الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة أبي بكر بلقايد – تلمسان –
Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

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

En : (Génie industriel)

Spécialité : Ingénierie de la production et Ingénierie des systèmes

Par :
- BALKHIR Zakaria
- MOULAY Omar Idriss

Sujet

Algorithme des chameaux pour la minimisation de makespan dans un atelier flow shop

Soutenu publiquement, le 20/06/2023 , devant le jury composé de :

Président : Pr SOUIER Mehdi	Professeur à l'Université de Tlemcen
Examinateur 1 : Dr LARIBI Imane	MCB à l'Université de Tlemcen
Examinateur 2 : Dr ABDELLAOUI Wassila	MAB à l'Université de Tlemcen
Encadrante : Dr HOUBAD Yamina	MCB à l'Université de Tlemcen
Co-encadrante: Dr BOUMEDIENE Fatima Zohra	Enseignante vacataire à l'Université de Tlemcen

Année universitaire : 2022 /2023


بسم الله الرحمن الرحيم
١٤٢٠

Résumé

Dans ce travail, nous avons abordé la résolution d'un problème d'ordonnancement dans le domaine de la production. L'ordonnancement de la production est une branche dans la recherche opérationnelle. Les problèmes d'ordonnancements sont classés parmi les problèmes NP-difficiles, et pour les résoudre, des méthodes basées sur le développement d'algorithmes sont utilisées en fonction de leur degré de difficulté. Ainsi, plusieurs méta-heuristiques ont été développées dans ce cadre. Dans notre étude, nous avons utilisé une métaheuristique hybride qui combine une nouvelle méta-heuristique appelée "Algorithmes des chameaux » avec la métaheuristique « recherche par dispersion ».

Enfin, nous avons comparé les résultats obtenus par ces méta-heuristiques suite aux simulations avec l'algorithme génétique, et il s'est avéré que l'algorithme des chameaux nous a donné d'excellents résultats. Pour améliorer les solutions de cette méta-heuristique, nous avons hybridées avec l'algorithme de recherche dispersée, ce qui nous a donné des bonnes solutions en comparaison avec celles fournies par l'algorithme génétique.

Mots-clés : algorithme des chameaux, ordonnancement, flow shop, algorithme hybride algorithme génétique, la recherche dispersée

الملخص

في هذه الأطروحة، تطرقنا إلى دراسة موضوع الترتيبات في الإنتاج. تُعتبر الترتيبات الإنتاج فرعًا من فروع البحوث العملياتية. كما تُعتبر مشكلات الترتيبات جزءًا من إدارة الإنتاج التي تهدف إلى تطوير أنظمة الإنتاج. تُصنف مشكلات الترتيبات ضمن المشكلات ذات الحدود الصعبة الكثيرة. ولحل هذه المشكلات، نحتاج إلى طرق محددة واستخدام خوارزميات وفقًا لصعوبتها. لذلك، تم تطوير عدة فوقيات استدلال لحل هذه المشكلات. في دراستنا، قمنا بتطبيق فوقيات استدلال جديدة تُسمى "خوارزميات الجمال".

وفي النهاية، قمنا بمقارنة النتائج التي أظهرتها الخوارزميات بعد المحاكاة باستخدام الخوارزمية الجينية، وتبين لنا أن خوارزمية الجمال أعطتنا نتائج جيدة جدًا وحلول محسنة. ولتحسين حلول هذه الخوارزمية، قمنا بدمجها مع خوارزمية البحث المشتت وأعطتنا حلولًا جيدة مقارنةً بالحلول التي أعطتها لنا الخوارزمية الجينية.

الكلمات المفتاحية: خوارزمية الجمال، ترتيبات الإنتاج، وحيدة المسار، الخوارزمية المهجنة، الخوارزمية الوراثة، خوارزمية البحث المتشتت

Abstract

In this thesis, we addressed the study of scheduling in the field of production. Production scheduling is considered a branch of operations research. Furthermore, scheduling issues are part of production management, which aims to develop production systems. Scheduling problems are classified as NP-hard problems, and to solve these problems, we need specific methods and algorithms based on their difficulty level. As a result, several meta-heuristics have been developed to tackle these issues. In our study, we applied new meta-heuristics called "Camel Algorithms". Finally, we compared the results obtained by these meta-heuristics through simulations with the genetic algorithm, and it was evident that the Camel Algorithm provided excellent results and improved solutions. To enhance the solutions of this meta-heuristic, we hybridized them with the scatter search algorithm, which yielded good solutions compared to those provided by the genetic algorithm.

Keywords: camel Algorithm, scheduling, flow shop, hybrid algorithm, genetic algorithm., scatter search

Remerciement

Tout d'abord, nous remercions Allah pour nous avoir accordé la réussite, le soutien, la persévérance et nous avoir aidés à accomplir cette Travail, à travers laquelle nous espérons avoir apporté une contribution de qualité dans le domaine de la recherche scientifique

Nous tenons à exprimer notre gratitude envers Mademoiselle ***Yamina HOUBAD*** *et Mademoiselle* ***BOUMADIENE Fatima Zohra*** *pour les informations et l'aide qu'elles nous ont fournies tout au long de la préparation.*

Nous tenons également à remercier tous ceux qui ont contribué à ce travail, qu'ils soient proches ou éloignés.

Nous adressons également nos sincères remerciements aux membres du jury qui ont accepté s'évaluer notre projet, et nous leur adressons leur respect et leurs meilleures salutations.

Dédicace

Je dédie ma remise de diplôme et le fruit de toutes les années que j'ai consacrées à mes parents formidables, qui ont travaillé dur et se sont efforcés de fournir tous les efforts afin que je puisse poursuivre mon parcours éducatif jusqu'à atteindre ce moment. À vous deux, je vous envoie tout mon amour, mon respect et mon admiration… Et à mes frères, sœurs et à ma famille en général, ainsi qu'à mes amis, je vous envoie mes expressions les plus sincères d'affection et de respect.

Zakaria BALKHIR **Omar Idriss MOULAY**

Table de matière

Liste des figures

Liste des algorithmes

Liste des tableaux

Liste des abréviations

SPT: Shortest processing time first
LPT: Longest processing time first
AG: Algorithme génétique
AC: Algorithme des chameaux

Introduction générale

Introduction générale

Le développement industriel récent a connu une accélération significative et une avancée qualitative dans tous les domaines industriels. Autrefois, les humains comptaient sur des méthodes et des machines traditionnelles pour leurs industries, qui étaient complexes tant du point de vue de leur fonctionnement que de la qualité de leur production. Cependant, ces méthodes sur lesquelles ils comptaient étaient moins productives, prenant beaucoup de temps et d'efforts pour atteindre les produits finis.

Avec l'avènement de la révolution industrielle, les événements se sont accélérés au point que les propriétaires d'usines ont dû suivre cette accélération en produisant des produits de haute qualité en peu de temps. En atteignant ces objectifs, les propriétaires d'usines ont rencontré plusieurs problèmes dans leurs usines, ce qui a incité les scientifiques industriels à mener plusieurs recherches pour améliorer la production, réduire le temps de production et en même temps faire fonctionner les usines de manière automatique. Ils ont également abordé l'étude des problèmes prévus tels que les problèmes de planification et les problèmes d'ordonnancement.

Il est remarqué que la plupart des usines rencontrent plusieurs problèmes dans leurs ateliers de production, notamment en ce qui concerne l'ordonnancement de la production. Chaque atelier à des méthodes pour résoudre les problèmes d'optimisation qui y sont présents, et les algorithmes sont utilisés en fonction du degré de complexité du problème. Parmi ces problèmes, on peut citer les problèmes d'ordonnancement des machines dans les ateliers (atelier flow shop), ainsi que les problèmes de minimisation du temps de sortie des produits.

