définissent l'espace des solutions réalisables et influencent directement la complexité du problème à résoudre.

Dans les systèmes industriels réels, les contraintes ne se limitent pas à la capacité des machines. Elles peuvent concerner les relations entre les tâches, les caractéristiques temporelles, la disponibilité des ressources ou encore les conditions particulières d'exploitation des équipements [2,3].

Les principales catégories de contraintes rencontrées dans les problèmes d'ordonnancement concernent notamment :

- les contraintes de capacité des ressources ;
- les contraintes de précédence entre les tâches ;
- les contraintes temporelles ;
- les contraintes de non-préemption ;
- les contraintes de disponibilité des machines.

Dans le problème étudié dans cette thèse, une attention particulière est accordée aux contraintes de capacité, de non-préemption et aux périodes d'indisponibilité de la machine.

1.3.1 Contraintes de capacité des ressources

La capacité d'une ressource définit le nombre maximal de tâches pouvant être traitées simultanément.

Dans le cas d'une machine unique, la capacité est limitée à une seule tâche à la fois. Par conséquent, deux tâches distinctes i et j ne peuvent pas être exécutées simultanément sur cette ressource.

Cette contrainte est généralement exprimée sous forme disjonctive :

$$(S_i + p_i \leq S_j) \quad \text{ou} \quad (S_j + p_j \leq S_i)$$

Elle signifie que deux configurations uniquement sont possibles :

- la tâche i est exécutée avant la tâche j ;
- la tâche j est exécutée avant la tâche i .

Cette contrainte constitue une caractéristique fondamentale des problèmes d'ordonnement sur machine unique [1, 3].

1.3.2 Contraintes de précédence

Les contraintes de précédence imposent un ordre obligatoire entre certaines tâches. Elles apparaissent lorsque l'exécution d'une opération dépend de l'achèvement préalable d'une autre tâche.

Pour deux tâches i et j , si la tâche i doit être réalisée avant la tâche j , alors :

$$S_j \geq C_i$$

Ces contraintes réduisent l'espace des solutions admissibles, car certaines permutations de tâches deviennent interdites.

Dans le problème étudié, les tâches sont supposées indépendantes. Par conséquent, aucune relation de précédence n'est imposée entre les différentes tâches.

1.3.3 Contraintes de non-préemption

Une contrainte importante dans les problèmes d'ordonnement industriels concerne la possibilité d'interrompre ou non l'exécution d'une tâche.

Dans un problème préemptif, une tâche peut être interrompue temporairement puis reprise ultérieurement. À l'inverse, dans un problème non préemptif, une tâche commencée doit être exécutée de manière continue jusqu'à son achèvement, sans interruption [2].

Ainsi, l'exécution d'une tâche i occupe un intervalle continu :

$$[S_i, C_i]$$

sans interruption intermédiaire.

Cette hypothèse est particulièrement adaptée aux applications industrielles où l'arrêt d'une opération en cours peut entraîner des pertes économiques, des réglages supplémentaires ou des contraintes techniques particulières.

Dans cette thèse, toutes les tâches considérées sont supposées non préemptives.

1.3.4 Contraintes liées aux périodes d'indisponibilité des machines

Dans les modèles classiques d'ordonnancement, les machines sont généralement supposées disponibles durant tout l'horizon de planification. Cette hypothèse simplificatrice ne permet cependant pas de représenter fidèlement les environnements industriels réels, où les équipements peuvent être soumis à des périodes temporaires d'indisponibilité.

Ces interruptions peuvent être dues notamment à :

- des opérations de maintenance préventive ;
- des arrêts techniques planifiés ;
- des inspections périodiques ;
- des interventions correctives.

Dans les modèles d'ordonnancement avec indisponibilité des machines, ces périodes sont représentées par des intervalles connus à l'avance :

$$[a_k, b_k]$$

où a_k représente l'instant de début de la k -ème période d'indisponibilité et b_k l'instant de remise en disponibilité de la machine.

Durant cet intervalle, aucune tâche ne peut être exécutée.

Pour une tâche non préemptive J_i , deux configurations uniquement sont possibles par rapport à une période d'indisponibilité :

Cas 1 : la tâche est exécutée entièrement avant l'arrêt

$$C_i \leq a_k$$

Cas 2 : la tâche est exécutée entièrement après la remise en service

$$S_i \geq b_k$$

Ainsi, une tâche ne peut jamais chevaucher une période durant laquelle la machine est indisponible. Cette contrainte modifie l'ensemble des séquences réalisables et augmente la difficulté de résolution du problème.

1.3.5 Impact des périodes d'indisponibilité sur la complexité

L'introduction des périodes d'indisponibilité modifie profondément la structure des problèmes classiques d'ordonnancement.

Dans le problème classique à une machine :

$$1 || \sum_{i=1}^n w_i C_i$$

la règle WSPT permet d'obtenir une solution optimale en ordonnant les tâches selon le ratio :

$$\frac{p_i}{w_i}$$

[8].

Cependant, lorsque des périodes d'indisponibilité sont introduites, le problème devient :

$$1|h_k| \sum_{i=1}^n w_i C_i$$

et la propriété d'optimalité de la règle WSPT n'est généralement plus garantie.

En effet, une séquence optimale pour le problème classique peut conduire, en présence d'arrêts machine, à placer certaines tâches importantes juste avant une période d'indisponibilité. Cette situation entraîne un déplacement de leurs dates d'achèvement et une augmentation de la fonction objectif.

Par conséquent, l'évaluation d'une solution nécessite désormais plusieurs étapes complémentaires :

- déterminer la séquence d'exécution des tâches ;
- calculer les dates réelles d'achèvement en tenant compte des arrêts de la machine ;
- vérifier la faisabilité de la solution par rapport aux périodes d'indisponibilité.

Cette complexité supplémentaire justifie le recours à des méthodes de recherche avancées capables d'explorer efficacement un espace de solutions fortement contraint.

1.4 Classification des problèmes d'ordonnancement

Les problèmes d'ordonnancement peuvent être classifiés selon plusieurs critères liés à la structure du système étudié, aux contraintes associées aux tâches et au critère d'optimisation re-

cherché. Cette classification permet de décrire précisément la nature d'un problème et d'orienter le choix des méthodes de résolution adaptées [1, 3].

La notation de Graham et al. [1], connue sous le nom de notation $\alpha|\beta|\gamma$, constitue une référence fondamentale pour la description des problèmes d'ordonnancement.

Elle permet de représenter un problème sous la forme :

$$\alpha|\beta|\gamma$$

où :

- α décrit l'environnement des machines ;
- β représente les contraintes particulières du problème ;
- γ indique le critère d'optimisation.

1.4.1 Classification selon l'environnement des machines

La notation classique des problèmes d'ordonnancement proposée par Graham et al. [1] permet de caractériser un problème selon trois champs principaux :

$$\alpha|\beta|\gamma$$

où le premier champ α définit l'environnement des machines, le deuxième champ β représente les contraintes particulières du problème et le troisième champ γ correspond au critère d'optimisation.

Dans cette section, l'analyse porte principalement sur le premier champ décrivant la configuration des ressources utilisées pour l'exécution des tâches.

1.4.1.1 Machine unique

Dans le cas d'une machine unique, toutes les tâches doivent être exécutées sur une seule ressource.

La notation correspondante est :

$$\alpha = 1$$

Ce modèle constitue l'une des configurations fondamentales de la théorie de l'ordonnancement. Malgré sa simplicité apparente, sa complexité augmente rapidement lorsque des contraintes supplémentaires sont introduites, telles que les périodes d'indisponibilité des machines, les temps de préparation ou les contraintes temporelles.

Le problème étudié dans cette thèse appartient à cette catégorie puisqu'il concerne l'ordonnancement de tâches indépendantes et non préemptives sur une machine unique.

1.4.1.2 Machines parallèles

Dans cette configuration, plusieurs machines peuvent exécuter les tâches d'un même ensemble.

On distingue généralement :

- les machines parallèles identiques, où toutes les machines possèdent les mêmes caractéristiques ;
- les machines parallèles uniformes, où les vitesses de traitement peuvent être différentes ;
- les machines parallèles non identiques, où chaque machine possède des caractéristiques propres.

L'objectif consiste alors à affecter les tâches aux différentes machines tout en optimisant le critère considéré.

1.4.1.3 Flow Shop

Dans un problème Flow Shop, toutes les tâches suivent le même parcours à travers plusieurs machines.

Chaque tâche doit respecter une séquence identique de passage :

$$M_1 \rightarrow M_2 \rightarrow \dots \rightarrow M_m$$

Ce modèle représente notamment les lignes de production industrielles où les produits passent successivement par plusieurs postes de travail.

1.4.1.4 Job Shop

Le problème Job Shop représente des environnements plus complexes dans lesquels chaque tâche possède son propre itinéraire entre les machines.

Chaque opération est associée à une machine spécifique et l'ordre d'exécution des opérations dépend de la tâche considérée.

Ces problèmes présentent une complexité élevée et constituent une classe importante des problèmes d'optimisation combinatoire [2, 3].

1.4.1.5 Open Shop

Dans un problème Open Shop, les tâches doivent être exécutées sur plusieurs machines, mais l'ordre de passage entre les machines n'est pas imposé à l'avance.

Cette flexibilité distingue l'Open Shop des modèles Flow Shop et Job Shop et offre un espace de solutions différent, mais conduit également à des problèmes d'optimisation complexes.

1.4.2 Classification selon les contraintes spécifiques du problème

Le deuxième champ β de la notation de Graham et al. [1] décrit les contraintes particulières associées au problème d'ordonnancement. Ces contraintes permettent de caractériser les propriétés spécifiques des tâches, des ressources ou de l'environnement d'exécution.

Les principales contraintes rencontrées dans les problèmes d'ordonnancement sont notamment :

- **Disponibilité des tâches** : certaines tâches peuvent être disponibles uniquement après une date donnée ;
- **Contraintes de précedence** : certaines tâches doivent être terminées avant le début d'autres tâches ;
- **Temps de préparation** : certaines opérations nécessitent un temps supplémentaire avant leur exécution ;
- **Préemption** : possibilité ou non d'interrompre une tâche durant son exécution ;
- **Indisponibilité des machines** : périodes pendant lesquelles une ressource ne peut pas être utilisée.

Dans le cadre de cette thèse, les caractéristiques spécifiques considérées sont les suivantes :

- une machine unique ;
- des tâches indépendantes ;
- des tâches non préemptives ;
- des périodes d'indisponibilité de la machine connues à l'avance.

Ainsi, dans l'adaptation de la notation $\alpha|\beta|\gamma$ adoptée dans cette thèse pour intégrer explicitement les périodes d'indisponibilité, le problème étudié peut être représenté par :

$$1|h_k|\sum_{i=1}^n w_i C_i$$

où :

- 1 représente un environnement constitué d'une machine unique ;
- h_k représente la présence de k périodes d'indisponibilité connues à l'avance ;
- $\sum_{i=1}^n w_i C_i$ représente le critère de minimisation de la somme pondérée des dates d'achèvement.

1.4.3 Classification selon les critères d'optimisation

Le troisième champ γ représente l'objectif recherché.

Les critères les plus couramment utilisés sont :

1.4.3.1 Minimisation du makespan

Le makespan correspond à la date d'achèvement de la dernière tâche :

$$C_{\max} = \max(C_1, C_2, \dots, C_n)$$

Ce critère vise à réduire la durée totale nécessaire pour terminer toutes les tâches.

1.4.3.2 Minimisation de la somme des dates d'achèvement

Ce critère est défini par :

$$\sum_{i=1}^n C_i$$

Il favorise l'exécution rapide de l'ensemble des tâches.

1.4.3.3 Minimisation de la somme pondérée des dates d'achèvement

Le critère utilisé dans cette thèse est :

$$\min \sum_{i=1}^n w_i C_i$$

Il permet de prendre en compte l'importance relative des tâches. Une tâche ayant un poids élevé aura une influence plus importante sur la fonction objectif.

Ce critère est particulièrement adapté aux situations industrielles où certaines opérations sont prioritaires par rapport à d'autres [3, 8].

1.4.4 Positionnement du problème étudié

À partir de cette classification, le problème traité dans cette thèse peut être positionné comme suit :

$$\boxed{1|h_k| \sum_{i=1}^n w_i C_i}$$

Il correspond donc à un problème d'ordonnancement sur une machine unique avec des périodes d'indisponibilité connues, où l'objectif est de minimiser la somme pondérée des dates d'achèvement.

Cette formulation permet de distinguer clairement le problème étudié des modèles classiques d'ordonnancement et justifie la nécessité de développer des méthodes de résolution adaptées prenant en compte les contraintes de disponibilité des équipements.

1.5 Ordonnancement avec périodes d'indisponibilité des machines

Dans les modèles classiques d'ordonnancement, les machines sont généralement supposées disponibles durant tout l'horizon de planification. Cette hypothèse simplificatrice permet de réduire la complexité du problème mais ne reflète pas toujours les conditions réelles rencontrées dans les environnements industriels.

Dans les systèmes de production modernes, les équipements peuvent être temporairement indisponibles en raison des opérations de maintenance préventive, des arrêts techniques programmés ou des défaillances des systèmes. La prise en compte de ces périodes d'indisponibilité constitue donc une extension importante des modèles classiques d'ordonnancement [3, 13].

Un problème d'ordonnancement avec indisponibilité machine consiste à déterminer une séquence de tâches réalisable tout en intégrant des intervalles pendant lesquels certaines ressources ne peuvent pas être utilisées.

Ces périodes sont généralement représentées par :

$$[a_k, b_k]$$

où a_k représente le début et b_k la fin de la k -ème période d'indisponibilité.

Durant cet intervalle :

$$[a_k, b_k]$$

la machine est considérée comme indisponible et aucune opération ne peut être exécutée.

1.5.1 Classification des périodes d'indisponibilité

Les travaux de la littérature distinguent généralement plusieurs types d'indisponibilité des machines selon le niveau d'information disponible.

1.5.1.1 Indisponibilités connues à l'avance

Dans ce cas, les périodes d'arrêt sont déterminées avant la construction de l'ordonnancement. Elles correspondent principalement aux :

- opérations de maintenance préventive ;

- arrêts planifiés ;
- inspections périodiques.

Le problème consiste alors à construire un calendrier d'exécution évitant ces intervalles. Cette catégorie correspond au cas étudié dans cette thèse.

1.5.1.2 Indisponibilités imprévues

Dans les systèmes réels, certaines interruptions peuvent apparaître de manière aléatoire, comme les pannes machines.

Ces situations nécessitent généralement des approches dynamiques capables de réagir aux modifications de l'environnement.

1.5.2 Effet des indisponibilités sur les problèmes classiques

L'ajout de périodes d'indisponibilité modifie profondément les propriétés des problèmes classiques.

Considérons le problème sans interruption :

$$1 || \sum w_i C_i$$

Pour ce problème, la règle WSPT de Smith [8] fournit une solution optimale en classant les tâches selon :

$$\frac{p_i}{w_i}$$

Cependant, lorsque des périodes d'indisponibilité sont introduites :

$$1 |h_k| \sum w_i C_i$$

la séquence obtenue par WSPT peut ne plus être optimale.

En effet, une tâche possédant un faible ratio :

$$\frac{p_i}{w_i}$$

peut être placée juste avant une période d'arrêt, ce qui entraîne un retard important de son achèvement.

Ainsi, l'évaluation d'une séquence dépend non seulement de l'ordre relatif des tâches, mais également de leur position par rapport aux périodes d'indisponibilité.

1.5.3 Approches exactes pour les problèmes avec indisponibilité

Plusieurs travaux ont étudié l'intégration des contraintes de disponibilité dans les modèles d'ordonnancement.

Les méthodes exactes basées sur la programmation mathématique permettent de représenter explicitement les périodes d'arrêt à travers des variables binaires supplémentaires.

Cependant, l'ajout de ces contraintes augmente fortement la complexité du modèle, particulièrement lorsque le nombre de tâches et le nombre de périodes d'indisponibilité augmentent.

Les formulations par programmation linéaire en nombres entiers mixtes (PLNE) sont alors principalement utilisées pour obtenir des solutions optimales sur des instances de petite taille ou comme référence pour évaluer les méthodes approchées [3, 14].

1.5.4 Approches heuristiques et métaheuristiques

Face aux limites des méthodes exactes, plusieurs approches approchées ont été proposées afin de résoudre les problèmes d'ordonnancement avec contraintes de disponibilité.

Les heuristiques de priorité constituent souvent la première étape de construction des solutions. La règle WSPT représente une méthode efficace pour générer une solution initiale dans les problèmes visant la minimisation de :

$$\sum w_i C_i$$

Cependant, lorsque les contraintes supplémentaires deviennent importantes, des métaheuristiques sont nécessaires pour améliorer les solutions obtenues.

Les algorithmes génétiques ont été largement utilisés grâce à leur capacité à explorer globalement l'espace des solutions par l'intermédiaire d'une population de chromosomes [9].