L'une des principales préoccupations des propriétaires d'usines est la gestion de la production, la planification et l'ordonnancement de la production.

Certaines de ces préoccupations sont faciles à résoudre et peuvent être résolues de manière simple, tandis que d'autres sont plus complexes et nécessitent des études approfondies pour être résolues. Tout dépend du degré de complexité du problème. Parmi ces problèmes, certains ont une solution unique, souvent moins complexes et peuvent être résolus à l'aide d'algorithmes connus tels que l'algorithme tabou, l'algorithme de recuit simulé, ou d'autres ont à population des solutions, telles que les algorithmes génétiques, l'algorithme des fourmis et l'algorithme de l'essaim et l'algorithme des chameaux.

La plupart de ces algorithmes sont inspirés de la nature et du comportement des êtres vivants. De plus, lorsqu'ils sont combinés avec d'autres algorithmes, ils peuvent nous aider à améliorer la qualité des solutions, comme l'algorithme de recherche dispersée.

L’organisation du manuscrit est comme suit : Dans le premier chapitre, nous avons abordé l'ordonnancement de la production, les problèmes d'ordonnancement de la production et les ateliers d'ordonnancement de la production.

Nous avons également présenté les différents types de complexité des problèmes d'ordonnancement de la production.

Dans le deuxième chapitre, nous avons abordé les problèmes d'optimisation, les différentes méthodes de résolution des problèmes d'optimisation, ainsi que la définition des métaheuristiques et des heuristiques. Nous avons également présenté quelques algorithmes tels que l'algorithme génétique, l'algorithme des chameaux et l'algorithme de recherche dispersée.

Dans le troisième chapitre, nous avons étudié le problème de l'ordonnancement de la production dans un atelier de type "Flow Shop". Nous avons adaptée l'algorithme génétique, l'algorithme des chameaux et comparé les résultats des deux algorithmes. Enfin, nous avons développé un nouvel algorithme hybride combinant l'algorithme des chameaux et l'algorithme de recherche dispersée, et nous avons comparé ses résultats avec les deux algorithmes précédents. Nous avons également analysé les résultats après leur application à plusieurs problèmes.

I. Chapitre : Généralité sur l'ordonnancement de la production

1. Introduction

En industrie, l'ordonnancement signifie déterminer l'ordre d'exécution des différentes tâches et allouer les ressources nécessaires en vue d'atteindre les objectifs de production avec le coût le plus bas et la meilleure qualité possible. Cela se fait par l'utilisation des ressources disponibles dans le domaine industriel, par exemple (les machines, les équipements, les travailleurs, les matières premières, l'énergie… etc.). De manière à garantir la réalisation des objectifs définis et répondre aux besoins en temps, en ressources et en qualité de production.

Les défis auxquels est confronté le domaine industriel en matière d'ordonnancement comprennent l'allocation des ressources et la gestion des contraintes de temps, de ressources, de qualité, de prix, etc. Les techniques de planification dans l'industrie sont donc très importantes pour améliorer la productivité, la réduction des coûts, augmenter la rentabilité des entreprises, améliorer la qualité des produits et aussi permettre l'obtention des solutions optimales.

Dans ce chapitre, nous présentons les différentes définitions relatives aux systèmes de production, les concepts de base de l'ordonnancement des systèmes de production, ainsi que la présentation des problèmes d'ordonnancement et des différents éléments qui les composent. Nous aborderons également les types d'ateliers et les différents problèmes qui se posent à chaque niveau d'atelier, ainsi que la typologie des problèmes d'ordonnancement et la présentation des caractéristiques générales de l'ordonnancement ainsi que le concept complexité des problèmes d'optimisation.

2. Systèmes de production

Un système de production est composé d'un ensemble organisé de ressources qui interagissent et interviennent ensemble pour produire les biens ou services requis. Comme tout système dans la vie réelle, les systèmes de production sont limités, car ils ne peuvent pas supporter une charge de travail illimitée. Ce qui détermine ces limites, ce sont les ressources disponibles en quantité et/ou capacité limitée. Par conséquent, ces systèmes sont classés sous le nom de "systèmes de production à partage de ressources**.** (BAHMANI, 2017)

Le concept de système de production doit être compris dans le sens large d'un système industriel manufacturier et doit inclure les principaux processus fonctionnels tels que la définition et la mise en œuvre des stratégies d'entreprise, la conception des produits et des processus de production, la conduite et la gestion de ces systèmes. (BENBOUZID SITAYEB, 2005).

3 Ordonnancement des systèmes de production

3.1 Définitions

L'ordonnancement consiste à ordonnancer plusieurs tâches en programmant leur mise en œuvre en s'appuyant sur les ressources requises pour ces tâches et en déterminant les dates de leurs débuts. La théorie de l'ordonnancement est basée sur des modèles mathématiques. Dans le rôle de cette théorie, nous pouvons analyser des situations réelles et complexes. Le développement de cette théorie est dû au résultat de la communication entre la théorie et la pratique. (EMBOUAZZA & BETTAHAR, 2020)

L'ordonnancement joue un rôle important en tant que processus de prise de décision dans la plupart des systèmes de fabrication et de production, ainsi que dans la plupart des environnements de traitement de l'information. Elle est également importante dans la planification des horaires de transport et de distribution, ainsi que dans d'autres types d'industries de services**.** (L & Pinedo, 2016)

L'ordonnancement se compose de trois étapes distinctes (MAMOUNI & OULD MOHAMED , 2018) :

La planification implique l'identification des opérations à effectuer, la fixation des dates correspondantes, ainsi que l'allocation des moyens nécessaires (matériels et humains) à leur réalisation.

La phase d'exécution a pour objectif de réaliser les opérations déterminées lors de la phase de planification.

La phase de contrôle implique la comparaison entre la planification et l'exécution, soit en termes de couts, soit en termes de délais d'exécution.

Il est nécessaire que la solution à un problème d'ordonnancement général réponde aux questions : quand et avec quels moyens ?

Une réponse qui satisfait ces questions est nommée "ordonnancement". Une technique qui permet de créer un ordonnancement est appelée "algorithme" ou "méthode de résolution". Un ordonnancement réalisable est un ordonnancement qui respecte toutes les contraintes du problème. Pour simplifier, nous utilisons le terme "ordonnancement" pour désigner un ordonnancement réalisable. (MAMOUNI & OULD MOHAMED , 2018) , (CHERGUI & DAHMANI, 2017).

3.2 Les paramètres d'un problème d'ordonnancement

À partir de notre définition de l'ordonnancement, il apparaît que quatre éléments clés sont responsables dans un problème d'ordonnancement, à savoir les tâches, les ressources, les contraintes et les objectifs (MAMOUNI & OULD MOHAMED , 2018).

3.2.1 Les tâches

Le terme "tâche" représente une entité de travail élémentaire définie temporellement par une date de début et de fin, et caractérisée par une durée et une quantité spécifique de ressources utilisées avec une certaine intensité. Il est supposé que cette intensité reste constante pendant l'exécution de la tâche. (HADRI, 2012)

Pour exécuter une tâche, il est nécessaire de programmer un ensemble d'opérations qui demandent des ressources spécifiques, afin d'optimiser un objectif particulier**.** (BOUMEDIENE , 2020)**.**

On peut identifier deux types de tâches.

3.2.1.1 Les tâches morcelables (préemptives)

Il est possible de diviser leur exécution en plusieurs parties, ce qui facilite la résolution de certains problèmes. (BOUMEDIENE , 2020)

3.2.1.2 Les tâches non morcelables (indivisibles)

Et qui nécessitent d'être exécutées en une seule fois sans interruption, et ne sont arrêtées qu'après leur achèvement. (BOUMEDIENE , 2020)

L'ordonnancement est la fonction de déterminer un plan optimal (ou réalisable) d'implantation.

Les données qui spécifient chaque tâche Jj peuvent être exprimées comme suit :

- P_j : Une durée de traitement de pièce
- r_j : Une date de disponibilité
- t_j: Une date de début
- c_j: Une date de fin
- d_j : Une date due
- w_j: Un facteur de priorité ou poids

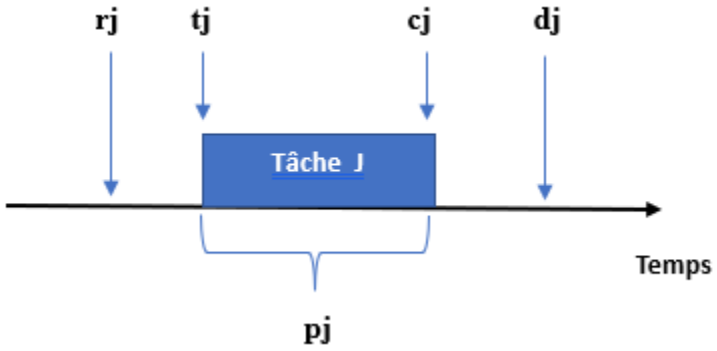

Figure 1.1 : Caractéristique d'une tâche

3.2.2 Les ressources

La ressource est un moyen humain ou technique utilisé pour réaliser différentes tâches dans un atelier, où elle est disponible en quantités limitées. (LARIBI, 2018)
Les ressources sont divisées en deux types. (Voir la figure 3)

3.2.2.1 Ressources renouvelables

À nouveau disponibles après avoir été affectées à une tâche (machines, personnes, etc.) (BENTTALEB, 2018).

Parmi les ressources renouvelables, il y a les ressources disjonctives qui ne peuvent effectuer qu'une seule opération à la fois, et les ressources cumulatives qui peuvent effectuer un nombre limité d'opérations simultanément**.** (SAOUDI & REMITA)

3.2.2.2 Ressource consommable

Lorsqu'une ressource reste disponible en même quantité (par exemple une équipe ou une machine dans un atelier), elle est considérée comme consommable. (AZEM, 2010)

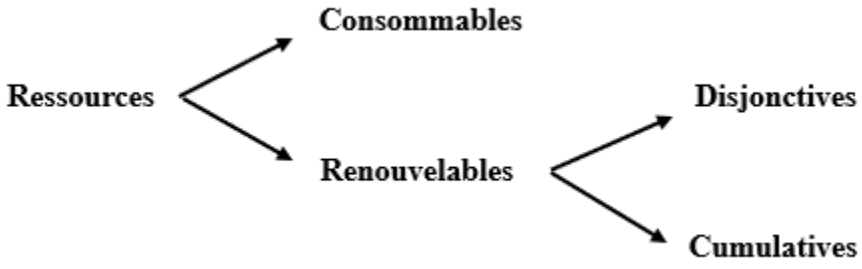

Figure 1.2 : Classification des ressources

3.2.3 Les contraintes

Les contraintes imposent des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision. Autrement dit, elles représentent les conditions à respecter lors de l'élaboration de l'ordonnancement afin qu'il soit exécutable. À mesure que le nombre de contraintes augmente, la difficulté de la problématique d'ordonnancement s'accroît (MAMOUNI & OULD MOHAMED , 2018) .

Dans les problèmes d'ordonnancement les contraintes se divisent en deux types : contraintes de ressources et contraintes temporelles. (LARIBI, 2018)

3.2.3.1 Les contraintes de ressources

Les limites imposées sur l'utilisation des ressources renouvelables reflètent la quantité de moyens nécessaires pour accomplir les tâches. Selon la nature des ressources, il existe deux types de contraintes distinctes (EMBOUAZZA & BETTAHAR, 2020)

- **Les contraintes disjonctives,** Impliquent que l'utilisation d'une même ressource est limitée dans le temps pour la réalisation des tâches, c'est-à-dire qu'elle ne peut être utilisée que par une seule tâche à la fois (BENTTALEB, 2018) .
- **Les contraintes cumulatives,** Limitent le nombre de tâches qui peuvent être effectuées simultanément. (BENTTALEB, 2018)

3.2.3.2 Les contraintes temporelles

On peut classer les contraintes temporelles en deux catégories : les contraintes de temps absolu et les contraintes de temps relatif. Les contraintes de temps absolu sont utilisées pour définir les limites possibles des dates des tâches**.** (SOUIER, 2012)

- **Contraintes de temps absolu,** Elle permet de garantir la cohérence des dates des tâches en définissant notamment la date de début et la date de fin optimales pour chaque tâche. (BELKAID, 2014)
- **Contraintes de temps relatif,** Cette contrainte est en relation avec les exigences de cohérence technologique, telles que les contraintes de gamme qui nécessitent de respecter la position relative des tâches. (BELKAID, 2014).

3.2.4 Les critères

Le critère comprend des exigences qualitatives et quantitatives qui doivent être satisfaites, afin de permettre l'évaluation de la qualité de l'ordonnancement établi**.** (BOUKEF BEN OTHMAN, 2009)

Certains critères ne sont pas indépendants les uns des autres, et certains d'entre eux peuvent même être équivalents. En effet, deux critères sont considérés comme équivalents si une solution optimale pour l'un d'entre eux est également optimale pour l'autre, et vice versa. (BOUKEF BEN OTHMAN, 2009)

Il existe différentes formes que peut prendre un critère (BOUMEDIENE , 2020)

- ✓ Réduire la durée d'exécution de la dernière tâche (makespan) au minimum.
- ✓ Réduire la moyenne des dates d'achèvement des tâches au minimum.
- ✓ Réduire les retards sur les dates d'achèvement des tâches au minimum.
- ✓ Réduire le maximum des retards sur les dates d'achèvement des tâches au minimum.
- ✓ Réduire le montant des encours au minimum.
- ✓ Réduire le coût de stockage des matières premières au minimum.
- ✓ Répartir équitablement les charges des machines.
- ✓ Optimiser les changements d'outils.

3.3 Les différents types des problèmes d'ordonnancement

On peut trouver différents types d'ateliers de production dans les systèmes industriels, où chaque atelier se caractérise par le nombre et le type de machines qui y sont présentes ainsi que l'ordre de passage des produits à fabriquer.

Les problèmes liés à l'ordonnancement de la production dans les usines sont caractérisés par de nombreux facteurs, notamment le nombre et l'agencement des machines dans l'usine, ainsi que le nombre d'opérations incluses dans chaque tâche et leur ordonnancement sur les machines, en plus du nombre de machines capables d'exécuter une seule opération en même temps.

Il existe une distinction entre les problèmes relatifs à une seule machine et ceux relatifs à plusieurs machines (machines parallèles, flow shop, flow shop hybride, job shop, job shop flexible et open shop).

3.3.1 Atelier à une seule machine

Cette opération consiste à ordonner un groupe de tâches, chacune ne comportant qu'une seule opération, sur une seule machine dans le but de réduire la durée d'exécution.la figure suivante représente l'atelier.