La recherche Tabou constitue également une approche adaptée aux problèmes d'ordonnancement complexes grâce à son mécanisme de mémoire permettant d'explorer des solutions voisines tout en évitant les cycles [10].

1.6 État de l'art des méthodes de résolution des problèmes d'ordonnancement

Les problèmes d'ordonnancement ont fait l'objet de nombreux travaux de recherche en raison de leur importance théorique et pratique. Les approches développées peuvent être regroupées en trois grandes catégories : les méthodes exactes, les méthodes heuristiques et les méthodes métaheuristiques.

L'évolution des méthodes de résolution montre une transition progressive des approches garantissant l'optimalité vers des méthodes capables de fournir des solutions de haute qualité pour des instances de grande taille.

1.6.1 Travaux fondamentaux sur l'ordonnancement classique

Les premiers travaux sur l'ordonnancement ont principalement concerné les problèmes classiques sans contraintes supplémentaires.

Smith [8] a proposé une règle de priorité optimale pour le problème à une machine visant la minimisation de la somme pondérée des dates d'achèvement :

$$1|| \sum_{i=1}^n w_i C_i$$

Cette règle, connue sous le nom de WSPT (*Weighted Shortest Processing Time*), consiste à ordonner les tâches selon le rapport :

$$\frac{p_i}{w_i}$$

en classant les tâches par ordre croissant de cette valeur.

Cependant, cette propriété d'optimalité disparaît lorsque des contraintes supplémentaires sont introduites, notamment les périodes d'indisponibilité des machines.

Graham et al. [1] ont proposé une classification générale des problèmes d'ordonnancement à travers la notation :

$$\alpha|\beta|\gamma$$

qui constitue aujourd'hui une référence pour la description des problèmes d'ordonnement.

Cette notation permet de distinguer :

- la configuration des machines ;
- les contraintes spécifiques ;
- le critère d'optimisation.

1.6.2 Complexité et limites des méthodes exactes

L'analyse de la complexité des problèmes d'ordonnancement a montré que de nombreux problèmes appartiennent à la classe NP-difficile.

Lenstra et al. [5] ont étudié la complexité des problèmes de séquençement et ont montré que plusieurs variantes deviennent difficilement résolubles lorsque la taille des instances augmente.

Garey et Johnson [4] ont également présenté une classification des problèmes d'optimisation combinatoire difficiles, parmi lesquels plusieurs problèmes d'ordonnement.

Ces résultats expliquent les limites des méthodes exactes telles que :

- la programmation dynamique ;

- la programmation linéaire en nombres entiers ;
- les méthodes Branch and Bound.

Bien que ces méthodes permettent d'obtenir des solutions optimales, leur temps de calcul peut devenir prohibitif pour les instances industrielles de grande taille.

1.6.3 Développement des approches heuristiques

Afin de réduire le coût computationnel, des règles heuristiques ont été développées pour construire rapidement des solutions réalisables.

Les règles de priorité constituent une première catégorie d'approches heuristiques. Elles permettent de définir un ordre d'exécution des tâches à partir d'un indicateur spécifique.

Parmi ces règles :

- SPT (*Shortest Processing Time*);
- WSPT (*Weighted Shortest Processing Time*);
- EDD (*Earliest Due Date*).

Baker [15] a présenté une synthèse des principales règles de séquençement utilisées dans les problèmes classiques.

Cependant, ces méthodes restent limitées car elles dépendent fortement de la structure du problème et ne permettent généralement pas d'explorer suffisamment l'espace des solutions.

1.6.4 Développement des métaheuristiques

Pour dépasser les limites des heuristiques classiques, les méthodes métaheuristiques ont été développées comme des stratégies générales d'optimisation.

Talbi [11] définit les métaheuristiques comme des méthodes d'optimisation capables de guider efficacement une recherche dans un espace de solutions complexe grâce à des mécanismes d'intensification et de diversification.

1.6.4.1 Algorithmes génétiques

Les algorithmes génétiques ont été introduits par Holland [9]. Ils sont inspirés des mécanismes de sélection naturelle et de l'évolution biologique.

Une solution est représentée par un chromosome et une population de solutions évolue à travers plusieurs opérateurs :

- sélection ;
- croisement ;

- mutation.

Dans les problèmes d'ordonnancement, les représentations par permutation sont largement utilisées afin de garantir la faisabilité des séquences.

1.6.4.2 Recherche Tabou

La recherche Tabou proposée par Glover [10] repose sur une exploration locale améliorée par une mémoire adaptative.

Contrairement aux méthodes locales classiques, elle autorise temporairement des mouvements non améliorants afin d'éviter le blocage dans des optimums locaux.

Son principe repose principalement sur :

- la définition d'un voisinage ;
- une liste Tabou mémorisant les mouvements récents ;
- un critère d'aspiration permettant d'accepter certains mouvements interdits.

Cette approche a été largement appliquée aux problèmes d'ordonnancement complexes.

1.6.5 Synthèse des travaux antérieurs

Le tableau suivant résume quelques travaux fondamentaux liés au problème étudié.

TAB. 1.2 : Synthèse des principaux travaux liés à l'ordonnancement

Auteur	Domaine étudié	Contribution principale
Smith (1956)	Machine unique	Introduction de la règle WSPT pour minimiser $\sum w_i C_i$.
Graham et al. (1979)	Classification des problèmes	Proposition de la notation $\alpha \beta \gamma$.
Lenstra et al. (1977)	Complexité	Analyse de la difficulté algorithmique des problèmes de séquencement.
Potts et Van Wassenhove (1985)	Ordonnancement	Étude des méthodes approchées pour les problèmes difficiles.
Osman et Laporte (1996)	Métaheuristiques	Synthèse des méthodes méta-heuristiques en optimisation combinatoire.
Glover (1997)	Recherche Tabou	Formalisation de la méthode Tabu Search.
Holland (1975)	Algorithmes génétiques	Introduction du principe des algorithmes génétiques.

1.6.6 Synthèse des lacunes scientifiques

L'étude bibliographique montre que les problèmes classiques d'ordonnancement ont été largement étudiés. Cependant, l'intégration simultanée des caractéristiques suivantes reste un défi :

- machine unique ;
- tâches indépendantes et non préemptives ;
- périodes d'indisponibilité connues ;
- critère de minimisation :

$$\sum w_i C_i$$

- résolution par comparaison entre GA et TS.

La contribution de cette thèse s'inscrit donc dans cette problématique à travers le développement d'un modèle exact et l'adaptation de deux approches métaheuristiques complémentaires.

1.7 Positionnement par rapport aux travaux récents

Les recherches récentes en ordonnancement industriel accordent une attention croissante à l'intégration de contraintes reflétant davantage les conditions réelles d'exploitation, notamment les périodes d'indisponibilité des machines, les opérations de maintenance et les contraintes de disponibilité des ressources. Cette évolution conduit à enrichir les modèles classiques d'ordonnancement afin de mieux représenter les interruptions et les contraintes opérationnelles rencontrées dans les systèmes de production [16–18].

Parallèlement, l'augmentation de la taille des instances et de la complexité des contraintes renforce l'intérêt des méthodes approchées et métaheuristiques. Les travaux récents montrent notamment le recours à des méthodes de recherche avancées et à des approches hybrides pour obtenir des solutions de bonne qualité dans des temps de calcul compatibles avec des problèmes de plus grande dimension [18–20].

Dans ce contexte, les travaux récents peuvent être regroupés autour de deux axes complémentaires :

- l'amélioration des formulations et des méthodes exactes afin de représenter plus fidèlement les contraintes d'ordonnancement et de disponibilité ;
- le développement et l'adaptation de méthodes heuristiques, métaheuristiques et hybrides afin de traiter efficacement des instances de complexité croissante.

1.7.1 Évolution des méthodes exactes

Les approches exactes demeurent une référence importante dans l'étude des problèmes d'ordonnancement, puisqu'elles permettent de garantir l'optimalité des solutions lorsque la taille et la structure des instances restent compatibles avec les capacités de résolution disponibles.

Les formulations basées sur la programmation linéaire en nombres entiers mixtes (PLNE), ainsi que les méthodes de séparation et évaluation (*Branch and Bound*), permettent notamment de représenter explicitement les contraintes de précédence, de capacité et de disponibilité des ressources [3, 14].

Toutefois, l'augmentation du nombre de tâches et l'introduction de contraintes supplémentaires peuvent accroître considérablement l'effort de calcul requis. Les travaux récents continuent ainsi à associer méthodes exactes et méthodes approchées selon la taille et la structure des problèmes considérés [18].

Dans ce contexte, les méthodes exactes jouent également un rôle important comme référence pour évaluer la qualité des solutions produites par les heuristiques et les métaheuristiques sur des instances pour lesquelles l'optimum peut être déterminé.

1.7.2 Développement des approches métaheuristiques récentes

Face aux limites computationnelles des méthodes exactes sur certaines instances de grande taille, les métaheuristiques occupent une place importante dans la résolution des problèmes d'ordonnancement. Elles visent à obtenir des solutions de bonne qualité dans des temps de calcul raisonnables, sans garantir nécessairement l'optimalité [11, 19].

Les recherches récentes montrent également un intérêt pour la combinaison de plusieurs mécanismes de recherche afin d'améliorer l'équilibre entre exploration et intensification. Ces approches peuvent associer des heuristiques constructives, des recherches locales et différentes métaheuristiques [18, 19].

Parmi les approches fréquemment mobilisées dans la littérature figurent notamment :

- les algorithmes génétiques ;
- la recherche Tabou ;
- le recuit simulé ;
- les approches hybrides combinant plusieurs mécanismes de recherche.

Les algorithmes génétiques reposent sur l'évolution d'une population de solutions et permettent d'explorer simultanément différentes régions de l'espace de recherche. Leur efficacité dépend notamment de la représentation des solutions, du mécanisme de sélection ainsi que des opérateurs de croisement et de mutation [9, 11].

Dans les problèmes d’ordonnancement représentés par des permutations, les opérateurs génétiques doivent préserver la validité des séquences. Des opérateurs adaptés aux permutations, tels que le croisement OX (*Order Crossover*) et les mutations par insertion ou échange, peuvent ainsi être utilisés dans la conception d’un algorithme génétique.

La recherche Tabou repose, quant à elle, sur une exploration locale guidée par des mécanismes de mémoire. L’utilisation d’une liste Tabou permet notamment de limiter les retours cycliques vers des solutions récemment explorées, tandis que les mécanismes d’aspiration, d’intensification et de diversification permettent de contrôler la dynamique de la recherche [10, 11].

Des travaux récents continuent à mobiliser différentes métaheuristiques dans les problèmes d’ordonnancement, notamment la recherche Tabou et le recuit simulé, ce qui confirme l’intérêt de ces approches pour l’exploration de problèmes combinatoires complexes [18].

1.7.3 Travaux récents sur les contraintes de disponibilité

Les problèmes d’ordonnancement intégrant des contraintes de disponibilité des machines demeurent une problématique active de recherche. La littérature a montré que l’introduction de périodes pendant lesquelles une machine ne peut pas traiter de tâche modifie les propriétés des modèles classiques et nécessite des méthodes de résolution adaptées [6, 7, 16].

Les recherches plus récentes confirment l’intérêt porté à des modèles d’ordonnancement intégrant des contraintes réalistes liées à la disponibilité, à la maintenance et aux conditions d’exploitation des ressources [16, 21].

Ces travaux montrent également que la nature des contraintes de disponibilité, le critère d’optimisation retenu et la taille des instances influencent directement le choix de la méthode de résolution. Selon la structure du problème, des approches exactes, heuristiques, métaheuristiques ou hybrides peuvent être envisagées [18, 19].

Dans cette perspective, les recherches consacrées aux contraintes de disponibilité s’orientent notamment vers :

- une représentation plus réaliste des périodes d’indisponibilité et des contraintes de maintenance ;
- le développement de formulations capables d’intégrer explicitement ces contraintes dans le processus d’ordonnancement ;
- la conception de méthodes approchées permettant de traiter des instances de taille croissante ;
- l’évaluation du compromis entre qualité des solutions et temps de calcul.

Le problème étudié dans cette thèse s’inscrit dans cette orientation, tout en se distinguant par la combinaison d’une machine unique, de tâches indépendantes et non préemptives, de périodes

d'indisponibilité déterministes connues à l'avance et du critère de minimisation de la somme pondérée des dates d'achèvement.

1.7.4 Positionnement scientifique de la thèse

L'analyse de la littérature montre que les problèmes d'ordonnancement avec indisponibilité des machines représentent une extension importante des modèles classiques.

Cependant, plusieurs difficultés demeurent :

- la complexité combinatoire augmente avec le nombre de tâches ;
- les règles classiques de priorité ne garantissent plus l'optimalité ;
- les contraintes de disponibilité modifient fortement la structure du voisinage des solutions ;
- les méthodes exactes deviennent rapidement limitées.

Dans ce contexte, cette thèse se positionne dans l'étude d'un problème :

$$1|h_k| \sum w_i C_i$$

avec tâches indépendantes et non préemptives sur une machine unique.

La contribution proposée repose sur :

- une formulation mathématique exacte du problème ;
- une génération de solution initiale basée sur la règle WSPT ;
- une adaptation de l'algorithme génétique ;
- une adaptation de la recherche Tabou ;
- une analyse comparative des performances obtenues.

1.8 Conclusion du chapitre

Ce premier chapitre a permis de présenter les fondements théoriques nécessaires à l'étude des problèmes d'ordonnancement et de situer précisément la problématique étudiée dans le contexte de la recherche en optimisation combinatoire.

Dans un premier temps, les concepts fondamentaux liés à l'ordonnancement ont été présentés en précisant les notions de tâche, de séquence, d'ordonnancement, de ressource et de critère d'optimisation. Une attention particulière a été accordée aux problèmes d'ordonnancement sur machine unique, qui constituent le cadre général de cette étude.

Ensuite, les principales contraintes rencontrées dans les systèmes industriels ont été analysées, notamment les contraintes de capacité, de non-préemption et de disponibilité des machines. L'intégration des périodes d'indisponibilité constitue une extension importante des modèles classiques, car elle modifie l'espace des solutions réalisables et augmente la complexité du problème.

L'analyse de la classification des problèmes d'ordonnancement selon la notation $\alpha|\beta|\gamma$ a permis de positionner précisément le problème traité. Celui-ci correspond à un problème de type :

$$1|h_k|\sum_{i=1}^n w_i C_i$$

caractérisé par une machine unique, des tâches indépendantes et non préemptives, des périodes d'indisponibilité connues à l'avance et un objectif de minimisation de la somme pondérée des dates d'achèvement.

L'étude bibliographique a également montré que les méthodes exactes, malgré leur capacité à fournir des solutions optimales, deviennent rapidement limitées pour les instances de grande taille en raison de la complexité combinatoire du problème. Cette difficulté a motivé le développement d'approches heuristiques et métaheuristiques capables d'explorer efficacement de grands espaces de solutions.

Les travaux antérieurs ont mis en évidence l'importance des règles de priorité, notamment la règle WSPT, qui fournit une solution optimale pour le problème classique sans indisponibilité. Cependant, lorsque des contraintes de disponibilité des machines sont introduites, cette propriété n'est plus garantie, ce qui justifie l'utilisation de méthodes d'amélioration avancées.

Dans ce contexte, les approches métaheuristiques apparaissent comme des outils adaptés. L'algorithme génétique offre une capacité d'exploration globale grâce à l'évolution d'une population de solutions, tandis que la recherche Tabou permet une exploration locale guidée par une mémoire adaptative.

Ainsi, ce chapitre a permis d'identifier la problématique scientifique, de justifier le choix des méthodes de résolution et de mettre en évidence la nécessité d'une approche adaptée aux caractéristiques particulières du problème étudié.

Le chapitre suivant sera consacré à la formulation mathématique détaillée du problème ainsi qu'à la présentation des méthodes de résolution proposées. Il présentera le modèle exact sous forme de programme linéaire en nombres entiers mixtes, la génération de la solution initiale par la règle WSPT ainsi que l'adaptation des approches métaheuristiques retenues.

Chapitre 2

Méthodes de résolution des problèmes d'ordonnancement

2.1 Introduction

Après avoir présenté, dans le chapitre précédent, les fondements théoriques des problèmes d'ordonnancement ainsi que les principales caractéristiques du problème étudié, le présent chapitre s'intéresse aux méthodes de résolution développées dans la littérature pour traiter ces problèmes.

Les problèmes d'ordonnancement constituent une classe importante de problèmes d'optimisation combinatoire rencontrés dans de nombreux domaines, notamment la production manufacturière, la maintenance, les systèmes informatiques et la gestion des ressources. Leur résolution consiste généralement à déterminer une organisation temporelle des tâches permettant d'optimiser un ou plusieurs critères de performance tout en respectant les contraintes imposées par le système considéré [2,3].