Les problèmes d'atelier qu'une seule machine est responsable de l'exécution de toutes les tâches dans ce cas. Chaque tâche ne consiste en qu'une seule opération qui requiert la même machine. (MAMOUNI & OULD MOHAMED , 2018)

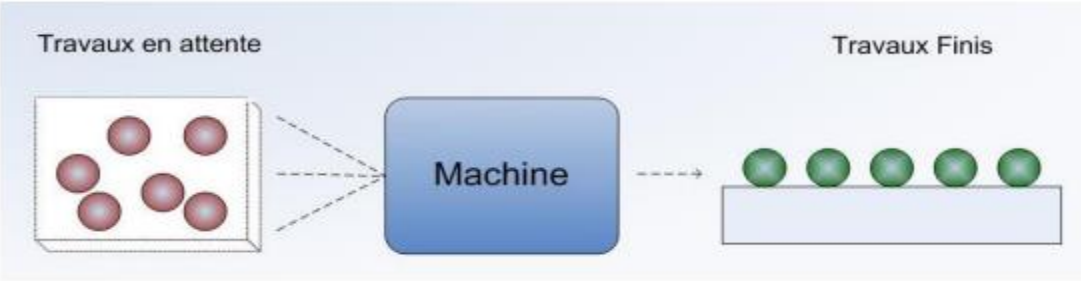

Figure 1.3: Atelier une seule machine

3.3.2 Atelier à Machines parallèles

L'atelier de machines parallèles est un type d'environnement de fabrication où plusieurs machines sont utilisées pour effectuer des tâches simultanément. Les machines sont disposées en parallèle les unes par rapport aux autres et peuvent traiter plusieurs tâches en même temps, ce qui permet de réduire le temps de traitement global et d'augmenter le rendement de production. Ceci représente une généralisation du problème d'ordonnancement à une seule machine, où les tâches sont composées d'une seule opération et chaque opération dispose d'un ensemble de machines parallèles, mais n'a besoin que d'une seule pour être exécutée. L'ordonnancement se déroule en deux phases : la première consiste à affecter les opérations aux machines et la deuxième à déterminer la séquence d'exécution de chaque opération sur chaque machine. (MAMOUNI & OULD MOHAMED , 2018)

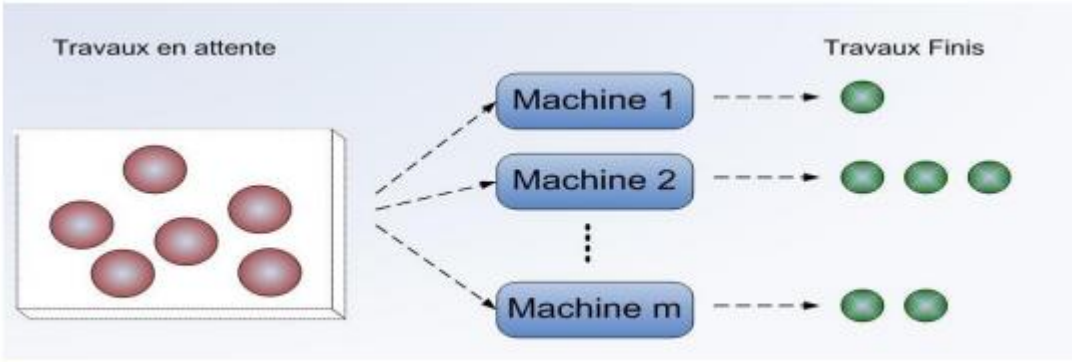

Figure 1.4 : Exemple d'atelier machines parallèles

Dans un atelier des machines parallèles il y a plusieurs types des machines (SAOUDI & REMITA):

- **Machines identiques (P),** Toutes les machines **Mi** et toutes les tâches **Jj**. Ont la même vitesse d'exécution.
- **Machines uniformes (Q),** Chaque machine **Mi** a une vitesse d'exécution propre et constante, de sorte que la vitesse d'exécution pour les tâches **Jj** spécifiques à chaque machine est égale au niveau de cette machine **Mi.**
- **Machines indépendantes (R),** Chaque machine **Mi.** Et chaque tâche **Jj** ont une vitesse d'exécution différente.

3.3.3 Atelier de type Flow shop

Dans ce type d'atelier, les machines sont disposées en série, chaque tâche est composée de plusieurs opérations et toutes les machines sont visitées selon le même ordre d'opérations défini pour la tâche.

Les ateliers de type "flow-shop" ont une ligne de production constituée de plusieurs machines disposées en série, où l'exécution des tâches doit se faire par toute les machines dans un ordre identique. Cette configuration est connue sous le nom "cheminement unique". (HOUARI, 2012)

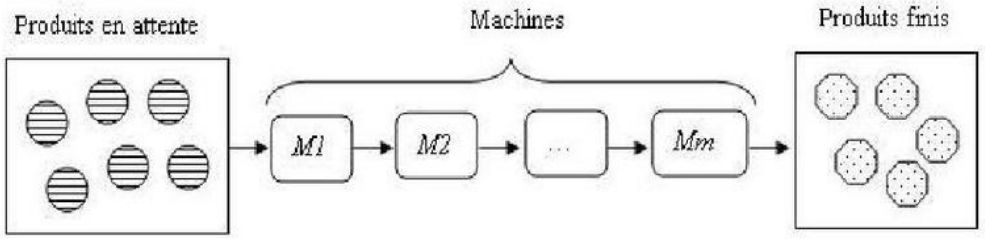

Figure 1.5 : Atelier Flow Shop

Donc Il existe quatre types de Flow Shop :

- **Flow shop hybride (flexible),** Dans cet atelier flow shop, chaque machine est remplacée par un ensemble de machines présentes à chaque étage, qui ne sont pas nécessairement identiques et sont disposées en parallèle. Chaque travail visite chaque étage successivement et ne doit passer que par une seule machine par étage. (MAMOUNI & OULD MOHAMED , 2018)

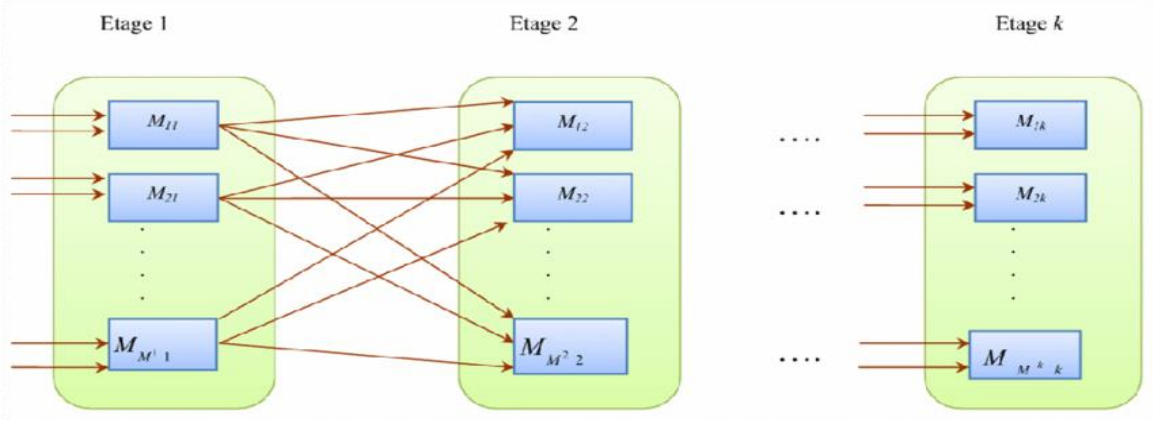

Figure 1.6 : Flow shop hybrid à k étage

- **Flow shop généralisé,** Si une tâche ne nécessite pas de traitement sur une machine spécifique, le temps nécessaire pour l'exécuter peut-être nul.
- **Flow shop pur,** Il n'y a aucun temps opératoire qui est négatif. (Positif).
- **Flow shop de permutation,** Toutes les tâches sont disponibles à l'instant 0, elles sont exécutées dans un ordre spécifié à cet instant, et le temps imparti ne peut pas être dépassé. (MAMOUNI & OULD MOHAMED , 2018)

3.3.4 Atelier de type Job shop

À la différence des problèmes d'ordonnancement en mode Flow shop, les tâches ne suivent pas un ordre prédéfini sur toutes les machines. En effet, chaque tâche est libre d'emprunter son propre chemin. (BELKAID, 2014)

Ont devisé en deux parties

Job-Shop, On réalise les opérations dans les ateliers de type "job-shop" selon un ordre précis et déterminé, qui varie en fonction de la tâche à accomplir. On nomme également ces ateliers des ateliers à chemins multiples. Dans cette situation, il est nécessaire de prévoir plusieurs changements d'outils. (MAMOUNI & OULD MOHAMED , 2018)

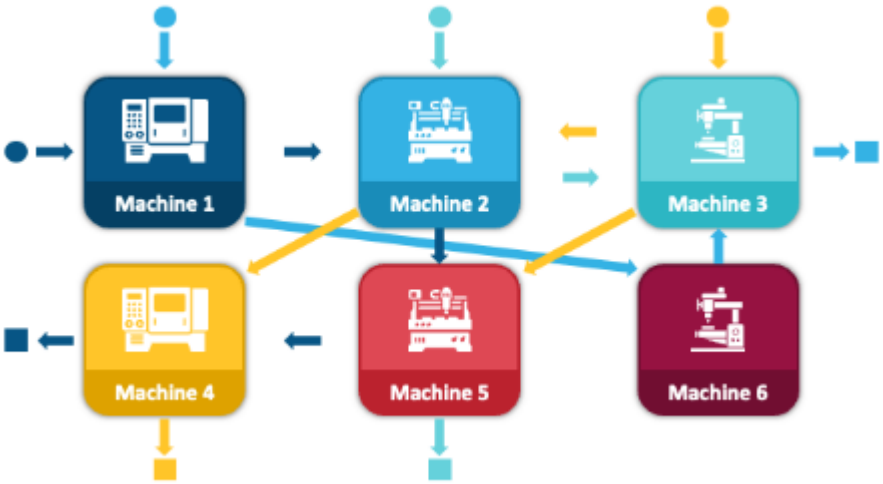

Figure 1.7 : Exemple d'un atelier Job shop

Job-Shop flexible, Le travail flexible dans le magasin de travail est une extension du modèle de magasin de travail traditionnel, et se caractérise par le fait que plusieurs machines peuvent potentiellement être capables d'exécuter une partie des opérations. (BOUMEDIENE , 2020)

3.3.5 Atelier de type Open shop

Comparé aux ateliers de type flow-shop ou job-shop, ce type d'atelier est moins restrictif, car l'ordre des opérations n'est pas prédéterminé à l'avance.

Ce type de problème peut être décrit comme permettant l'exécution des opérations nécessaires pour accomplir chaque tâche dans n'importe quel ordre approprié. Autrement dit, ce type de problème se produit lorsque la fabrication de chaque produit nécessite l'exécution d'une séquence d'opérations, mais dans un ordre complètement libre, où le planificateur est responsable de déterminer l'orientation de chaque tâche et de leur ordre d'exécution. (BELKAID, 2014)

3.4 La typologie des problèmes d'ordonnancement

Selon la nature des variables impliquées et des contraintes associées, plusieurs classifications pour les problèmes d'ordonnancement sont proposées, dont la complexité est souvent de type NP difficile. Une typologie de ces problèmes peut être dressée dans l'architecture de conception suivante**.** (Remla, 2019)

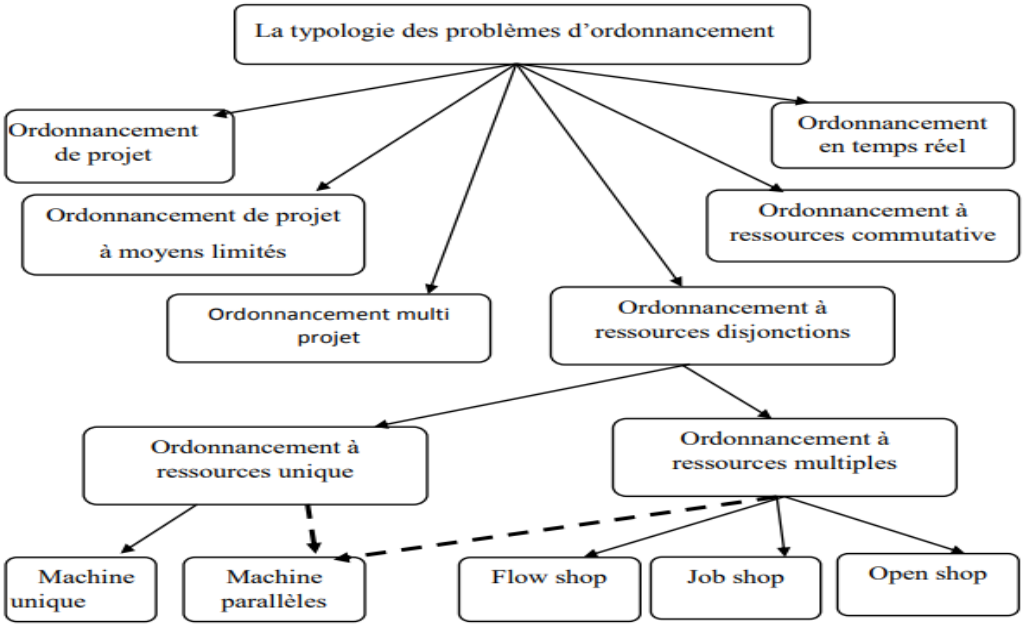

Figure 1.8 : la représentation des types d'ordonnancement (Remla, 2019)

3.5 Complexité des problèmes d'ordonnancement

En général, on peut dire que les problèmes d'ordonnancement des ateliers sont considérés comme des problèmes de combinaison difficiles, et il n'existe aucune méthode universelle capable de résoudre tous les cas (BOUKEF BEN OTHMAN, 2009)**.**

D'un autre côté, les systèmes qui sont définis par un processus de réentrance se caractérisent par la réintégration des machines de traitement plusieurs fois, ce qui accroît leur complexité (BELKAID, 2014)**:**

L'application de la théorie de la complexité computationnelle est très importante pour l'analyse des problèmes d'ordonnancement et pour déterminer s'il existe une chance

raisonnable de trouver un algorithme polynomial pour résoudre ces problèmes. (A.J & C.N, On the complexity of coupled tasks scheduling, 1997)

Il est possible de mentionner deux formes de complexité :

La complexité méthodologique, Elle exprime la relation entre le nombre d'opérations arithmétiques de base effectuées par la méthode ou l'algorithme de résolution et le nombre de données du problème traité. (BOUKEF BEN OTHMAN, 2009)

La complexité problématique, Cela concerne à la fois la complexité du problème à résoudre et le nombre d'opérations de base qu'un algorithme déterministe peut effectuer pour atteindre l'optimum en fonction de la taille du problème. (BOUKEF BEN OTHMAN, 2009)

La complexité du problème dépend de sa difficulté à être résolu et du nombre d'opérations élémentaires que l'algorithme peut effectuer pour rechercher la solution optimale en fonction de la taille du problème spécifié. (LARIBI, 2018)

Diverses catégories de difficulté sont utilisées pour classer les problèmes d'optimisation combinatoire en fonction de leur niveau de complexité.