La difficulté de résolution dépend fortement de la structure du problème, du nombre de tâches et de ressources, ainsi que de la nature des contraintes considérées. Pour une machine unique et n tâches indépendantes, l'espace des séquences peut comporter jusqu'à $n!$ permutations. Toutefois, la taille factorielle de cet espace ne constitue pas, à elle seule, une preuve de NP-difficulté. La complexité algorithmique doit être établie pour chaque variante du problème à partir de résultats de complexité appropriés [2,4].

Certaines classes de problèmes d'ordonnancement peuvent être résolues en temps polynomial grâce à des propriétés structurelles particulières, tandis que d'autres sont NP-difficiles et nécessitent des méthodes de résolution plus élaborées. L'introduction de contraintes supplémentaires peut notamment modifier profondément la structure et la difficulté du problème [3].

Les contraintes de disponibilité des machines constituent, à cet égard, une extension particulièrement importante des modèles classiques. Dans les environnements industriels réels, une machine peut être temporairement indisponible en raison d'une opération de maintenance pré-

ventive, d'un arrêt technique planifié, d'une inspection ou d'une autre intervention. Lorsque ces périodes d'indisponibilité sont connues à l'avance, elles doivent être explicitement intégrées dans la construction et l'évaluation des ordonnancements [6, 13].

Dans le cadre de cette thèse, l'attention est portée sur un problème d'ordonnancement sur une machine unique comportant des périodes d'indisponibilité déterministes, des tâches indépendantes et non préemptives, et ayant pour objectif la minimisation de la somme pondérée des dates d'achèvement. Selon la notation adoptée dans cette thèse, ce problème est désigné par :

$$1|h_k|\sum_{i=1}^n w_i C_i.$$

La présence de périodes d'indisponibilité modifie la construction des ordonnancements réalisables et peut remettre en cause certaines propriétés d'optimalité valables dans les modèles classiques à disponibilité continue. Elle nécessite par conséquent le recours à des méthodes de résolution capables de prendre simultanément en considération le séquençement des tâches, leurs caractéristiques et les intervalles durant lesquels la machine n'est pas disponible [6].

Les méthodes développées pour résoudre les problèmes d'ordonnancement peuvent être regroupées en plusieurs familles. Les méthodes exactes visent à déterminer et à certifier une solution optimale. Elles comprennent notamment l'énumération exhaustive, la programmation dynamique, la programmation mathématique et les méthodes de séparation et évaluation (*Branch-and-Bound*). Leur efficacité dépend cependant fortement de la structure et de la taille des instances considérées [2, 3].

Lorsque la résolution exacte devient trop coûteuse, les méthodes approchées constituent une alternative importante. Les heuristiques exploitent généralement des propriétés particulières du problème afin de construire rapidement des solutions réalisables, tandis que les métaheuristiques mettent en œuvre des mécanismes plus généraux d'exploration de l'espace de recherche. Elles visent à obtenir des solutions de haute qualité dans des temps de calcul compatibles avec les exigences pratiques, sans garantir nécessairement l'optimalité globale.

Dans cette perspective, ce chapitre présente et analyse les principales méthodes de résolution susceptibles d'être mobilisées pour le problème étudié. Les méthodes exactes sont d'abord examinées, suivies des méthodes heuristiques et des approches métaheuristiques. Une attention particulière est ensuite accordée aux algorithmes génétiques et à la recherche Tabou, qui constituent les deux approches retenues dans cette thèse. Enfin, une analyse comparative permet de mettre en évidence les avantages et les limites des différentes familles de méthodes et de justifier le choix des approches développées dans la suite du travail.

2.2 Méthodes exactes de résolution

Les méthodes exactes regroupent les approches algorithmiques permettant, sous réserve de mener le processus de résolution à son terme, de déterminer une solution optimale et d'en établir

l'optimalité. Elles occupent une place fondamentale en optimisation combinatoire, aussi bien pour la résolution directe de certaines instances que pour la production de valeurs de référence permettant d'évaluer la qualité des méthodes approchées [2, 3].

Dans les problèmes d'ordonnancement, ces méthodes reposent sur différentes stratégies permettant d'organiser et de réduire l'exploration de l'espace des solutions. Elles peuvent exploiter, selon le problème considéré, des relations de récurrence, des bornes inférieures et supérieures, des règles de dominance, des relaxations mathématiques ou encore des mécanismes d'élagage permettant d'éliminer les sous-problèmes ne pouvant pas conduire à une meilleure solution.

La principale force des méthodes exactes réside dans la garantie d'optimalité qu'elles peuvent fournir. Cette propriété est particulièrement importante lorsque l'objectif consiste à établir des résultats de référence ou à mesurer précisément la performance d'une méthode heuristique ou métaheuristique.

Cependant, leur efficacité computationnelle dépend fortement de la structure du problème. Pour certaines classes d'ordonnancement, des algorithmes polynomiaux existent, alors que d'autres variantes sont NP-difficiles [2, 4]. L'introduction de contraintes supplémentaires, notamment des périodes d'indisponibilité des machines, peut accroître sensiblement la difficulté de résolution. La littérature consacrée aux contraintes de disponibilité montre ainsi que la complexité dépend du nombre et de la nature des périodes d'indisponibilité, de l'environnement des machines, du caractère préemptif ou non préemptif des tâches et du critère d'optimisation considéré [6, 13].

Parmi les principales méthodes exactes utilisées dans le domaine de l'ordonnancement, on distingue notamment :

- l'énumération complète des solutions admissibles ;
- la programmation dynamique ;
- la programmation mathématique, notamment les formulations en nombres entiers ou mixtes ;
- les méthodes de séparation et évaluation (*Branch-and-Bound*).

Ces approches présentent des principes de fonctionnement différents, mais poursuivent un objectif commun : exploiter la structure mathématique du problème afin d'identifier une solution optimale tout en réduisant, autant que possible, l'effort computationnel nécessaire.

Dans le contexte de cette thèse, l'étude des méthodes exactes présente un double intérêt. D'une part, elle permet de comprendre les mécanismes fondamentaux utilisés pour traiter les problèmes d'ordonnancement. D'autre part, elle permet de mettre en évidence les limites computationnelles qui motivent ensuite le recours à des approches heuristiques et métaheuristiques pour les instances de plus grande taille.

2.2.1 Énumération complète

L'énumération complète, également appelée recherche exhaustive, constitue l'une des approches exactes les plus directes pour résoudre un problème d'ordonnancement. Elle consiste à générer l'ensemble des séquences admissibles, à évaluer chacune d'elles selon le critère d'optimisation considéré, puis à retenir celle qui présente la meilleure valeur de la fonction objectif [2, 3].

Dans le cas d'un problème non préemptif comportant n tâches distinctes à ordonnancer sur une machine unique, et en l'absence de restrictions éliminant a priori certaines permutations, le nombre de séquences possibles est :

$$N = n!.$$

La taille de l'espace de recherche augmente donc de manière factorielle avec le nombre de tâches. À titre d'illustration :

$$10! = 3\,628\,800, \quad 20! \approx 2,43 \times 10^{18}.$$

Ces valeurs montrent que l'examen systématique de toutes les permutations devient rapidement impraticable. Une procédure qui parcourt explicitement les $n!$ séquences possède un nombre factoriel de candidats à examiner. De plus, si l'évaluation complète d'une séquence nécessite un parcours des n tâches, une implémentation directe peut nécessiter un effort de l'ordre de $O(n \cdot n!)$.

Il convient toutefois de souligner que cette croissance factorielle de l'espace des permutations ne constitue pas, à elle seule, une preuve de NP-difficulté. La complexité d'une variante d'ordonnancement doit être établie à partir d'une analyse algorithmique propre au problème considéré [4, 5].

Dans le problème étudié dans cette thèse, l'évaluation d'une séquence doit également tenir compte des périodes d'indisponibilité de la machine. Pour chaque séquence candidate, il est donc nécessaire de construire un ordonnancement réalisable respectant les intervalles d'indisponibilité avant de calculer la valeur du critère

$$\sum_{i=1}^n w_i C_i.$$

La présence de ces contraintes rend l'utilisation d'une énumération exhaustive d'autant moins attractive lorsque la taille des instances augmente [6, 13].

Malgré cette limitation, l'énumération complète conserve un intérêt méthodologique pour les très petites instances. Elle peut notamment servir à déterminer une solution optimale de référence et, par conséquent, à évaluer la qualité des solutions produites par des méthodes heuristiques ou métaheuristiques. Son utilisation pratique demeure néanmoins limitée par la croissance factorielle du nombre de séquences à examiner [2, 3].

2.2.2 Programmation dynamique

La programmation dynamique est une méthode exacte fondée sur le principe d'optimalité de Bellman. Elle consiste à décomposer un problème d'optimisation en une succession de sous-problèmes interdépendants et à exploiter les solutions optimales de ces sous-problèmes afin de construire progressivement une solution optimale du problème initial [22]. Cette approche est particulièrement adaptée aux problèmes possédant une structure séquentielle et des sous-problèmes qui se répètent.

Dans le domaine de l'ordonnancement, la programmation dynamique peut être utilisée lorsque l'état du système peut être décrit par un ensemble limité d'informations permettant de caractériser les décisions déjà prises. Pour un ensemble de tâches $I = \{1, \dots, n\}$, un état peut, par exemple, être défini par un sous-ensemble $J \subseteq I$ représentant les tâches déjà ordonnancées ou, selon la formulation adoptée, les tâches restant à ordonnancer [2, 3].

Soit $F(J)$ la valeur optimale associée à un sous-problème défini par l'ensemble J . Une relation de récurrence générique peut alors s'écrire sous la forme :

$$F(J) = \min_{i \in J} \{F(J \setminus \{i\}) + g_i(J)\},$$

avec la condition initiale :

$$F(\emptyset) = 0,$$

où $g_i(J)$ représente la contribution induite par le choix de la tâche i dans l'état considéré. La définition précise de cette contribution dépend du problème d'ordonnancement et du critère d'optimisation étudiés.

Dans le problème classique sur une machine unique visant à minimiser la somme pondérée des dates d'achèvement, si la tâche i est placée en dernière position parmi les tâches de J , sa date d'achèvement est égale à la somme des durées de traitement des tâches appartenant à J . En posant

$$P(J) = \sum_{j \in J} p_j,$$

une récurrence peut alors être exprimée sous la forme :

$$F(J) = \min_{i \in J} \{F(J \setminus \{i\}) + w_i P(J)\}.$$

Cette formulation illustre le principe de la programmation dynamique : la solution optimale associée à un ensemble de tâches est construite à partir des solutions optimales de sous-ensembles de taille inférieure.

Toutefois, l'introduction de périodes d'indisponibilité de la machine modifie cette structure. Dans ce cas, la date d'achèvement d'une tâche ne dépend plus uniquement de la somme des

durées de traitement des tâches qui la précèdent, mais également de leur position temporelle par rapport aux intervalles d'indisponibilité. Une formulation dynamique adaptée au problème étudié doit donc enrichir la description de l'état afin d'intégrer suffisamment d'informations temporelles pour déterminer la faisabilité et le coût des décisions successives.

La programmation dynamique permet d'éviter la réévaluation répétée de sous-problèmes identiques en mémorisant les valeurs déjà calculées. Cependant, cette réduction ne supprime pas nécessairement la croissance combinatoire du nombre d'états. Une formulation reposant sur l'ensemble des sous-ensembles de n tâches peut comporter jusqu'à

$$2^n$$

états potentiels. Les besoins en temps de calcul et en mémoire peuvent ainsi devenir importants lorsque n augmente [2, 3].

La programmation dynamique constitue donc une méthode exacte particulièrement intéressante lorsque le problème possède une structure permettant de définir des états et des relations de récurrence de taille maîtrisée. Dans le contexte de périodes d'indisponibilité, son application nécessite cependant une représentation plus riche de l'état du système, ce qui peut accroître sensiblement la complexité de la procédure. Cette limite contribue à expliquer le recours à d'autres méthodes exactes, telles que le Branch-and-Bound et la programmation mathématique, ainsi qu'à des méthodes approchées pour les instances de plus grande taille. ““

2.2.3 Méthode Branch-and-Bound

La méthode Branch-and-Bound, ou méthode de séparation et évaluation, constitue l'une des principales méthodes exactes utilisées en optimisation combinatoire. L'un des travaux fondateurs de cette approche est celui de Land et Doig [23]. Elle repose sur une exploration structurée de l'espace des solutions permettant d'éviter, dans de nombreux cas, l'énumération explicite de l'ensemble des configurations possibles.

Le principe consiste à représenter l'espace de recherche sous la forme d'une arborescence. Chaque nœud correspond à un sous-problème obtenu en fixant progressivement certaines décisions, tandis que les branches représentent les choix permettant de construire une solution complète. Dans un problème d'ordonnancement, ces décisions peuvent notamment concerner la position d'une tâche dans la séquence ou les relations de précédence entre plusieurs tâches [2,3].

La procédure repose principalement sur trois mécanismes complémentaires :

- **Séparation (*Branching*)** : le problème initial est décomposé en plusieurs sous-problèmes en fixant progressivement certaines décisions d'ordonnancement ;
- **Évaluation (*Bounding*)** : une borne est calculée pour chaque sous-problème afin d'estimer la meilleure valeur de la fonction objectif susceptible d'être obtenue à partir de celui-ci ;

- **Élagage (*Fathoming*)** : les sous-problèmes qui ne peuvent pas conduire à une solution meilleure que la meilleure solution réalisable connue sont éliminés de l'espace de recherche.

Dans le cas d'un problème de minimisation, soit UB la valeur de la meilleure solution réalisable connue, qui constitue une borne supérieure, et soit $LB(Y)$ une borne inférieure associée à un nœud Y de l'arborescence. Si :

$$LB(Y) \geq UB,$$

alors aucune solution issue du sous-problème représenté par Y ne peut améliorer la meilleure solution connue. Le nœud Y , ainsi que l'ensemble de ses descendants, peut donc être élagué.

L'efficacité de la méthode dépend fortement de la qualité des bornes utilisées. Une borne inférieure suffisamment serrée permet d'éliminer rapidement les sous-problèmes non prometteurs. De même, la disponibilité précoce d'une bonne solution réalisable permet d'obtenir une borne supérieure de qualité et d'accroître l'efficacité de l'élagage. Des règles de dominance, des relaxations du problème et différentes stratégies de sélection des nœuds peuvent également être utilisées pour améliorer les performances de la recherche [2, 3].

Dans les problèmes d'ordonnancement avec contraintes de disponibilité, la construction des bornes revêt une importance particulière, puisque les périodes pendant lesquelles la machine est indisponible modifient l'ensemble des ordonnancements réalisables. La littérature montre que les méthodes exactes peuvent être efficaces pour certaines variantes et certaines tailles d'instances, mais que leurs performances dépendent fortement de la structure des contraintes de disponibilité considérées [6, 7, 13].

Ainsi, la méthode Branch-and-Bound fournit un cadre exact permettant de certifier l'optimalité d'une solution tout en réduisant potentiellement l'espace exploré par rapport à une énumération exhaustive. Néanmoins, dans le pire cas, une part importante de l'arborescence peut encore devoir être explorée. Cette limite justifie le recours à d'autres approches lorsque la taille ou la complexité des instances devient importante.

2.2.4 Programmation mathématique

La programmation mathématique constitue un cadre général de modélisation et de résolution des problèmes d'optimisation. Elle permet de représenter explicitement une fonction objectif, des variables de décision et un ensemble de contraintes traduisant les caractéristiques du système étudié. Dans le domaine de l'ordonnancement, les formulations en programmation linéaire en nombres entiers ou mixtes occupent une place importante, notamment lorsqu'il est nécessaire de représenter des décisions de séquençement, d'affectation ou de positionnement temporel [2, 3].

De manière générale, un programme linéaire en nombres entiers mixtes (*Mixed-Integer Linear Programming*, MILP) peut s'écrire sous la forme :

$$\min c^T x$$

sous les contraintes :

$$Ax \leq b,$$

$$x_j \in \mathbb{Z}, \quad \forall j \in \mathcal{I},$$

$$x_j \in \mathbb{R}, \quad \forall j \notin \mathcal{I},$$

où x représente le vecteur des variables de décision, c le vecteur des coefficients de la fonction objectif, A la matrice des coefficients des contraintes, b le vecteur des seconds membres et $\mathcal{I}$ l'ensemble des indices des variables soumises à des restrictions d'intégralité.

Dans un problème d'ordonnancement, les variables peuvent notamment représenter les dates de début et d'achèvement des tâches ainsi que des décisions binaires de précédence. Pour deux tâches J_i et J_j , une variable binaire peut, par exemple, être définie par :

$$x_{ij} = \begin{cases} 1, & \text{si la tâche } J_i \text{ précède } J_j, \\ 0, & \text{sinon.} \end{cases}$$

Cette représentation permet de formuler des contraintes de non-chevauchement entre les tâches. Pour chaque paire de tâches distinctes, les variables de précédence permettent de déterminer laquelle doit être exécutée avant l'autre. Des variables binaires supplémentaires peuvent également être introduites afin de positionner une tâche avant ou après une période d'indisponibilité donnée.