Les problèmes de classe P (polynomial) : En d'autres termes, résoudre les problèmes polynomiaux nécessite l'utilisation d'algorithmes qui les résolvent en un temps polynomial dépendant de la taille du problème. (BOUMEDIENE , 2020)

La classe des problèmes NP (NP : Non Polynomial) : Dans cette classe, on peut résoudre les problèmes en utilisant un algorithme polynomial non-déterministe. Il est possible de distinguer deux sous-classes à l'intérieur de cette catégorie (BOUMEDIENE , 2020) :

La classe NP-complet : Si un algorithme polynomial existe pour résoudre un problème NP-complet, alors il existe des algorithmes polynomiaux pour résoudre tous les autres problèmes de la même classe.

La classe NP-difficile : Ces problèmes ne peuvent pas être résolus en temps polynomial, cependant, on peut élaborer des algorithmes pseudo-polynomiaux ou des méthodes approchées qui, au cours de leur exécution, effectuent des choix dont l'optimalité ne peut être démontrée.

3.6 Les Méthodes de résolution des problèmes d'ordonnancement

Comme mentionné précédemment, les problèmes appartenant à la classe P se caractérisent par la présence d'algorithmes efficaces et d'une complexité multi-limites, tels que la méthode du chemin critique et la méthode PERT dans la planification de projets. Pour les autres problèmes, il est difficile de trouver des algorithmes similaires ,on utilise une famille de méthodes précises ou approximatives. (MOUHOUB, 2011)

Les POC « problèmes de l'ordonnancement combinatoire » font partie de la catégorie des problèmes difficiles en termes de calcul, NP-difficiles, et il existe de nombreuses méthodes utilisées pour résoudre ces problèmes. Dans cette section, différentes méthodes seront présentées pour résoudre le problème de l'ordonnancement combinatoire « l'ordonnancement » (HOUARI, 2012).

3.6.1 Méthodes exactes

Les méthodes exactes sont des techniques de résolution qui, grâce à une exploration intelligente de l'espace des solutions, fournissent des solutions optimales pour des problèmes combinatoires, mais pas systématiquement dans un temps polynomial. Cependant, les problèmes d'ordonnancement, qui sont généralement NP-difficiles, ne peuvent pas être résolus en temps polynomial par des algorithmes généraux. Par conséquent, ces techniques ne sont pas couramment utilisées dans les milieux industriels. (BELKAID, 2014)

Cependant, les résultats des techniques précises pour les petites instances peuvent être utilisés pour des problèmes de taille industrielle afin d'essayer de trouver des tendances dans les techniques de résolution. La programmation linéaire, la programmation dynamique, la procédure par séparation et l'évaluation sont quelques exemples de méthodes exactes. (BELKAID, 2014)

3.6.2 Méthodes approchées

Les chercheurs ont trouvé des solutions alternatives appelées méthodes approchées en raison des difficultés rencontrées par les méthodes exactes pour résoudre des problèmes d'optimisation de large taille dans un temps raisonnable. Ces techniques

nous permettent d'obtenir une solution satisfaisante dans un temps de calcul raisonnable. La performance de ces méthodes est généralement donnée par l'estimation du pourcentage d'erreur entre la valeur de la solution fournie et la valeur de la solution optimale si elle est calculable. Si la solution optimale n'est pas calculable, il est également possible d'étudier et d'évaluer expérimentalement le comportement d'une méthode en comparant ses performances à celles d'autres méthodes ou à des bornes inférieures. (LARIBI, 2018)

Ces techniques sont appelées heuristiques lorsqu'elles sont conçues pour résoudre un problème spécifique de manière simple, rapide et ciblée. Cependant, elles sont appelées métaheuristiques lorsqu'elles sont générales, adaptables et applicables à plusieurs catégories des problèmes d'optimisation combinatoire. (LARIBI, 2018)

4 Conclusion

Dans ce chapitre, nous avons abordé le système de production en le définissant, puis nous avons étudié l'ordonnancement en la définissant ainsi que les différentes configurations qui nous permettent de définir les problèmes d'ordonnancement. Il est apparu que quatre éléments clés sont responsables dans un problème d'ordonnancement, à savoir les tâches, les ressources, les contraintes et les critères. Ainsi, l'ordonnancement peut être appliquée à plusieurs types des problèmes, qu'ils soient simples ou complexes, afin d'obtenir des solutions et des résultats optimaux.

II. Chapitre : Les méthodes de résolution des problèmes d'optimisation

1 Introduction

Avec le développement de la science et de l'industrie, nous constatons que l'industrie a fait un bond qualitatif, ce qui a conduit à une grande évolution industrielle. Cependant, avec cette évolution, les problèmes industriels sont devenus plus complexes et nécessitent plus de concentration et de régulation.

La raison en est que nous trouvons des solutions, mais la plupart du temps, elles ne sont pas considérées comme des solutions de qualité optimal.

Cela a poussé les scientifiques à rechercher des solutions plus efficaces pour répondre aux exigences de l'industrie.

C'est ce qui les a poussés à étudier le comportement des organismes et la façon dont ils gèrent leurs problèmes, à déduire des algorithmes de leur comportement, et à appliquer ces algorithmes pour résoudre et optimiser la qualité des solutions aux problèmes de production, en particulier les problèmes d'ordonnancement de la production.

Les algorithmes d'optimisation sont considérés comme des méthodes et des outils qui permettent de résoudre les problèmes d'optimisation afin de trouver des solutions optimales et de haute qualité.

Au fil des ans, des nombreuses méthodes ont été proposées pour résoudre des problèmes des difficultés diverses, dont les problèmes d'ordonnancement de production.

Ces derniers consistent à assigner un ensemble des tâches à un groupe des ressources en respectant les contraintes imposées.

Les chercheurs ont donc déployé des grands efforts pour améliorer les performances des méthodes de résolution qu'ils ont proposées.

Afin de simplifier la tâche, ils ont divisé ces méthodes en deux catégories principales en fonction de leur degré de difficulté et de complexité pour chaque problème.