Dans le problème considéré dans cette thèse, la programmation mathématique présente donc un intérêt particulier, puisqu'elle permet de représenter explicitement la fonction objectif :

$$\min \sum_{i=1}^n w_i C_i,$$

ainsi que les contraintes de séquençement, de non-chevauchement entre les tâches et de respect des périodes d'indisponibilité de la machine dans un contexte non préemptif.

Les formulations MILP présentent l'avantage de fournir un cadre rigoureux et générique, compatible avec des solveurs d'optimisation capables d'exploiter des techniques telles que le Branch-and-Bound et le Branch-and-Cut. Elles permettent également, lorsque la résolution est menée jusqu'à l'optimalité, de disposer d'une certification de l'optimalité de la solution obtenue [3].

Toutefois, la taille d'une formulation peut augmenter rapidement avec le nombre de tâches. Dans une formulation fondée sur des décisions binaires de précédence pour chaque paire de

tâches, le nombre de paires non ordonnées est égal à :

$$\binom{n}{2} = \frac{n(n-1)}{2},$$

ce qui conduit à une croissance quadratique, de l'ordre de $O(n^2)$, du nombre de décisions de précédence. L'introduction de variables supplémentaires pour représenter le positionnement des tâches par rapport aux périodes d'indisponibilité accroît encore la taille du modèle. Par conséquent, la résolution exacte de grandes instances peut devenir coûteuse en temps de calcul et en mémoire.

La formulation mathématique détaillée correspondant au problème $1|h_k|\sum_{i=1}^n w_i C_i$ sera présentée dans le chapitre consacré à la modélisation du problème. Dans la présente section, l'objectif est essentiellement de situer la programmation mathématique parmi les principales méthodes exactes de résolution.

2.3 Méthodes approchées et heuristiques

Lorsque la taille ou la structure d'un problème rend sa résolution exacte trop coûteuse, les méthodes approchées constituent une alternative permettant de rechercher des solutions de bonne qualité avec un effort computationnel maîtrisé. Contrairement aux méthodes exactes, elles ne fournissent généralement pas de certificat d'optimalité, mais privilégient l'efficacité pratique et la qualité des solutions obtenues [3, 11].

Il convient néanmoins de distinguer les heuristiques des métaheuristiques. Une heuristique exploite généralement des propriétés spécifiques du problème afin de construire ou d'améliorer rapidement une solution. Une métaheuristique constitue, quant à elle, un cadre de recherche plus général destiné à guider l'exploration de l'espace des solutions, notamment grâce à des mécanismes d'intensification et de diversification [11].

Dans les problèmes d'ordonnancement, les méthodes approchées peuvent être utilisées à plusieurs niveaux : construction d'une solution initiale, amélioration locale d'un ordonnancement existant, génération d'une borne supérieure pour une méthode exacte ou encore recherche directe de solutions de haute qualité pour des instances difficiles.

Cet intérêt devient particulièrement marqué lorsque des contraintes d'indisponibilité sont introduites. Ces contraintes peuvent modifier la performance des règles de priorité classiques et nécessiter des mécanismes spécifiques permettant de construire des séquences temporellement réalisables [6, 7].

2.3.1 Méthodes heuristiques

Les heuristiques sont des procédures de résolution conçues pour produire rapidement des solutions réalisables, généralement en exploitant certaines propriétés structurelles du problème

étudié. Leur faible coût computationnel les rend particulièrement intéressantes pour construire des solutions initiales ou traiter des instances pour lesquelles une résolution exacte devient trop coûteuse [3].

Dans le domaine de l'ordonnancement, plusieurs règles de priorité classiques peuvent être utilisées selon le critère d'optimisation considéré :

- **SPT (*Shortest Processing Time*)** : les tâches sont classées selon des durées de traitement croissantes. Sur une machine unique disponible en permanence, cette règle est optimale pour la minimisation de la somme des dates d'achèvement $\sum C_i$;
- **EDD (*Earliest Due Date*)** : les tâches sont classées selon des dates d'échéance croissantes. Dans le problème classique sur une machine unique, cette règle est notamment associée à la minimisation du retard algébrique maximal ;
- **WSPT (*Weighted Shortest Processing Time*)** : les tâches sont classées par ordre croissant du rapport

$$\frac{p_i}{w_i},$$

ou, de manière équivalente, par ordre décroissant de w_i/p_i . Pour le problème classique

$$1 || \sum_{i=1}^n w_i C_i,$$

cette règle fournit une séquence optimale [3, 8].

La règle WSPT présente un intérêt particulier dans cette thèse puisque le critère considéré est précisément la somme pondérée des dates d'achèvement. Cependant, sa propriété d'optimalité concerne le modèle classique dans lequel la machine est disponible de manière continue. Lorsque des périodes d'indisponibilité sont introduites, le positionnement temporel des tâches par rapport à ces périodes intervient également dans la valeur de la fonction objectif. L'ordre WSPT ne peut donc plus être considéré, de manière générale, comme garantissant l'optimalité pour le problème étudié [6, 7].

Dans cette recherche, WSPT constitue ainsi une règle constructive pertinente pour générer une solution initiale de bonne qualité, destinée à être ensuite améliorée par les métaheuristiques retenues. Cette stratégie permet de combiner l'information structurelle fournie par la règle de Smith avec des mécanismes d'exploration plus élaborés capables de prendre en compte les effets des périodes d'indisponibilité.

Les heuristiques présentent donc plusieurs avantages, notamment leur simplicité, leur rapidité et leur capacité à produire des solutions réalisables avec un faible coût computationnel. Leur principal inconvénient réside cependant dans leur caractère généralement spécifique au

problème et dans l'absence de garantie d'optimalité en dehors des classes pour lesquelles leurs propriétés ont été formellement établies [2, 3].

2.4 Métaheuristiques

Les métaheuristiques constituent une famille de méthodes d'optimisation destinées à explorer efficacement des espaces de recherche complexes lorsque l'utilisation de méthodes exactes devient difficile ou trop coûteuse. Elles fournissent un cadre général permettant de guider le processus de recherche sans nécessiter l'exploration exhaustive de l'ensemble des solutions [11].

Contrairement aux méthodes exactes, les métaheuristiques ne garantissent généralement pas l'obtention d'un optimum global. Leur objectif consiste plutôt à produire des solutions de haute qualité dans un temps de calcul compatible avec les exigences du problème étudié. Elles sont ainsi particulièrement adaptées aux problèmes d'optimisation combinatoire possédant des espaces de recherche de grande taille.

L'efficacité d'une métaheuristique repose notamment sur l'équilibre entre deux mécanismes fondamentaux : l'intensification et la diversification. L'intensification vise à exploiter en profondeur les régions prometteuses de l'espace de recherche, tandis que la diversification cherche à explorer de nouvelles régions afin de réduire le risque de convergence prématurée vers un optimum local [10, 11].

Les métaheuristiques peuvent être classées, de manière générale, selon la structure de recherche qu'elles utilisent. Une distinction courante oppose les méthodes fondées principalement sur une solution courante, telles que la recherche Tabou et le recuit simulé, aux méthodes fondées sur une population de solutions, telles que les algorithmes génétiques. Cette classification n'est toutefois pas absolue, notamment en raison du développement de nombreuses approches hybrides combinant plusieurs mécanismes de recherche.

Dans les problèmes d'ordonnancement, le choix d'une métaheuristique dépend notamment de la représentation des solutions, de la structure du voisinage, du critère d'évaluation et des contraintes propres au problème. La prise en compte des périodes d'indisponibilité constitue, à cet égard, une difficulté supplémentaire, puisqu'une séquence doit être évaluée en tenant compte des intervalles pendant lesquels la machine ne peut exécuter aucune tâche [6].

Des travaux récents confirment l'intérêt des métaheuristiques pour des problèmes d'ordonnancement intégrant des contraintes de disponibilité ou de maintenance. Chen et al. ont notamment développé un algorithme génétique pour un problème d'ordonnancement sur machine unique avec contraintes de disponibilité et maintenance préventive [20]. De même, Huang et al. ont proposé une stratégie génétique intégrant un mécanisme d'immigration pour un problème sur machine unique comportant des activités flexibles de maintenance préventive [17].

Dans cette thèse, deux métaheuristiques complémentaires sont principalement retenues : l'algorithme génétique (GA), qui repose sur l'évolution d'une population de solutions et favorise une exploration relativement globale de l'espace de recherche, et la recherche Tabou (TS), qui

s'appuie sur une exploration locale guidée par des mécanismes de mémoire. L'étude comparative de ces deux stratégies permet d'analyser leurs comportements respectifs en termes de qualité des solutions et de temps de calcul.

2.4.1 Algorithmes génétiques

Les algorithmes génétiques (AG), ou *Genetic Algorithms* (GA), constituent une famille d'algorithmes évolutionnaires inspirés des mécanismes de sélection et d'évolution des populations. Les principes fondamentaux de cette approche ont été développés par Holland [9]. Un algorithme génétique maintient une population de solutions candidates qui évolue au cours des générations sous l'action d'opérateurs de sélection, de croisement et de mutation.

Dans un problème d'ordonnancement par permutation, un individu peut être représenté par un chromosome décrivant l'ordre d'exécution des tâches. Pour n tâches, une solution candidate peut ainsi être représentée par :

$$X = (J_{\pi(1)}, J_{\pi(2)}, \dots, J_{\pi(n)}),$$

où π représente une permutation de l'ensemble des tâches.

Cette représentation est particulièrement adaptée au problème étudié dans cette thèse, puisque chaque tâche doit apparaître exactement une fois dans la séquence. La séquence représentée par le chromosome doit ensuite être transformée en un ordonnancement temporel réalisable en tenant compte des périodes d'indisponibilité de la machine.

La qualité de chaque chromosome est déterminée à partir de la fonction objectif. Dans le problème considéré, l'évaluation d'une solution repose sur :

$$F(X) = \sum_{i=1}^n w_i C_i,$$

où les dates d'achèvement C_i sont calculées en respectant les périodes d'indisponibilité de la machine. Le problème étant un problème de minimisation, une solution est considérée comme meilleure lorsqu'elle produit une valeur plus faible de $F(X)$.

Le fonctionnement général d'un algorithme génétique peut être décrit par les étapes suivantes :

1. générer une population initiale de solutions ;
2. évaluer chaque individu à l'aide de la fonction objectif ;
3. sélectionner les individus parents ;
4. appliquer un opérateur de croisement afin de produire de nouveaux individus ;
5. appliquer un opérateur de mutation afin de maintenir la diversité de la population ;

6. évaluer les nouveaux individus et construire la génération suivante ;
7. répéter le processus jusqu'à satisfaction d'un critère d'arrêt.

La sélection favorise généralement les individus présentant une meilleure qualité tout en cherchant à maintenir une diversité suffisante dans la population. Le croisement vise à transmettre aux descendants certaines caractéristiques des solutions parentes. Dans le cas d'un codage par permutation, il est nécessaire d'utiliser des opérateurs adaptés afin d'éviter la duplication ou la disparition de tâches dans les chromosomes produits.

Parmi les opérateurs adaptés aux représentations par permutation figure l'*Order Crossover* (OX), qui permet de préserver une partie de l'ordre relatif des tâches. Les opérateurs de mutation par échange (*swap*) ou par insertion peuvent également être utilisés pour modifier localement une séquence tout en conservant sa structure de permutation.

La mutation joue un rôle important dans le maintien de la diversité génétique. Elle permet d'introduire de nouvelles caractéristiques dans la population et contribue ainsi à réduire le risque de convergence prématurée. La performance d'un algorithme génétique dépend donc de plusieurs choix méthodologiques, notamment la représentation des solutions, la stratégie de sélection, les opérateurs de croisement et de mutation, la taille de la population et les critères d'arrêt.

L'utilisation des algorithmes génétiques dans des problèmes comportant des contraintes de disponibilité des machines est également documentée dans la littérature. Chen et al. ont étudié un problème d'ordonnancement sur machine unique intégrant des contraintes de disponibilité liées à la fiabilité de la machine et à la maintenance préventive. Les auteurs ont développé un algorithme génétique adapté à cette variante et évalué ses performances au moyen d'expériences numériques [20].

Plus récemment, Huang et al. ont étudié un problème sur machine unique combinant des dates de disponibilité des tâches et des activités flexibles de maintenance préventive. Ils proposent un algorithme génétique intégrant une stratégie d'immigration fondée sur la trajectoire de recherche afin d'améliorer la diversification de la population [17]. Ces travaux illustrent l'utilisation de mécanismes génétiques élaborés pour traiter des variantes d'ordonnancement soumises à des contraintes temporelles et de maintenance.

Dans le contexte spécifique de cette recherche, Lounissi et al. (2025) ont proposé une étude comparative entre un algorithme génétique et une recherche Tabou pour un problème d'ordonnancement sur machine unique visant à minimiser la somme pondérée des dates d'achèvement [24]. Cette étude constitue un point d'appui pertinent pour la comparaison des deux paradigmes de recherche considérés dans la présente thèse.

L'intérêt de l'algorithme génétique pour le problème étudié réside ainsi dans sa capacité à maintenir simultanément plusieurs solutions candidates et à explorer différentes régions de l'espace des permutations. Cette capacité d'exploration relativement globale est complémentaire de l'intensification locale offerte par la recherche Tabou, présentée dans la section suivante.

2.4.2 Recherche Tabou

La recherche Tabou, ou *Tabu Search* (TS), est une métaheuristique fondée sur une recherche locale enrichie par des mécanismes de mémoire. Elle a été développée par Glover afin de dépasser certaines limitations des procédures de recherche locale classiques, notamment leur tendance à rester bloquées dans des optimums locaux [10].

Contrairement à une méthode de descente classique, la recherche Tabou peut accepter un mouvement qui ne conduit pas nécessairement à une amélioration immédiate de la fonction objectif. Cette propriété lui permet de quitter un optimum local et d'explorer de nouvelles régions de l'espace des solutions. Son fonctionnement repose principalement sur la définition d'un voisinage, la sélection des mouvements, l'utilisation d'une mémoire Tabou et, éventuellement, l'application d'un critère d'aspiration.

Soit X une solution courante et $N(X)$ son voisinage. À chaque itération, la procédure examine un ensemble de solutions $X' \in N(X)$ obtenues par l'application de mouvements locaux. Dans un problème d'ordonnancement par permutation, ces mouvements peuvent notamment correspondre à :

- l'échange de deux tâches (*swap*);
- l'insertion d'une tâche dans une nouvelle position;
- l'inversion d'une sous-séquence;
- le déplacement ou la permutation d'un bloc de tâches.

La solution courante est ensuite remplacée par la meilleure solution admissible du voisinage selon la stratégie retenue. Cette solution peut être moins bonne que la solution courante, ce qui distingue la recherche Tabou d'une simple procédure de descente et contribue à éviter une stagnation prématurée.

Le mécanisme de mémoire constitue l'un des éléments centraux de la méthode. Les attributs associés aux mouvements récemment effectués sont enregistrés dans une liste Tabou pendant un nombre déterminé d'itérations, appelé durée Tabou ou *tabu tenure*. Pendant cette période, les mouvements concernés sont temporairement interdits afin d'éviter les cycles et les retours immédiats vers des configurations récemment visitées [10].

Cette interdiction n'est cependant pas nécessairement absolue. Un *critère d'aspiration* peut autoriser un mouvement Tabou lorsque celui-ci conduit à une solution présentant un intérêt particulier. Un critère classique consiste à lever le statut Tabou lorsqu'un mouvement permet d'obtenir une solution meilleure que la meilleure solution globale trouvée depuis le début de la recherche.

Dans le problème étudié, une solution peut être représentée par une permutation :

$$X = (J_{\pi(1)}, J_{\pi(2)}, \dots, J_{\pi(n)}),$$

et sa qualité est évaluée à partir de la fonction objectif :

$$F(X) = \sum_{i=1}^n w_i C_i.$$

Comme pour l’algorithme génétique, le calcul des dates d’achèvement C_i doit tenir compte des périodes d’indisponibilité de la machine. Toute modification de la séquence doit donc être suivie d’une reconstruction temporelle permettant de garantir la faisabilité de l’ordonnancement et d’évaluer correctement la fonction objectif.

L’efficacité de la recherche Tabou dépend fortement de plusieurs choix méthodologiques, notamment la structure du voisinage, la durée Tabou, le critère d’aspiration, la stratégie de sélection des mouvements et le critère d’arrêt. Un voisinage trop restreint peut limiter la capacité d’exploration, tandis qu’un voisinage très large peut augmenter considérablement le coût computationnel de chaque itération.

L’intérêt de la recherche Tabou pour les problèmes d’ordonnancement soumis à des contraintes de disponibilité est documenté dans la littérature.

Plus récemment, Lv et al. (2024) ont étudié un problème d’ordonnancement sur machine unique avec dates de disponibilité et durées de traitement proportionnellement détériorantes, sous le critère de minimisation de la somme pondérée des dates d’achèvement. Les auteurs considèrent notamment une méthode Branch-and-Bound, une recherche Tabou, un recuit simulé et une heuristique, illustrant l’utilisation de plusieurs paradigmes de résolution pour une variante complexe du problème de machine unique [18].

Dans le contexte spécifique de cette recherche, Lounissi et al. (2025) ont comparé une recherche Tabou et un algorithme génétique pour un problème d’ordonnancement sur machine unique visant à minimiser la somme pondérée des dates d’achèvement [24]. Cette comparaison constitue un point d’appui pertinent pour l’étude développée dans cette thèse.

La recherche Tabou présente ainsi un intérêt particulier pour le problème considéré. Alors que l’algorithme génétique exploite simultanément une population de solutions afin de favoriser une exploration relativement globale, la recherche Tabou intensifie principalement l’exploration autour d’une solution courante tout en utilisant des mécanismes de mémoire pour limiter les cycles et favoriser la diversification. Ces caractéristiques justifient l’étude comparative des deux approches dans la suite de cette thèse.

2.4.3 Recuit simulé

Le recuit simulé, ou *Simulated Annealing* (SA), est une métaheuristique inspirée du processus physique de recuit des matériaux. Son adaptation aux problèmes d’optimisation combinatoire a notamment été formalisée par Kirkpatrick et al. [25]. Son principe consiste à autoriser, sous certaines conditions, l’acceptation temporaire de solutions moins bonnes que la solution courante afin de réduire le risque de convergence prématurée vers un optimum local.

Soit X une solution courante et X' une solution voisine. Pour un problème de minimisation, on définit :

$$\Delta = F(X') - F(X).$$

Lorsque $\Delta \leq 0$, la nouvelle solution améliore ou conserve la valeur de la fonction objectif et peut être acceptée. Lorsque $\Delta > 0$, la solution dégradante peut être acceptée avec une probabilité généralement définie par :

$$P(\text{acceptation}) = \exp\left(-\frac{\Delta}{T}\right),$$

où $T > 0$ représente la température courante.

Au début de la recherche, une température relativement élevée augmente la probabilité d'accepter des solutions dégradantes et favorise ainsi l'exploration de l'espace des solutions. La température diminue ensuite progressivement selon un calendrier de refroidissement. À mesure que T diminue, les mouvements dégradants deviennent moins susceptibles d'être acceptés et la recherche tend progressivement vers un comportement davantage orienté vers l'intensification.

La performance du recuit simulé dépend notamment de la température initiale, de la loi de refroidissement, du nombre d'itérations effectuées à chaque niveau de température, de la structure du voisinage et du critère d'arrêt.

Cette approche a également été appliquée à des problèmes d'ordonnancement comportant des contraintes particulières. Plus récemment, Lv et al. (2024) ont considéré le recuit simulé dans une étude portant sur la minimisation de la somme pondérée des dates d'achèvement sur machine unique, aux côtés d'une recherche Tabou et d'une méthode exacte [18].

Bien que le recuit simulé ne constitue pas l'approche principale retenue dans cette thèse, sa présentation permet de situer l'algorithme génétique et la recherche Tabou par rapport à une autre stratégie classique de recherche capable d'accepter temporairement des mouvements dégradants afin d'échapper aux optimums locaux.

2.4.4 Approches hybrides et méthodes combinées

Les approches hybrides combinent plusieurs mécanismes de résolution afin d'exploiter leurs propriétés complémentaires. Elles peuvent notamment associer une métaheuristique à une recherche locale, combiner plusieurs métaheuristiques ou intégrer une procédure heuristique au sein d'une méthode exacte.

L'objectif de l'hybridation est généralement d'obtenir un meilleur équilibre entre intensification et diversification. Une méthode d'exploration globale peut, par exemple, être associée à une procédure de recherche locale chargée d'améliorer les solutions prometteuses. Inversement, une méthode exacte peut être alimentée par des solutions heuristiques de bonne qualité afin d'obtenir rapidement de meilleures bornes supérieures.

Les travaux consacrés aux métaheuristiques hybrides en ordonnancement montrent l'intérêt de combiner des mécanismes complémentaires d'exploration, d'intensification et de gestion de la diversité [19].

Les approches hybrides présentent ainsi un intérêt pour les problèmes d'ordonnancement complexes, dans la mesure où elles permettent d'associer plusieurs mécanismes de recherche au sein d'une même procédure. Leur efficacité dépend néanmoins de la manière dont les différentes composantes sont intégrées ainsi que du compromis recherché entre qualité des solutions et coût computationnel.

Ces développements montrent que la frontière entre les différentes familles de métaheuristiques n'est pas toujours stricte. Néanmoins, dans cette thèse, l'algorithme génétique et la recherche Tabou sont étudiés séparément afin de permettre une comparaison claire de leurs comportements respectifs avant d'envisager, éventuellement, des stratégies d'hybridation.

2.5 Comparaison des méthodes de résolution

L'analyse comparative des différentes méthodes montre qu'aucune approche ne peut être considérée comme universellement supérieure pour l'ensemble des problèmes d'ordonnancement. Le choix d'une méthode dépend notamment de la structure du problème, de la taille des instances, des contraintes considérées, du critère d'optimisation et du niveau de garantie recherché.

Les méthodes exactes sont particulièrement adaptées lorsque la certification de l'optimalité constitue une exigence essentielle et que la structure et la taille des instances permettent une résolution complète. Leur coût computationnel peut toutefois augmenter fortement avec la difficulté du problème.

À l'inverse, les heuristiques et les métaheuristiques offrent une plus grande flexibilité pour traiter des instances complexes. Les approches hybrides présentent également un intérêt, puisqu'elles permettent de combiner des mécanismes complémentaires afin d'améliorer l'équilibre entre intensification, diversification et coût computationnel.

Dans le cas des problèmes d'ordonnancement intégrant des périodes d'indisponibilité des machines, la difficulté de résolution est renforcée par les contraintes temporelles supplémentaires imposées au positionnement des tâches. Ces caractéristiques justifient l'étude de méthodes capables de construire des ordonnancements réalisables tout en recherchant un compromis satisfaisant entre qualité des solutions et temps de calcul [6, 16].

2.6 Analyse critique des travaux antérieurs

L'analyse des travaux antérieurs met en évidence la diversité des approches développées pour résoudre les problèmes d'ordonnancement. Les méthodes exactes constituent un outil essentiel

lorsqu'une certification de l'optimalité est recherchée. Leur efficacité dépend néanmoins de la structure du problème, de la formulation adoptée et de la taille des instances [2, 3].

Les approches heuristiques permettent, quant à elles, de construire rapidement des solutions réalisables en exploitant certaines propriétés du problème. Leur performance dépend cependant fortement de la règle utilisée et des caractéristiques de l'instance considérée.

Les métaheuristiques offrent des mécanismes d'exploration plus généraux et sont particulièrement intéressantes lorsque l'espace des solutions devient difficile à explorer par des méthodes exactes. Leur efficacité dépend néanmoins de plusieurs éléments, notamment la représentation des solutions, la structure du voisinage, les mécanismes d'intensification et de diversification ainsi que le réglage des paramètres [10, 11].

Dans le cas particulier des problèmes intégrant des périodes d'indisponibilité des machines, la littérature montre que ces contraintes modifient sensiblement la structure du problème et peuvent nécessiter des méthodes de résolution spécifiquement adaptées [6, 7, 13]. Les travaux plus récents confirment également l'intérêt porté aux problèmes de machine unique comportant des périodes d'indisponibilité et à leurs différentes extensions [16].

Le tableau 2.1 présente une synthèse critique de travaux et références représentatifs mobilisés dans ce chapitre, en mettant en évidence le problème ou le cadre étudié, l'approche considérée ainsi que les principales limites au regard du problème traité dans cette thèse.

TAB. 2.1 : Synthèse critique de travaux et références représentatifs

Référence	Problème ou cadre étudié	Approche considérée	Apport et limite au regard de cette thèse
Graham et al. (1979)	Classification des problèmes d'ordonnancement	Cadre théorique et notation des problèmes d'ordonnancement	Référence fondamentale pour la classification, mais ne traite pas spécifiquement les périodes d'indisponibilité étudiées ici
Brucker (2007)	Problèmes classiques d'ordonnancement	Méthodes exactes et algorithmiques	Cadre méthodologique général, nécessitant une adaptation aux contraintes spécifiques d'indisponibilité
Ma et al. (2010)	Ordonnancement avec contraintes déterministes de disponibilité	Revue des méthodes et des résultats de complexité	Synthèse de référence sur les contraintes de disponibilité, couvrant plusieurs environnements et critères d'ordonnancement
Kacem et al. (2008)	Machine unique avec contrainte de disponibilité et somme pondérée des dates d'achèvement	Méthodes de résolution adaptées au problème	Problème particulièrement proche de celui étudié dans cette thèse, mais considéré dans un cadre spécifique de disponibilité
Pinedo (2022)	Théorie et applications de l'ordonnancement	Méthodes exactes, heuristiques et approchées	Cadre général récent de référence, mais non centré sur la variante particulière étudiée dans cette thèse
Shabtay (2022)	Machine unique avec périodes d'indisponibilité	Analyse algorithmique d'une variante avec temps de traitement dépendant des ressources	Confirme l'intérêt récent pour les contraintes d'indisponibilité, mais considère une extension différente du modèle étudié ici
Lounissi et al. (2025)	Ordonnancement sur machine unique	Algorithme génétique et recherche Tabou	Comparaison directe de deux métaheuristiques pertinentes pour cette recherche, mais limitée au cadre expérimental et aux approches étudiées

Cette synthèse met en évidence la complémentarité des différentes approches. Les méthodes

exactes permettent de certifier l'optimalité lorsque leur résolution peut être menée à terme, tandis que les heuristiques et les métaheuristiques offrent davantage de flexibilité pour explorer des instances plus complexes.

Elle montre également que les contraintes d'indisponibilité ne peuvent être considérées comme un simple ajout au problème classique, puisqu'elles interagissent directement avec le positionnement temporel des tâches et, par conséquent, avec leurs dates d'achèvement. Pour le critère $\sum_{i=1}^n w_i C_i$, cette interaction peut modifier la qualité des séquences obtenues à partir des règles classiques de priorité [6, 7].

Le positionnement adopté dans cette thèse consiste ainsi à étudier le problème de machine unique avec périodes déterministes d'indisponibilité en comparant deux stratégies métaheuristiques complémentaires : l'algorithme génétique et la recherche Tabou. L'objectif n'est pas de supposer *a priori* la supériorité d'une méthode, mais d'évaluer expérimentalement leurs comportements respectifs en termes de qualité des solutions et de temps de calcul.

2.7 Conclusion

Ce chapitre a présenté et analysé les principales familles de méthodes de résolution mobilisées dans le domaine de l'ordonnancement. Les méthodes exactes, les heuristiques et les métaheuristiques ont été examinées en mettant en évidence leurs principes de fonctionnement, leurs avantages, leurs limites ainsi que leur pertinence au regard du problème étudié dans cette thèse.

Les méthodes exactes, notamment l'énumération complète, la programmation dynamique, le Branch-and-Bound et la programmation mathématique, permettent de déterminer et de certifier une solution optimale lorsque la résolution peut être menée à son terme. Leur efficacité dépend toutefois fortement de la structure du problème, de la formulation retenue et de la taille des instances. L'augmentation du nombre de tâches et l'introduction de contraintes supplémentaires peuvent accroître sensiblement l'effort computationnel nécessaire à leur résolution [2, 3].

Les méthodes heuristiques constituent une alternative permettant de construire rapidement des solutions réalisables en exploitant certaines propriétés structurelles du problème. Dans le cas de la minimisation de la somme pondérée des dates d'achèvement sur une machine unique disponible en permanence, la règle WSPT fournit une solution optimale en ordonnant les tâches par ordre croissant du rapport p_i/w_i , ou, de manière équivalente, par ordre décroissant de w_i/p_i [3, 8]. Toutefois, cette propriété d'optimalité ne s'étend pas directement au problème considéré dans cette thèse lorsque des périodes d'indisponibilité sont introduites [6, 7].

Les métaheuristiques offrent alors un cadre particulièrement intéressant pour explorer efficacement l'espace des solutions sans procéder à une énumération exhaustive. Leur performance repose notamment sur la capacité à équilibrer l'intensification de la recherche autour des solutions prometteuses et la diversification vers de nouvelles régions de l'espace de recherche [10, 11]. Les travaux consacrés aux problèmes d'ordonnancement avec contraintes de

disponibilité confirment l'intérêt de développer des stratégies adaptées à la structure particulière de ces problèmes [6, 16].

Dans cette perspective, deux métaheuristiques complémentaires ont été retenues dans cette recherche : l'algorithme génétique et la recherche Tabou. L'algorithme génétique repose sur l'évolution d'une population de solutions et favorise ainsi l'exploration de différentes régions de l'espace de recherche. La recherche Tabou exploite, quant à elle, une recherche locale guidée par des mécanismes de mémoire permettant d'intensifier l'exploration tout en limitant les cycles. Leur comparaison présente donc un intérêt méthodologique pour évaluer deux stratégies de recherche différentes appliquées au même problème.

Le problème considéré dans cette thèse, noté selon la convention adoptée $1|h_k|\sum_{i=1}^n w_i C_i$, associe un critère classique de somme pondérée des dates d'achèvement à des périodes déterministes d'indisponibilité de la machine. Sa résolution nécessite non seulement de déterminer un ordre pertinent des tâches, mais également de garantir la faisabilité temporelle de l'ordonnement par rapport aux intervalles d'indisponibilité.

Le chapitre suivant sera consacré à la formalisation du problème étudié et à la présentation détaillée des approches de résolution retenues. Il précisera les hypothèses et les notations du modèle, la formulation mathématique exacte, la construction d'une solution initiale fondée sur la règle WSPT, ainsi que l'adaptation de l'algorithme génétique et de la recherche Tabou aux contraintes d'indisponibilité considérées.

Chapitre 3

Modélisation mathématique exacte et résolution du problème d'ordonnancement par des approches métaheuristiques

3.1 Introduction

Les problèmes d'ordonnancement constituent une classe importante des problèmes d'optimisation combinatoire en raison de leur complexité et de leurs nombreuses applications dans les systèmes industriels. La présence de contraintes réalistes, telles que les périodes d'indisponibilité des machines, augmente davantage la difficulté de résolution en réduisant l'ensemble des séquences admissibles et en élargissant l'espace de recherche.

Dans ce travail, le problème considéré concerne l'ordonnancement de tâches indépendantes et non préemptives sur une machine unique soumise à des périodes d'indisponibilité connues à l'avance. L'objectif consiste à déterminer une séquence réalisable des tâches permettant de minimiser la somme pondérée des dates d'achèvement :

$$\min Z = \sum_{i=1}^n w_i C_i$$

où C_i représente la date d'achèvement de la tâche i et w_i son poids associé.

Ce problème appartient à la classe des problèmes d'optimisation combinatoire difficiles [1, 4]. En effet, le nombre de séquences possibles pour n tâches peut atteindre $n!$, ce qui rend l'exploration exhaustive de l'espace des solutions rapidement impraticable pour les instances de grande taille [3].

Afin de définir précisément le problème étudié, une formulation mathématique exacte sous forme d'un programme linéaire en nombres entiers mixtes (PLNE) est d'abord présentée. Cette formulation permet de représenter les contraintes de séquencement des tâches ainsi que les contraintes temporelles liées aux périodes d'indisponibilité de la machine.

Cependant, malgré leur capacité à fournir des solutions optimales, les méthodes exactes deviennent difficiles à appliquer lorsque la taille des instances augmente en raison de la complexité combinatoire du problème. Les approches métaheuristiques constituent alors une alternative efficace grâce à leur capacité d'exploration de grands espaces de recherche dans un temps de calcul raisonnable [11].

Dans ce chapitre, deux approches métaheuristiques sont étudiées et adaptées au problème considéré : l'algorithme génétique (*Genetic Algorithm*, GA) et la recherche Tabou (*Tabu Search*, TS). Après la présentation de la formulation mathématique exacte, les principes et les mécanismes de résolution des deux approches sont détaillés. Enfin, une analyse expérimentale comparative est réalisée afin d'évaluer leurs performances en termes de qualité des solutions obtenues et de temps de calcul.

3.2 Modélisation mathématique exacte du problème

Le problème étudié concerne l'ordonnancement de n tâches indépendantes et non préemptives sur une machine unique soumise à des périodes d'indisponibilité connues à l'avance.

Chaque tâche doit être exécutée exactement une seule fois et ne peut pas être interrompue durant son traitement. La machine ne peut traiter qu'une seule tâche à la fois.