2 Les méthodes d'optimisation

Il existe plusieurs méthodes pour résoudre les problèmes d'optimisation nous le mentionnons dans l'organigramme suivant

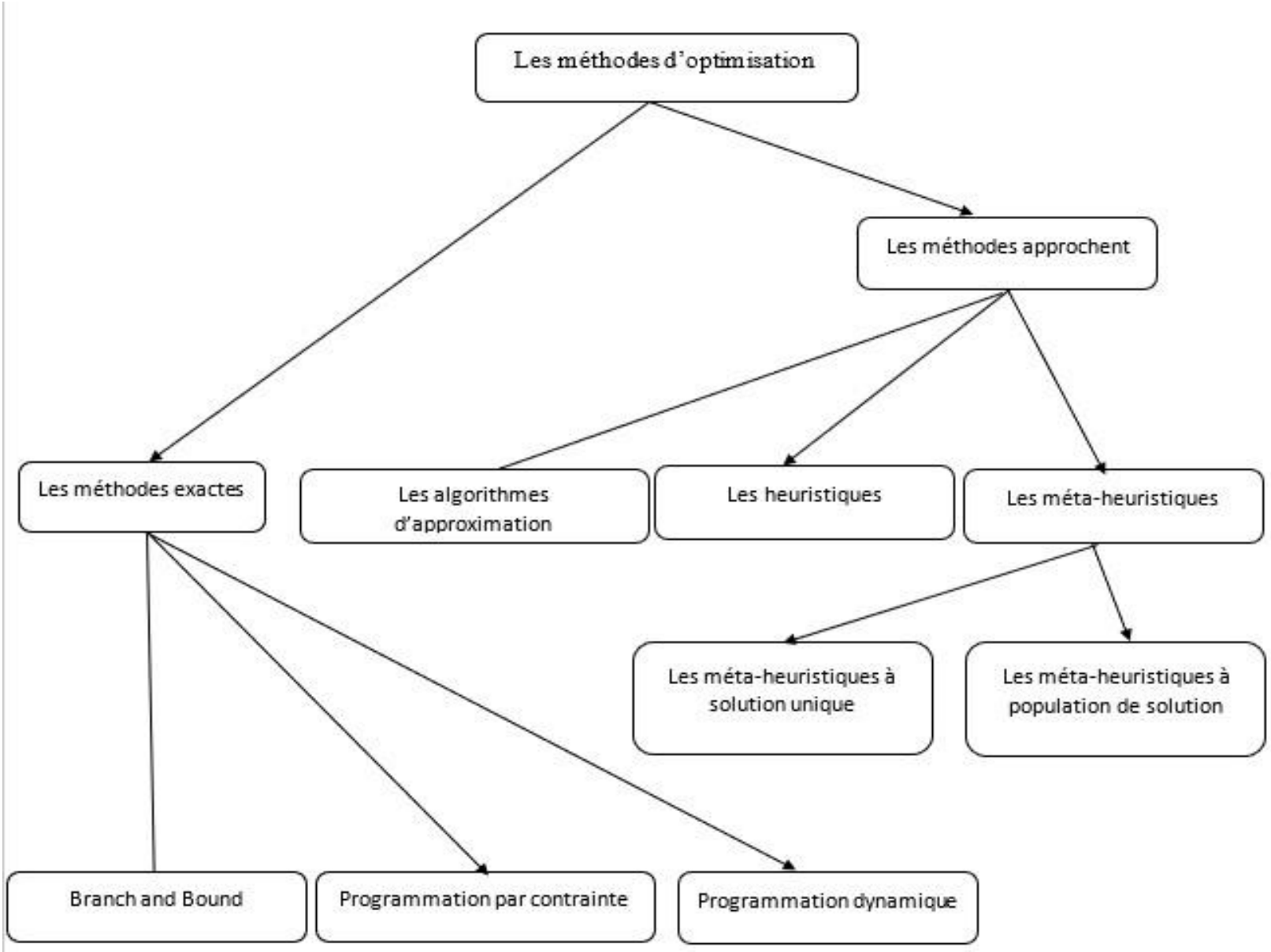

Figure 2.1 : L'organigramme présenter les méthodes d'optimisation

Nous pouvons dire que les méthodes d'optimisation sont divisées en deux grandes catégories : les méthodes exactes et les méthodes approchées

2.1 Les méthodes exactes

Les méthodes exactes sont des approches qui peuvent fournir une solution optimale pour un problème d'optimisation, mais elles sont très gourmandes en ressources et nécessitent des temps d'exécution considérables.

Une méthode courante pour résoudre les problèmes d'optimisation discrets consiste à énumérer toutes les solutions admissibles, mais cette méthode peut s'avérer impossible pour les problèmes de grande taille en raison des temps de calcul prohibitifs qu'elle entraîne. Par conséquent, l'utilisation de méthodes exactes est généralement plus efficace pour résoudre des problèmes de petite taille. (NOUIRI, 2017)

Les méthodes exactes utilisent surtout deux approches de résolution très connues : la programmation dynamique et les procédures par séparation et évaluation.

Ces méthodes sont souvent utilisées pour résoudre les problèmes combinatoires de manière exacte, en ordonnancement tout particulièrement.

Ce sont des méthodes d'énumération implicite l'énumération explicite construit toutes les solutions réalisables et retient une parmi les meilleures. (CHERGUI & DAHMANI, 2017)

L'énumération implicite consiste à explorer l'ensemble des toutes les solutions réalisables en éliminant des sous-ensembles des solutions moins intéressantes sans avoir à les construire. (CHERGUI & DAHMANI, 2017)

Nous pouvons citer trois approches particulièrement célèbres : La programmation dynamique introduite par Bellman dans les années 50, la méthode par séparation et évaluation (Branch and Bound en anglais : notée B&B) et l'algorithme de retour arrière (Backtracking). (CHERGUI & DAHMANI, 2017)

2.1.1 Programmation dynamique

La programmation dynamique est une méthode puissante pour résoudre les problèmes d'optimisation combinatoire dont la fonction objective peut être décomposée en parties. Elle a été initiée par Richard Bellman dans les années 1950 pour résoudre le problème du plus court chemin dans un graphe. Elle est également largement utilisée pour résoudre d'autres problèmes qui présentent certaines caractéristiques et qui peuvent être résolus à l'aide d'une formule de récurrence. L'efficacité de cette méthode repose sur le principe d'optimalité de Bellman, qui stipule que "toute politique optimale est composée de sous-politiques optimales". (LARIBI, 2018)

La programmation dynamique est un ensemble d'outils et d'algorithmes mathématiques pour étudier les processus de décision séquentiels et calculer les stratégies optimales (exactes ou approximatives). Elles consistent en des règles de prise de décision qui, pour chaque situation possible (état du système), nous indiquent quelles décisions (ou actions) prendre afin d'optimiser une fonction objective globale. Souvent, cette fonction objective est une espérance mathématique. Parfois, la politique optimale peut être caractérisée par des théorèmes (théorie) et souvent, elle peut être calculée ou approchée, mais dans certains cas, sa résolution peut être trop difficile. (Bastin, 2014)

2.1.2 Programmation par contrainte

La programmation par contraintes est une méthode apparue dans les années 80 qui vise à résoudre les problèmes de programmation logique et d'intelligence artificielle. Cette technique consiste à modéliser les problèmes en utilisant un ensemble des relations et des contraintes logiques, qui peuvent prendre différentes formes telles que des relations mathématiques ou des fonctions logiques. Les contraintes limitent l'ensemble des valeurs que peuvent prendre simultanément plusieurs variables. Ainsi, une contrainte c ∈ C sur les variables xi1, ..., xik pour i1, …, ik ∈ {1, …, n} est une relation mathématique entre ces variables. Les variables qui sont impliquées dans la contrainte c sont désignées par var(c). (ACHOURI , 2022)

La programmation par contraintes repose sur la notion centrale de problème de satisfaction des contraintes.

Défini comme un tuple P = hX, D, Cij où :

X est un ensemble fini des variables ;

D est une fonction associant à chaque variable X ∈ X son domaine D(X) ;

C'est un ensemble fini des contraintes d'arrêt finies sur X.