Les tâches sont disponibles dès le début de l'horizon d'ordonnancement et sont caractérisées par leur durée de traitement p_i et leur poids associé w_i .

Cette formulation mathématique exacte permet de représenter précisément le problème étudié en intégrant les contraintes de séquençement des tâches ainsi que les contraintes temporelles liées aux périodes d'indisponibilité de la machine.

3.2.1 Notations et paramètres

L'ensemble des tâches est défini par :

$$J = \{1, \dots, n\}$$

Les paramètres du problème sont définis comme suit :

- p_i : durée de traitement de la tâche i ;
- w_i : poids associé à la tâche i ;
- $[a_k, b_k]$: intervalle correspondant à la k -ème période d'indisponibilité de la machine ;
- M : constante suffisamment grande utilisée dans la formulation linéaire du modèle.

3.2.2 Variables de décision

Les variables de décision sont définies comme suit :

- S_i : date de début d'exécution de la tâche i ;
- C_i : date d'achèvement de la tâche i ;
- x_{ij} : variable binaire indiquant que la tâche i précède la tâche j ;
- y_{ik} : variable binaire indiquant la position de la tâche i par rapport à la période d'indisponibilité k .

La relation entre la date de début et la date d'achèvement est donnée par :

$$C_i = S_i + p_i$$

Cette relation permet de déterminer la date d'achèvement de chaque tâche à partir de sa date de début et de sa durée de traitement.

3.2.3 Variables binaires de séquençement

Afin de représenter les relations de précédence entre les tâches, une variable binaire de séquençement est introduite :

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

avec :

$$i, j \in \{1, \dots, n\}, \quad i \neq j$$

La contrainte de précédence est définie par :

$$x_{ij} + x_{ji} = 1 \quad \forall i, j, i \neq j$$

Cette contrainte garantit qu'entre deux tâches distinctes, une seule relation de précédence est possible.

3.2.4 Fonction objectif

L'objectif du problème consiste à minimiser la somme pondérée des dates d'achèvement des tâches. La fonction objectif est formulée comme suit :

$$\min Z = \sum_{i=1}^n w_i C_i$$

où C_i représente la date d'achèvement de la tâche i et w_i son poids associé.

Cette fonction permet de privilégier l'ordonnancement des tâches ayant les poids les plus élevés, car leur contribution au critère d'optimisation est plus importante.

3.2.5 Contraintes du modèle

Le modèle mathématique doit respecter plusieurs contraintes permettant de garantir la faisabilité de l'ordonnancement obtenu.

3.2.6 Contraintes du modèle

Le modèle mathématique doit respecter plusieurs contraintes permettant de garantir la faisabilité de l'ordonnancement obtenu tout en considérant les contraintes liées au séquençement des tâches et aux périodes d'indisponibilité de la machine.

3.2.6.1 Contrainte de précedence entre les tâches

Afin d'imposer un ordre unique entre deux tâches distinctes, la contrainte de précedence suivante est introduite :

$$x_{ij} + x_{ji} = 1 \quad \forall i, j = 1, \dots, n, i \neq j$$

Cette contrainte garantit qu'entre deux tâches différentes, une seule relation de précedence est possible : soit la tâche i précède la tâche j , soit la tâche j précède la tâche i .

Ainsi, chaque solution obtenue correspond à une séquence complète d'ordonnancement des tâches.

3.2.6.2 Contraintes de non-chevauchement des tâches

Dans le cas d'une machine unique, deux tâches ne peuvent pas être exécutées simultanément.

Ainsi, si la tâche i précède la tâche j , la contrainte suivante est imposée :

$$S_j \geq C_i - M(1 - x_{ij}) \quad \forall i, j, i \neq j$$

où M représente une constante suffisamment grande.

Lorsque :

$$x_{ij} = 1$$

la contrainte devient :

$$S_j \geq C_i$$

ce qui signifie que la tâche j ne peut commencer qu'après l'achèvement de la tâche i .
Lorsque :

$$x_{ij} = 0$$

la contrainte est désactivée grâce au terme de relaxation M .

3.2.6.3 Contraintes liées aux périodes d'indisponibilité

La machine considérée peut être soumise à des périodes d'indisponibilité connues à l'avance notées :

$$[a_k, b_k]$$

Durant ces intervalles, aucune tâche ne doit être interrompue. Chaque tâche doit donc être exécutée entièrement avant la période d'indisponibilité ou après la remise en disponibilité de la machine.

Une variable binaire supplémentaire y_{ik} est introduite afin de définir la position de la tâche i par rapport à la période d'indisponibilité k :

$$y_{ik} = \begin{cases} 1, & \text{si la tâche } i \text{ est exécutée avant la période } k, \\ 0, & \text{si la tâche } i \text{ est exécutée après cette période.} \end{cases}$$

Les contraintes associées sont données par :

$$C_i \leq a_k + M(1 - y_{ik}) \quad \forall i, k$$

et

$$S_i \geq b_k - My_{ik} \quad \forall i, k$$

avec :

$$y_{ik} \in \{0, 1\} \quad \forall i, k$$

Lorsque :

$$y_{ik} = 1$$

la première contrainte impose :

$$C_i \leq a_k$$

ce qui signifie que la tâche doit être terminée avant le début de la période d'indisponibilité.

Lorsque :

$$y_{ik} = 0$$

la seconde contrainte impose :

$$S_i \geq b_k$$

ce qui signifie que la tâche doit commencer après la remise en disponibilité de la machine.

Ces contraintes garantissent donc le caractère non préemptif des tâches en présence des périodes d'arrêt de la machine.

3.2.6.4 Contraintes de domaine des variables

Les variables binaires de séquençement et de positionnement doivent respecter les conditions suivantes :

$$x_{ij} \in \{0, 1\} \quad \forall i, j, i \neq j$$

$$y_{ik} \in \{0, 1\} \quad \forall i, k$$

Les dates d'exécution doivent également être positives :

$$S_i \geq 0, \quad C_i \geq 0 \quad \forall i = 1, \dots, n$$

Ces contraintes complètent la formulation mathématique du problème et assurent que les solutions obtenues respectent l'ensemble des conditions physiques et temporelles de l'ordonnancement.

3.2.7 Formulation finale du modèle PLNE

En regroupant la fonction objectif et l'ensemble des contraintes précédentes, le modèle mathématique exact du problème d'ordonnancement étudié peut être formulé sous la forme d'un programme linéaire en nombres entiers mixtes (PLNE) comme suit :

$$\min Z = \sum_{i=1}^n w_i C_i$$

sous les contraintes :

$$C_i = S_i + p_i \quad \forall i = 1, \dots, n$$

$$x_{ij} + x_{ji} = 1 \quad \forall i, j, i \neq j$$

$$S_j \geq C_i - M(1 - x_{ij}) \quad \forall i, j, i \neq j$$

Les contraintes relatives aux périodes d'indisponibilité de la machine sont exprimées par :

$$C_i \leq a_k + M(1 - y_{ik}) \quad \forall i, k$$

$$S_i \geq b_k - My_{ik} \quad \forall i, k$$

où y_{ik} est une variable binaire indiquant la position de la tâche i par rapport à la période d'indisponibilité k .

Les contraintes de domaine des variables sont définies par :

$$x_{ij} \in \{0, 1\} \quad \forall i, j, i \neq j$$

$$y_{ik} \in \{0, 1\} \quad \forall i, k$$

$$S_i \geq 0, \quad C_i \geq 0 \quad \forall i = 1, \dots, n$$

où M représente une constante suffisamment grande permettant de linéariser les contraintes de précédence et les contraintes liées aux périodes d'indisponibilité.

Ce modèle constitue une formulation exacte du problème étudié. Il permet de représenter l'ensemble des solutions réalisables et d'obtenir théoriquement une solution optimale. Cependant, en raison de la complexité combinatoire du problème, la résolution exacte devient rapidement difficile lorsque la taille des instances augmente. Cette limite justifie l'utilisation d'approches métaheuristiques capables de fournir des solutions de haute qualité dans un temps de calcul raisonnable.

Afin d'initialiser les méthodes d'amélioration proposées, une solution initiale réalisable est ensuite construite à l'aide d'une règle heuristique d'ordonnancement.

3.3 Génération de la solution initiale par la règle WSPT

Avant l'application des méthodes métaheuristiques proposées, une solution initiale réalisable respectant les contraintes du problème est générée afin de fournir un point de départ efficace pour le processus de recherche.

Dans cette étude, la solution initiale est obtenue à l'aide de la règle WSPT (*Weighted Shortest Processing Time*), une règle heuristique classique d'ordonnancement basée sur le rapport entre

la durée de traitement et le poids associé à chaque tâche [8].

Pour chaque tâche i , le critère de priorité suivant est calculé :

$$r_i = \frac{p_i}{w_i}$$

où p_i représente la durée de traitement de la tâche i et w_i son poids associé.

Les tâches sont ensuite classées selon un ordre croissant de la valeur r_i . Ainsi, les tâches présentant les plus faibles rapports

$$\frac{p_i}{w_i}$$

sont exécutées en priorité dans la séquence d'ordonnancement.

Dans le problème étudié, la règle WSPT ne garantit pas nécessairement l'obtention de la solution optimale en raison de la présence des périodes d'indisponibilité de la machine. Elle est donc utilisée comme une procédure de construction permettant de générer une solution initiale de bonne qualité avant l'application des méthodes métaheuristiques.

La séquence obtenue par cette règle constitue la solution de départ utilisée par l'algorithme génétique et la recherche Tabou. Dans le cas de l'algorithme génétique, cette solution peut également être intégrée directement dans la population initiale afin d'améliorer la qualité de la phase d'initialisation.

3.4 Algorithme génétique adapté au problème étudié

Les algorithmes génétiques constituent une famille de méthodes métaheuristiques inspirées des mécanismes de sélection naturelle et de l'évolution biologique. Ils permettent d'explorer efficacement des espaces de recherche complexes en manipulant une population de solutions candidates.

Dans ce travail, un algorithme génétique est développé pour résoudre le problème d'ordonnancement étudié. L'adaptation proposée repose sur une représentation par permutation des tâches, une fonction d'évaluation prenant en compte les périodes d'indisponibilité de la machine, ainsi que des opérateurs génétiques adaptés au problème de séquençement.

3.4.1 Représentation des solutions (chromosome)

Dans l'algorithme génétique proposé, chaque solution candidate est représentée par un chromosome sous forme d'une permutation des tâches à ordonnancer.

Un chromosome correspond à une séquence contenant l'ensemble des tâches du problème, chaque tâche apparaissant exactement une seule fois. Cette représentation permet de respecter naturellement la contrainte d'exécution unique des tâches.

Pour un problème comportant n tâches, un chromosome peut être représenté par :

$$X = (J_1, J_2, \dots, J_i, \dots, J_n)$$

où J_i représente la tâche située à la position i dans la séquence d'ordonnancement.

Par exemple, pour un problème comportant cinq tâches, un chromosome possible est :

$$X = (J_3, J_1, J_5, J_2, J_4)$$

Cette représentation par permutation facilite l'application des opérateurs génétiques, notamment le croisement et la mutation, tout en conservant la faisabilité des solutions générées.

Lors de l'évaluation d'un chromosome, la séquence obtenue est décodée afin de calculer les dates de début S_i et d'achèvement C_i des tâches en tenant compte des périodes d'indisponibilité de la machine. La valeur de la fonction objectif associée à chaque chromosome est ensuite utilisée comme critère d'évaluation pour guider l'évolution de la population.

3.4.2 Initialisation de la population

La génération de la population initiale constitue une étape importante dans l'algorithme génétique, car la qualité et la diversité des individus influencent directement la capacité d'exploration de l'espace des solutions.

Dans l'approche proposée, le premier individu de la population correspond à la séquence obtenue par la règle WSPT présentée précédemment. Cette stratégie permet d'introduire une solution initiale de bonne qualité dans la population.

Afin d'éviter une convergence prématurée vers une région limitée de l'espace de recherche, les autres individus sont générés par des permutations aléatoires de cette séquence initiale afin d'introduire une diversité suffisante dans la population.

3.4.3 Fonction d'évaluation (fitness)

Chaque chromosome de la population est évalué afin de déterminer sa qualité par rapport au critère d'optimisation du problème.

Pour chaque chromosome, une procédure de décodage est appliquée afin de déterminer les dates de début S_i et les dates d'achèvement C_i des tâches. Lors de ce calcul, les périodes d'indisponibilité de la machine sont prises en compte afin d'obtenir une séquence réalisable respectant les contraintes du problème.

La qualité de chaque solution est évaluée par la fonction objectif définie comme suit :

$$F(X) = \sum_{i=1}^n w_i C_i$$

où C_i représente la date d'achèvement de la tâche i et w_i son poids associé.

L'algorithme génétique cherche à minimiser cette fonction objectif. Ainsi, les chromosomes présentant les plus faibles valeurs de $F(X)$ correspondent aux meilleures solutions et possèdent une probabilité plus élevée d'être sélectionnés pour les opérations de reproduction.

Cette fonction d'évaluation permet donc de mesurer la performance de chaque individu et de guider l'évolution de la population au cours des générations.

3.4.4 Opérateur de sélection

L'opérateur de sélection permet de choisir les individus qui participeront à la création de la nouvelle génération. Son objectif est de favoriser les chromosomes présentant les meilleures performances tout en maintenant une diversité suffisante au sein de la population.

Dans cette étude, une sélection par tournoi est utilisée. Cette méthode consiste à sélectionner aléatoirement un nombre fixé d'individus dans la population, puis à comparer leurs valeurs de fonction objectif.

L'individu présentant la meilleure performance, c'est-à-dire la plus faible valeur de :

$$F(X) = \sum_{i=1}^n w_i C_i$$

est sélectionné pour participer aux opérations de reproduction.

La sélection par tournoi présente l'avantage d'être simple à mettre en œuvre et d'assurer un compromis entre la pression de sélection et la préservation de la diversité génétique. Une pression de sélection trop élevée peut entraîner une convergence prématurée vers des solutions sous-optimales, tandis qu'une pression trop faible peut ralentir l'amélioration de la population.

3.4.5 Opérateur de croisement

L'opérateur de croisement permet de générer de nouveaux individus en combinant les caractéristiques de deux chromosomes parents. Dans le cas des problèmes d'ordonnancement représentés par des permutations, cet opérateur doit préserver la validité des chromosomes, c'est-à-dire éviter la répétition ou l'absence de tâches dans les solutions générées.

Dans cette étude, l'opérateur de croisement OX (*Order Crossover*) est utilisé. Cet opérateur est particulièrement adapté aux représentations par permutation, car il permet de conserver l'ordre relatif des tâches tout en créant de nouvelles séquences.

Le principe du croisement OX consiste à sélectionner aléatoirement deux points de coupure dans les chromosomes parents. Le segment situé entre ces deux positions est copié du premier parent vers l'enfant. Les positions restantes sont ensuite complétées à partir du deuxième parent en parcourant les tâches selon leur ordre d'apparition et en éliminant celles déjà présentes dans le segment conservé.

Par exemple, pour deux chromosomes parents :

$$P_1 = (J_1, J_2, J_3, J_4, J_5, J_6)$$

$$P_2 = (J_4, J_5, J_6, J_1, J_2, J_3)$$

un chromosome enfant est obtenu en conservant une partie du premier parent, puis en complétant les positions restantes avec les tâches issues du deuxième parent tout en respectant leur ordre relatif.

L'utilisation du croisement OX permet ainsi de préserver la structure de permutation des solutions, de maintenir la faisabilité des chromosomes et de générer de nouvelles séquences réalisables pour l'exploration de l'espace de recherche.

3.4.6 Opérateur de mutation

L'opérateur de mutation permet d'introduire de nouvelles variations dans la population afin de maintenir la diversité génétique et d'éviter une convergence prématurée vers des optima locaux.

Dans cette étude, une mutation par insertion (*Insertion Mutation*) est utilisée. Elle consiste à sélectionner aléatoirement une tâche dans le chromosome, à la retirer de sa position initiale, puis à la réinsérer dans une nouvelle position de la séquence.

Pour un chromosome donné :

$$X = (J_1, J_2, \dots, J_i, \dots, J_j, \dots, J_n)$$

la mutation par insertion consiste à extraire la tâche J_i de sa position initiale et à l'insérer à une nouvelle position j , produisant ainsi une nouvelle permutation X' .

Cette opération conserve la validité du chromosome puisque chaque tâche apparaît exactement une seule fois dans la nouvelle solution. Elle permet également d'explorer de nouvelles régions de l'espace de recherche en modifiant localement la structure de la séquence.

La mutation est appliquée selon une probabilité fixée afin d'assurer un équilibre entre l'exploration de nouvelles solutions et l'exploitation des solutions prometteuses déjà présentes dans la population.

3.4.7 Critère d'arrêt

Le processus d'évolution de l'algorithme génétique est arrêté lorsque le nombre maximal de générations fixé est atteint.

Dans cette étude, le critère d'arrêt correspond à l'atteinte de :

$$G_{\max} = 400$$

où $G_{\max}$ représente le nombre maximal de générations autorisées.

Ce paramètre permet de contrôler le temps d'exécution de l'algorithme tout en garantissant un nombre suffisant de générations pour permettre l'amélioration progressive des solutions. À la fin du processus d'évolution, le meilleur chromosome obtenu dans la population finale est retenu comme solution fournie par l'algorithme génétique.

3.4.8 Pseudo-code de l'algorithme génétique

Algorithm 1 Algorithme génétique proposé

Générer la population initiale en intégrant la solution WSPT comme premier individu

Évaluer chaque chromosome selon la fonction objectif

$G = 0$

while $G < G_{\max}$ **do**

 Sélectionner les chromosomes parents par sélection tournoi

 Appliquer le croisement OX

 Appliquer la mutation par insertion

 Construire la nouvelle population

 Évaluer les nouveaux chromosomes

 Conserver la meilleure solution trouvée

$G = G + 1$

end while

Retourner le meilleur chromosome obtenu

3.5 Recherche Tabou adaptée au problème étudié

La recherche Tabou (*Tabu Search*, *TS*) est une méthode métaheuristique basée sur une recherche locale guidée par une mémoire adaptative. Introduite par Glover [10], elle permet d'explorer efficacement des espaces de recherche complexes en évitant les cycles de recherche grâce à l'utilisation d'une liste Tabou qui mémorise temporairement certains mouvements déjà explorés.

Dans cette étude, la recherche Tabou est adaptée au problème d'ordonnancement sur une machine unique avec périodes d'indisponibilité. La méthode repose sur une représentation des solutions par permutation des tâches, une structure de voisinage basée sur l'échange de tâches (*Swap*), une mémoire Tabou et une fonction d'évaluation intégrant les contraintes temporelles du problème.

L'objectif consiste à rechercher une séquence réalisable minimisant la fonction objectif suivante :

$$F(X) = \sum_{i=1}^n w_i C_i$$

tout en respectant les contraintes liées aux périodes d'indisponibilité de la machine.

3.5.1 Représentation d'une solution

Dans la recherche Tabou, une solution correspond à une permutation des tâches représentant leur ordre d'exécution sur la machine unique. Cette représentation est identique à celle utilisée dans l'algorithme génétique afin d'assurer une comparaison équitable entre les deux approches proposées.

Une solution peut être représentée sous la forme :

$$X = (J_1, J_2, \dots, J_n)$$

où chaque élément J_i représente une tâche et sa position dans la séquence détermine son ordre d'exécution.

Par exemple, pour un problème comportant cinq tâches, une solution possible est :

$$X = (J_2, J_5, J_1, J_4, J_3)$$

Cette représentation garantit que chaque tâche apparaît exactement une seule fois dans la solution. Elle permet également l'application des mouvements de recherche locale tout en conservant la faisabilité des séquences générées.

3.5.2 Génération de la solution initiale

Comme pour l'algorithme génétique, la qualité de la solution initiale influence l'efficacité de la recherche Tabou. Dans l'approche proposée, la solution initiale est générée à partir de la séquence obtenue par la règle WSPT.

Cette solution constitue un point de départ réalisable pour l'exploration locale. Elle respecte la séquence obtenue par la règle de priorité WSPT et les dates d'achèvement sont calculées en tenant compte des périodes d'indisponibilité de la machine.

À partir de cette solution initiale, la recherche Tabou explore itérativement son voisinage en appliquant des mouvements de recherche locale afin d'améliorer la valeur de la fonction objectif :

$$F(X) = \sum_{i=1}^n w_i C_i$$

À chaque itération, la meilleure solution admissible du voisinage est sélectionnée tout en respectant les mécanismes de mémoire Tabou.

L'objectif est ainsi d'obtenir une meilleure séquence d'ordonnancement tout en conservant les contraintes du problème étudié.

3.5.3 Structure de voisinage

La définition de la structure de voisinage constitue une étape essentielle dans la recherche Tabou, car elle détermine les mouvements permettant d'explorer les solutions voisines de la solution courante.

Dans le problème d'ordonnancement étudié, le voisinage est construit à partir d'une opération d'échange de deux tâches (*Swap*). Cette opération consiste à sélectionner deux positions différentes dans la séquence et à permuter les tâches correspondantes.

Pour une solution donnée :

$$X = (J_1, J_2, \dots, J_i, \dots, J_j, \dots, J_n)$$

un mouvement d'échange permet d'obtenir une nouvelle solution voisine :

$$X' = (J_1, J_2, \dots, J_j, \dots, J_i, \dots, J_n)$$

Cette structure de voisinage conserve la représentation par permutation des tâches et garantit que chaque solution générée reste réalisable du point de vue du séquençement.

À chaque itération, les solutions voisines générées sont évaluées selon la fonction objectif :

$$F(X) = \sum_{i=1}^n w_i C_i$$

avec prise en compte des périodes d'indisponibilité de la machine lors du calcul des dates d'achèvement.

3.5.4 Exploration du voisinage et choix du mouvement

À chaque itération de la recherche Tabou, l'ensemble des solutions voisines de la solution courante est généré à partir de la structure de voisinage définie précédemment. Chaque solution candidate est ensuite évaluée selon la fonction objectif du problème.

La meilleure solution admissible parmi les solutions voisines est sélectionnée comme nouvelle solution courante. Un mouvement est considéré comme admissible s'il n'appartient pas à la liste Tabou, sauf dans le cas où le critère d'aspiration est satisfait et permet d'accepter une solution interdite présentant une amélioration de la meilleure solution connue.

Le processus de recherche permet ainsi d'améliorer progressivement la qualité des solutions en explorant différentes régions de l'espace de recherche.

Cette stratégie permet d'assurer un compromis entre l'intensification de la recherche autour des meilleures solutions rencontrées et la diversification vers de nouvelles configurations afin d'éviter le blocage dans des optima locaux.

3.5.5 Liste Tabou et critère d'aspiration

La liste Tabou constitue une mémoire à court terme permettant de conserver l'historique des mouvements récemment effectués durant le processus de recherche. Son rôle principal est d'éviter les cycles de recherche et d'empêcher le retour immédiat vers des solutions déjà explorées.

Après l'application d'un mouvement d'échange entre deux tâches, le mouvement correspondant est ajouté temporairement à la liste Tabou. Il devient alors interdit pendant un nombre déterminé d'itérations appelé durée Tabou (*Tabu tenure*).

Dans cette étude, la durée Tabou utilisée dans les expérimentations est fixée à :

$$T_{tenure} = 7$$

itérations.

Cependant, certains mouvements interdits peuvent conduire à une amélioration de la meilleure solution connue. Afin de ne pas éliminer ces solutions potentiellement intéressantes, un critère d'aspiration est introduit.

Il permet d'accepter un mouvement Tabou lorsque la solution obtenue présente une meilleure valeur de la fonction objectif que la meilleure solution enregistrée jusqu'à présent.

Ainsi, la liste Tabou favorise la diversification de la recherche, tandis que le critère d'aspiration maintient la capacité de la méthode à exploiter les solutions prometteuses.

3.5.6 Critère d'arrêt

Comme toute méthode métaheuristique, la recherche Tabou nécessite la définition d'un critère d'arrêt afin de contrôler la durée du processus d'exploration.

Dans cette étude, l'algorithme est arrêté lorsque le nombre maximal d'itérations fixé à l'avance est atteint.

Le nombre maximal d'itérations utilisé dans les expérimentations est :

$$I_{\max} = 400$$

où $I_{\max}$ représente le nombre maximal d'itérations autorisées.

Ce critère permet de limiter le temps de calcul tout en garantissant un nombre suffisant d'itérations pour explorer efficacement l'espace des solutions.

À la fin du processus, la meilleure solution rencontrée au cours de la recherche est retenue comme solution finale fournie par l'algorithme Tabou.

3.5.7 Pseudo-code de la recherche Tabou

Algorithm 2 Recherche Tabou proposée

```
Générer une solution initiale à partir de la règle WSPT
Évaluer la solution initiale
Initialiser la liste Tabou
 $it = 0$ 
while  $it < I_{\max}$  do
    Générer les solutions voisines par échange de deux tâches
    Évaluer les solutions voisines
    Sélectionner la meilleure solution admissible
    Vérifier le critère d'aspiration
    Mettre à jour la solution courante
    Actualiser la liste Tabou
     $it = it + 1$ 
end while
Retourner la meilleure solution obtenue
```

3.5.8 Validation sur une instance illustrative de 20 tâches

Afin d'illustrer le comportement des approches métaheuristiques proposées, une première expérimentation est réalisée sur une instance composée de 20 tâches. Cette instance permet de présenter de manière détaillée le processus de génération de la solution initiale ainsi que l'amélioration obtenue par les deux méthodes proposées.

Le problème considéré comporte deux périodes d'indisponibilité de la machine :

$$[300, 350] \quad \text{et} \quad [700, 750]$$

Durant ces intervalles, aucune tâche ne peut être exécutée. Les caractéristiques des tâches utilisées dans cette expérimentation sont présentées dans le tableau suivant.

TAB. 3.1 : Caractéristiques de l'instance de 20 tâches

Tâche	Durée p_i	Poids w_i	Ratio p_i/w_i
T1	70	9	0.13
T2	59	8	0.14
T3	45	4	0.09
T4	81	2	0.02
T5	79	5	0.06
T6	87	8	0.09
T7	93	3	0.03
T8	45	5	0.11
T9	14	6	0.43
T10	51	1	0.02
T11	23	9	0.39
T12	84	8	0.10
T13	72	1	0.01
T14	43	3	0.07
T15	82	4	0.05
T16	75	6	0.08
T17	87	8	0.09
T18	42	3	0.07
T19	25	4	0.16
T20	68	6	0.09

En appliquant la règle WSPT, la solution initiale obtenue est :

$$X_{WSPT} = (T8, T10, T18, T1, T7, T11, T5, T16, T2, T19, T15, T17, T13, T4, T14, T6, T3, T9, T12, T20)$$

Cette solution fournit une valeur initiale de la fonction objectif :

$$F(X_{WSPT}) = 54616$$

3.5.8.1 Résultat obtenu par la Recherche Tabou

La recherche Tabou est appliquée à partir de la solution WSPT. La durée Tabou utilisée est fixée à :

$$T_{tenure} = 7$$

La méthode explore le voisinage des solutions par des mouvements de permutation afin d'améliorer progressivement la fonction objectif.

Le résultat obtenu est :

$$F(X_{TS}) = 51432$$

soit une amélioration relative de :

$$Amlioration_{TS} = 5.83\%$$

avec un temps de calcul de :

$$32\ ms$$

3.5.8.2 Résultat obtenu par l'Algorithme Génétique

L'algorithme génétique est appliqué avec une population de 50 individus et 400 générations. Les opérateurs utilisés sont :

- sélection par tournoi ;
- croisement Order Crossover (OX) ;
- mutation par insertion.

La meilleure solution obtenue est :

$$F(X_{GA}) = 49202$$

avec une amélioration relative de :

$$Amlioration_{GA} = 9.91\%$$

et un temps de calcul de :

$$138\ ms$$

Cette première expérimentation montre que les deux approches permettent d'améliorer la solution initiale obtenue par la règle WSPT. L'algorithme génétique obtient cependant une meilleure qualité de solution grâce à sa capacité d'exploration globale, tandis que la recherche Tabou présente un avantage en termes de rapidité d'exécution.

3.5.9 Analyse des résultats obtenus

3.5.9.1 Qualité des solutions obtenues

Les performances des méthodes proposées sont évaluées en comparant les valeurs de la fonction objectif obtenues par la solution initiale WSPT, la recherche Tabou et l'algorithme

génétique.

L'amélioration relative obtenue par chaque méthode est calculée selon :

$$Amélioration(\%) = \frac{Obj_{WSPT} - Obj_{methode}}{Obj_{WSPT}} \times 100$$

où Obj_{WSPT} représente la valeur de la fonction objectif obtenue par la solution initiale générée par la règle WSPT et $Obj_{methode}$ la valeur obtenue par l'approche métaheuristique considérée.

TAB. 3.2 : Valeurs moyennes de la fonction objectif et amélioration obtenue par rapport à WSPT

n	Obj. WSPT	Obj. TS	Obj. GA	Amélioration TS (%)	Amélioration GA (%)
50	214685	211282	210420	1.59	1.99
100	810052	806275	802866	0.47	0.89
150	1674147	1671546	1665800	0.16	0.50
200	3172325	3166024	3160422	0.20	0.38
300	6851234	6845949	6837962	0.08	0.19
500	19496480	19483440	19466270	0.07	0.16

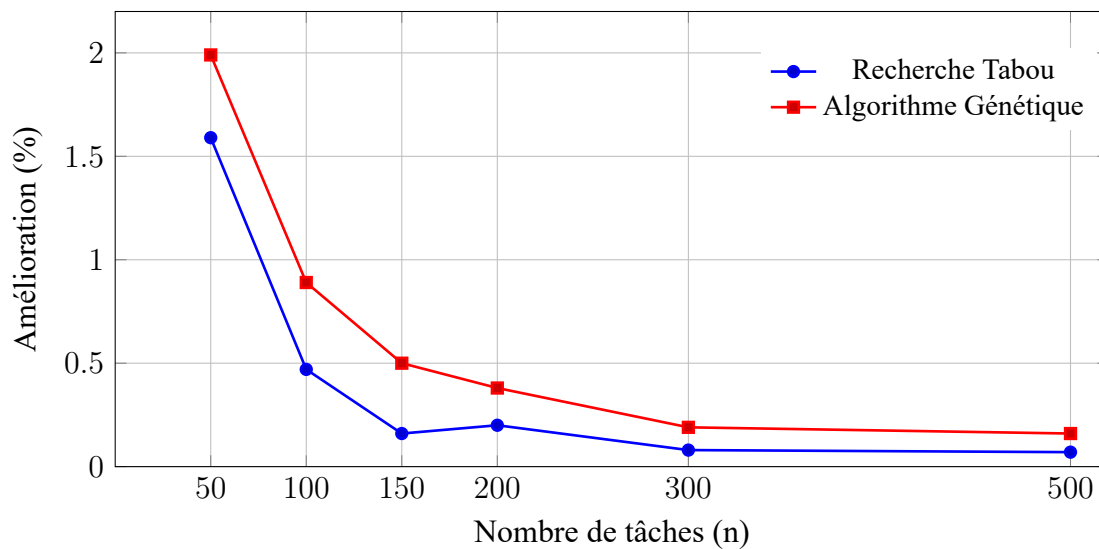


FIG. 3.1 : Amélioration des solutions obtenues par rapport à la solution WSPT

Les résultats présentés dans le tableau 3.2 montrent que les deux approches métaheuristicques permettent d'améliorer la solution initiale obtenue par la règle WSPT.

L'algorithme génétique fournit les meilleures valeurs de la fonction objectif pour l'ensemble des instances testées. Cette performance s'explique par sa capacité à explorer simultanément plusieurs régions de l'espace de recherche grâce à l'utilisation d'une population de solutions et aux opérateurs de croisement et de mutation.

La recherche Tabou améliore également la solution initiale grâce à son mécanisme d'exploration locale guidée par une mémoire adaptative. Toutefois, ses améliorations restent légèrement inférieures à celles obtenues par l'algorithme génétique.

Cependant, l'écart relatif entre les deux approches diminue progressivement lorsque la taille des instances augmente. Cette diminution des gains relatifs s'explique par l'augmentation de la complexité de l'espace de recherche. Lorsque le nombre de tâches devient important, les deux méthodes nécessitent davantage d'exploration afin d'identifier des améliorations supplémentaires par rapport à la solution initiale WSPT.

3.5.9.2 Temps de calcul

En complément de la qualité des solutions obtenues, le temps de calcul constitue un critère essentiel pour évaluer l'efficacité des méthodes proposées.

Le tableau 3.3 présente les temps d'exécution moyens obtenus par la recherche Tabou et l'algorithme génétique pour les différentes tailles d'instances considérées.

TAB. 3.3 : Temps de calcul moyen des algorithmes TS et GA

n	Temps TS (s)	Temps GA (s)
50	0.13	0.47
100	0.60	3.51
150	1.69	12.55
200	3.69	33.64
300	13.35	147.43
500	78.15	1201.96

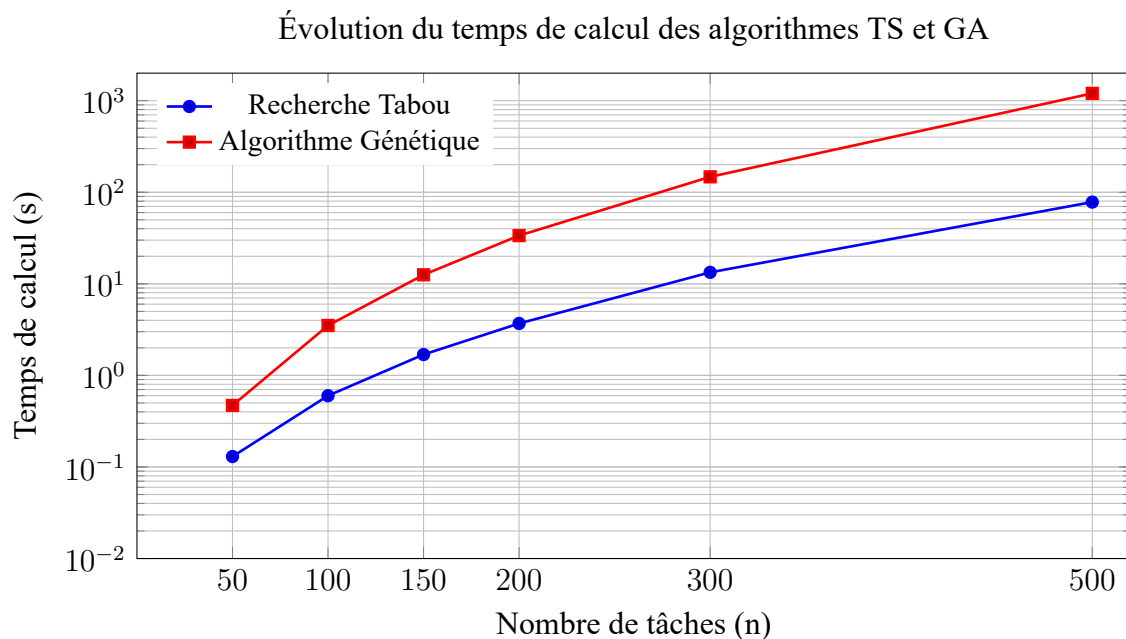


FIG. 3.2 : Évolution du temps de calcul en fonction de la taille des instances

Les résultats montrent que la recherche Tabou présente des temps de calcul inférieurs à ceux de l'algorithme génétique pour toutes les tailles d'instances considérées.

Cette différence s'explique par la nature des deux approches. La recherche Tabou effectue une exploration locale autour d'une solution courante, tandis que l'algorithme génétique nécessite la gestion d'une population complète ainsi que l'application successive des opérateurs de sélection, de croisement et de mutation durant plusieurs générations.

Cette différence devient particulièrement importante pour les grandes tailles d'instances, où le coût computationnel de l'algorithme génétique augmente de manière significative en raison du nombre d'individus évalués à chaque génération.

Ainsi, la recherche Tabou présente un avantage important en termes de rapidité d'exécution, tandis que l'algorithme génétique fournit généralement de meilleures solutions au prix d'un coût computationnel plus élevé.

Ces résultats mettent en évidence un compromis classique entre qualité de solution et temps de calcul : le GA est plus adapté lorsque l'objectif principal est l'amélioration de la fonction objectif, alors que la TS constitue une solution efficace lorsque la rapidité d'obtention d'un planning amélioré est prioritaire.

3.6 Conclusion du chapitre

Ce chapitre a présenté la modélisation mathématique exacte ainsi que les approches méta-heuristiques développées pour résoudre le problème d'ordonnancement étudié sur une machine unique avec périodes d'indisponibilité.

Dans un premier temps, une formulation mathématique sous forme d'un programme linéaire en nombres entiers mixtes (PLNE) a été proposée. Cette formulation permet de représenter précisément les contraintes de séquençement des tâches, les contraintes de non-chevauchement ainsi que les contraintes liées aux périodes d'indisponibilité de la machine, tout en considérant l'objectif de minimisation de la somme pondérée des dates d'achèvement.

Ensuite, une solution initiale réalisable a été générée à l'aide de la règle heuristique WSPT. Cette solution a servi de point de départ pour les deux approches métaheuristiques étudiées : l'algorithme génétique et la recherche Tabou. Ces deux méthodes ont été adaptées au problème considéré en utilisant une représentation des solutions par permutation des tâches et des mécanismes de recherche spécifiques.

Les résultats expérimentaux ont montré que les deux approches proposées permettent d'améliorer la solution initiale obtenue par la règle WSPT. L'algorithme génétique a obtenu les meilleures améliorations de la fonction objectif pour les différentes tailles d'instances testées grâce à sa capacité d'exploration globale basée sur une population de solutions et sur les opérateurs de croisement et de mutation.

En revanche, la recherche Tabou présente des performances intéressantes en termes de temps de calcul grâce à son mécanisme d'exploration locale guidée par une mémoire adaptative. Elle

permet ainsi d'obtenir rapidement des solutions améliorées avec un coût computationnel inférieur à celui de l'algorithme génétique.

L'analyse comparative met donc en évidence un compromis entre la qualité des solutions et l'efficacité computationnelle. Le choix de l'approche dépend du contexte d'utilisation : l'algorithme génétique est davantage adapté lorsque la recherche d'une solution de meilleure qualité constitue la priorité, tandis que la recherche Tabou représente une alternative efficace lorsque la rapidité d'exécution est un facteur déterminant.

Ainsi, les approches proposées constituent des outils complémentaires pour la résolution du problème d'ordonnancement étudié et permettent d'obtenir des solutions performantes malgré la complexité combinatoire du problème.

Conclusion générale

Les problèmes d’ordonnancement constituent un domaine majeur de la recherche opérationnelle et de l’optimisation combinatoire. Leur complexité augmente considérablement lorsque des contraintes réalistes sont introduites, notamment les périodes d’indisponibilité des machines qui représentent des situations fréquentes dans les environnements industriels tels que les opérations de maintenance préventive, les arrêts planifiés ou les changements de configuration.

Cette thèse s’est intéressée à l’étude d’un problème d’ordonnancement sur une machine unique soumise à des périodes déterministes d’indisponibilité. Les tâches considérées sont indépendantes et non préemptives, et l’objectif consiste à déterminer une séquence réalisable permettant de minimiser la somme pondérée des dates d’achèvement :

$$\min_{\sigma \in \Sigma} \sum_{i=1}^n w_i C_i(\sigma)$$

où Σ représente l’ensemble des ordonnancements admissibles, σ un ordonnancement donné, w_i le poids associé à la tâche i et $C_i(\sigma)$ sa date d’achèvement dans l’ordonnancement considéré.

Ce problème présente une difficulté particulière en raison de l’interaction entre les décisions de séquençement des tâches et les contraintes temporelles liées à la disponibilité de la machine. La présence des périodes d’indisponibilité modifie en effet l’ensemble des solutions réalisables et rend l’obtention d’une solution optimale plus complexe.

Les trois chapitres développés dans cette thèse ont permis de construire progressivement le cadre théorique, méthodologique et expérimental nécessaire à l’étude du problème considéré.

Le premier chapitre, consacré aux fondements théoriques des problèmes d’ordonnancement, a présenté les concepts fondamentaux relatifs aux tâches, aux ressources, aux contraintes et aux critères d’optimisation. Il a également présenté les principales classifications des problèmes d’ordonnancement selon la notation $\alpha|\beta|\gamma$ ainsi que les notions relatives à leur complexité algorithmique. Une attention particulière a été accordée aux problèmes intégrant des contraintes d’indisponibilité des machines.

Le deuxième chapitre a été consacré à l’analyse des principales méthodes de résolution des problèmes d’ordonnancement. Les méthodes exactes, telles que la programmation mathématique, la programmation dynamique et la méthode Branch-and-Bound, permettent de garantir l’optimalité des solutions lorsque la résolution est menée jusqu’à son terme. Cependant, leur coût computationnel augmente rapidement avec la taille des instances, ce qui limite leur utilisation.

tion pour les problèmes de grande dimension. Cette limitation justifie le recours aux méthodes approchées, notamment les métaheuristiques, capables de fournir des solutions de bonne qualité dans des temps de calcul raisonnables.

Le troisième chapitre a présenté l'approche de résolution proposée basée sur deux métaheuristiques complémentaires : l'algorithme génétique (*Genetic Algorithm, GA*) et la recherche Tabou (*Tabu Search, TS*). Une modélisation adaptée du problème a été développée afin de prendre en compte les contraintes de non-préemption et les périodes d'indisponibilité de la machine. Le calcul des dates d'achèvement a été conçu de manière à garantir qu'aucune tâche ne chevauche une période d'indisponibilité.

La méthode de recherche Tabou proposée repose sur une exploration locale guidée par une mémoire adaptative permettant d'éviter les cycles et de favoriser la diversification de la recherche. L'algorithme génétique exploite quant à lui une population de solutions associée à des opérateurs de sélection, de croisement et de mutation adaptés aux problèmes d'ordonnancement. Les deux approches ont été évaluées expérimentalement sur différentes tailles d'instances afin d'analyser leurs performances selon deux critères principaux : la qualité des solutions obtenues et le temps de calcul nécessaire.

Les résultats expérimentaux montrent que l'algorithme génétique permet généralement d'obtenir des solutions de meilleure qualité pour les instances étudiées, grâce à sa capacité d'exploration globale de l'espace de recherche. Cependant, cette amélioration s'accompagne d'un coût computationnel plus élevé, notamment lorsque la taille des instances augmente.

À l'inverse, la recherche Tabou présente un avantage significatif en termes de rapidité d'exécution tout en conservant des performances compétitives. Son mécanisme de recherche locale permet d'obtenir rapidement des solutions de bonne qualité, ce qui représente un avantage important dans les applications industrielles où le temps de décision constitue une contrainte essentielle.

Ainsi, les résultats obtenus mettent en évidence l'existence d'un compromis entre la qualité des solutions et l'efficacité computationnelle. Pour les instances considérées expérimentalement, l'algorithme génétique fournit des solutions plus performantes selon la valeur de la fonction objectif, tandis que la recherche Tabou constitue une alternative intéressante lorsque la rapidité de résolution est un facteur déterminant.

Les principales contributions de cette thèse concernent :

- la modélisation d'un problème d'ordonnancement sur une machine unique avec périodes déterministes d'indisponibilité ;
- l'adaptation de deux métaheuristiques classiques, à savoir l'algorithme génétique et la recherche Tabou, à ce contexte particulier ;
- l'analyse comparative expérimentale des performances des deux approches en termes de qualité des solutions et de temps de calcul.

Les résultats obtenus confirment l'intérêt des approches métaheuristiques pour le traitement de problèmes d'ordonnancement complexes lorsque les méthodes exactes deviennent difficilement applicables.

Malgré les résultats obtenus, plusieurs perspectives peuvent être envisagées. Une première extension concerne la généralisation du modèle vers des systèmes multi-machines tels que les environnements *flow shop* ou *job shop* avec contraintes d'indisponibilité. Il serait également intéressant d'intégrer d'autres contraintes industrielles, telles que les dates de disponibilité des tâches, les délais de livraison, les coûts de retard ou les dépendances entre opérations.

Par ailleurs, l'amélioration automatique des paramètres des métaheuristiques constitue une perspective importante. L'utilisation de méthodes d'optimisation des paramètres ou d'approches hybrides combinant plusieurs stratégies pourrait permettre d'améliorer davantage les performances obtenues.

Enfin, l'intégration d'incertitudes liées aux conditions réelles d'exploitation représente une direction de recherche prometteuse afin de développer des modèles d'ordonnancement plus robustes et plus proches des besoins des systèmes industriels modernes.

Bibliographie

- [1] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling : A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.
- [2] Peter Brucker. *Scheduling Algorithms*. Springer, Berlin, Heidelberg, 5 edition, 2007.
- [3] Michael L. Pinedo. *Scheduling : Theory, Algorithms, and Systems*. Springer, Cham, 6 edition, 2022.
- [4] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.
- [5] Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
- [6] Ying Ma, Chengbin Chu, and Chunrong Zuo. A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, 58(2) :199–211, 2010.
- [7] Imed Kacem, Chengbin Chu, and Ahmed Souissi. Single-machine scheduling with an availability constraint to minimize the weighted sum of completion times. *Computers & Operations Research*, 35(3) :827–844, 2008.
- [8] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2) :59–66, 1956.
- [9] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [10] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [11] El-Ghazali Talbi. *Metaheuristics : From Design to Implementation*. Wiley, 2009.
- [12] Ibrahim H. Osman and Gilbert Laporte. Metaheuristics : A bibliography. *Annals of Operations Research*, 63:513–623, 1996.

- [13] Günter Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1) :1–15, 2000.
- [14] Jacek Blazewicz, Klaus Ecker, Erwin Pesch, Günter Schmidt, and Jan Weglarz. *Handbook on Scheduling : From Theory to Applications*. Springer, Berlin, 2007.
- [15] Kenneth R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley & Sons, New York, 1974.
- [16] Dvir Shabtay. Single-machine scheduling with machine unavailability periods and resource dependent processing times. *European Journal of Operational Research*, 296(2) :423–439, 2022.
- [17] Shenquan Huang, Ya-Chih Tsai, and Fuh-Der Chou. A trajectory-based immigration strategy genetic algorithm to solve a single-machine scheduling problem with job release times and flexible preventive maintenance. *Algorithms*, 16(4) :207, 2023.
- [18] Zheng-Guo Lv, Li-Han Zhang, Xiao-Yuan Wang, and Ji-Bo Wang. Single machine scheduling proportionally deteriorating jobs with ready times subject to the total weighted completion time minimization. *Mathematics*, 12(4) :610, 2024.
- [19] Robert Pellerin, Nathalie Perrier, and François Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2) :395–416, 2020.
- [20] Lu Chen, Jinfeng Wang, and Wenhui Yang. A single machine scheduling problem with machine availability constraints and preventive maintenance. *International Journal of Production Research*, 59(9) :2708–2721, 2021.
- [21] Alessandro Agnetis, Jean-Charles Billaut, Michael L. Pinedo, and Dvir Shabtay. Scheduling with resource availability constraints : A survey. *European Journal of Operational Research*, 327(2) :367–393, 2025.
- [22] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.
- [23] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3) :497–520, 1960.
- [24] Nabila Lounissi, Omar Selt, and Allaoua Hemmak. A comparative study between a genetic algorithm and tabu search for scheduling problems on a single machine. *Journal of Information Systems Engineering and Management*, 10(59s) :27–34, 2025.
- [25] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

Résumé

Le travail réalisé dans cette thèse porte sur l'élaboration et l'évaluation des algorithmes génétiques pour résoudre un problème d'ordonnancement des tâches sur une machine unique sous contrainte de disponibilité. Ce problème est considéré comme NP-difficile, ce qui rend difficile la détermination d'un algorithme polynomial permettant de le résoudre de manière exacte pour des instances de grande taille.

Dans ce contexte, nous avons étudié la performance des algorithmes génétiques afin d'obtenir des solutions approchées de bonne qualité, tout en réduisant le coût lié au temps d'exécution. Le problème étudié est noté, selon la notation adoptée :

$$1 \mid h_k \mid \sum_{i=1}^n w_i C_i$$

où w_i représente le poids de la tâche i et C_i sa date d'achèvement.

Mots-clés : ordonnancement, machine unique, algorithme génétique, recherche Tabou, périodes d'indisponibilité.

Abstract

The work carried out in this thesis focuses on the development and evaluation of genetic algorithms to solve a single-machine scheduling problem under availability constraints. This problem is considered NP-hard, which makes it difficult to determine a polynomial algorithm capable of solving it exactly for large-size instances.

In this context, we studied the performance of genetic algorithms in order to obtain good-quality approximate solutions while reducing the computational cost in terms of execution time. The studied problem is denoted, according to the adopted notation, as follows :

$$1 \mid h_k \mid \sum_{i=1}^n w_i C_i$$

where w_i represents the weight of job i and C_i its completion time.

Keywords : scheduling, single machine, genetic algorithm, Tabu Search, machine unavailability.

الملخص

يتناول هذا العمل، في إطار هذه الأطروحة، إعداد وتقييم الخوارزميات الجينية من أجل حل مشكلة جدولة المهام على آلة واحدة مع وجود فترات عدم إتاحة. وتُعد هذه المشكلة من المسائل صعبة الحل من نوع NP، مما يجعل إيجاد خوارزمية متعددة الحدود قادرة على حلها بشكل دقيق أمراً صعباً، خاصة عند التعامل مع مسائل ذات حجم كبير.

في هذا السياق، قمنا بدراسة أداء الخوارزميات الجينية بهدف الحصول على حلول تقريبية ذات جودة جيدة، مع تقليل الكلفة الحسابية المرتبطة بزمن التنفيذ.

$$1 \mid h_k \mid \sum_{i=1}^n w_i C_i$$

حيث يمثل w_i وزن المهمة i ، ويمثل C_i زمن إتمام المهمة. الكلمات المفتاحية: الجدولة، آلة واحدة، الخوارزميات الجينية، البحث المحظور، فترات عدم الإتاحة.