Les variables sont simplement des identificateurs pouvant être utilisés dans la spécification des contraintes, et sont associées à un domaine. Dans le cadre de la programmation par contraintes, le domaine des variables est généralement supposé être fini, donc discret. (Xavier, 2014)

Pour simplifier les notations et sans perte de généralité, nous supposons que les domaines des variables sont des sous-ensembles finis de nombres entiers, c'est-à-dire que pour toute variable X ∈ X, le domaine D(X) de X est un sous-ensemble de Z. Bien que cette hypothèse soit naturelle si les contraintes sont des contraintes arithmétiques sur les nombres entiers, il existe d'autres types de contraintes, et en particulier des contraintes sur d'autres types de variables, tels que les nombres flottants , les ensembles , les arbres ou les graphes .Il est donc préférable de considérer la programmation par contraintes sur ces types de variables comme des adaptations des techniques de la programmation par contraintes sur les variables entières, et de considérer que la programmation par contraintes sans mention explicite du type de variable considéré se réfère à la programmation par contraintes sur les variables entières. (Xavier, 2014)

Enfin, une contrainte est une spécification d'un sous-ensemble du produit cartésien des domaines d'un sous-ensemble de variables, qui correspond à l'ensemble des modèles d'une formule d'une logique appropriée. (Xavier, 2014)

2.1.3 Branch and Bound

L'algorithme Branch-and-Bound est une méthode qui consiste à construire un arbre d'énumération des solutions potentielles d'un modèle IP de manière intelligente, en utilisant des bornes inférieures et supérieures pour éviter de générer tous les nœuds de l'arbre. La tâche principale de cet algorithme est de diviser répétitivement le problème

initial en problèmes sous-jacents plus petits et plus faciles à résoudre, c'est ce qu'on appelle la phase de "branching". La phase d'évaluation, ou de "bound", consiste à identifier les sous-ensembles qui peuvent contenir la solution optimale et à supprimer ceux qui ne la contiennent pas. (Grainia, 2015)

L'algorithme Branch-and-Bound construit et parcourt l'arbre de recherche. Sa racine est le domaine réalisable du problème initial, le modèle IP, tandis que les nœuds représentent les domaines réalisables des sous-problèmes. (Grainia, 2015)

2.2 Les méthodes approchées

Comme nous l'avons mentionné précédemment, les méthodes exactes nécessitent des temps de traitement prohibitifs pour résoudre les problèmes de grande taille.

Afin d'obtenir malgré tous des solutions de qualité, des méthodes approchées ont été développées. Ces méthodes donnent des solutions sous-optimales, mais en un temps de calcul raisonnable. (MAMOUNI & OULD MOHAMED, 2018)

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si l'optimalité n'est pas primordiale, c'est-à-dire, trouver une solution de bonne qualité en un temps de calcul raisonnable sans garantir l'optimalité de la solution obtenue. Certains problèmes demeurent hors de portée des méthodes exactes. Les méthodes approchées, dites aussi d'approximation, constituent une alternative intéressante pour traiter les problèmes d'ordonnancement de grande taille. Elles ne garantissent pas toujours l'obtention de solutions optimales de haute qualité.

Les méthodes approches sont des méthodes d'amélioration ou d'optimisation qui visent à trouver une solution praticable pour un problème cible en peu de temps.

Ce qui distingue ces méthodes, c'est qu'elles peuvent être appliquées à n'importe quel problème, qu'il soit difficile ou facile. Les algorithmes d'amélioration ont prouvé leur puissance et leur efficacité dans la résolution de problèmes d'optimisation complexes. (CHERGUI & DAHMANI, 2017)

Les méthodes approchées sont divisées en deux catégories principales : les heuristiques et les métaheuristiques.

La caractéristique qui distingue les métaheuristiques est leur capacité à être appliquée à de nombreux problèmes, tandis que les heuristiques sont destinées à résoudre un problème spécifique, cela signifie que les métaheuristiques peuvent être utilisées pour résoudre différents types de problèmes, tandis que les heuristiques sont généralement utilisées pour résoudre un problème spécifique

2.2.1 Les heuristiques

Le terme heuristique est d'origine grecque et signifie "servant à la recherche et à l'exploration". Einstein a inclus ce terme dans son article sur la physique quantique, qui a remporté le prix Nobel en 1905. Il a souligné que cette approche qu'il présentait était incomplète, mais utile d'une certaine manière. (Gigerenzer & Gaissmaier, 2011)

Le mot "heuristique" est un terme qui fait référence à des principes et des méthodes utilisés pour identifier et résoudre des problèmes simples et à multiples solutions, et pour choisir la solution la plus efficace. Les heuristiques représentent un équilibre entre la nécessité de simplifier les critères et en même temps de créer une distinction entre les bonnes et les mauvais choix. L'efficacité de ces méthodes dépend de leur solidité et de leur capacité à utiliser les connaissances disponibles pour le problème. (MAMOUNI & OULD MOHAMED, 2018)

Les heuristiques sont des méthodes qui visent à obtenir des solutions de qualité en peu de temps et de manière raisonnable, sans être en mesure de garantir l'optimalité. Ces méthodes reposent sur des règles simplifiées pour améliorer un ou plusieurs critères. Leur principe consiste à intégrer des stratégies de décision pour construire une solution proche de l'optimalité. (LARIBI, 2018)

De tout cela, nous pouvons citer plusieurs exemples d'heuristiques utilisées dans l'ordonnancement de la production, notamment SPT, LPT, WSPT, ALGORITHME DE JOHNSON… etc.

- SPT : Cette heuristique consiste à sélectionner la tâche nécessitant le moins de temps pour la réaliser en premier dans le tableau de production
- LPT : Cette heuristique consiste à sélectionner la tâche nécessitant le plus de temps pour la réaliser en premier dans le tableau de production.
- WSPT : Cette heuristique utilise un facteur de priorité pondéré pour sélectionner les tâches à exécuter en ce moment.
- ALGORITHME DE JONHSON : Cette heuristique analyse les tâches à exécuter dans le tableau de production, identifie les tâches nécessitant le temps le plus court et celles nécessitant le temps le plus long pour être complétées, puis établit l'ordre des tâches dans le tableau.

2.2.2 Les métaheuristiques

Le terme métaheuristique est composé de deux mots : "méta", qui signifie "au-delà" ou "dans un niveau supérieur", et "heuristique ", qui signifie "trouver quelque chose".

Souvent, la métaheuristique est définie comme une procédure qui exploite au mieux la structure du problème considéré, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible. (BOUMEDIENE, 2020)

Les métaheuristiques sont des algorithmes inspirés de la nature utilisés pour résoudre des problèmes de grande taille et complexes. Elles exploitent généralement des processus aléatoires dans l'exploration de l'espace de recherche pour faire face à l'explosion combinatoire engendrée par l'utilisation de méthodes exactes (HACHIMI, 2013)

La métaheuristique est un processus ou une méthode qui guide et modifie les opérations de l'heuristique subordonnée pour produire efficacement des solutions de haute qualité. (BOUMEDIENE, 2020)

On peut manipuler une solution unique complète (ou incomplète) ou un ensemble de solutions à chaque itération, les heuristiques subordonnées peuvent être de haut niveau ou de bas niveau. (BOUMEDIENE, 2020)

Les métaheuristiques sont divisés en deux catégories principales : les métaheuristiques à solution unique et les métaheuristiques à population de solution.

2.2.2.1 Les métaheuristiques à solution unique

Les métaheuristiques à solution unique, également connues sous le nom de méthodes de recherche locale ou de méthodes de trajectoire, sont l'une des méthodes de résolution de problèmes d'optimisation. Elles sont basées sur le concept de voisinage, qui représente un ensemble de solutions obtenues à partir d'une solution donnée en effectuant un certain nombre de transformations. Le but de ces transformations locales est d'explorer le voisinage de la solution courante afin d'améliorer progressivement sa qualité au cours des différentes itérations. Cette technique vise à trouver à la fois des solutions locales et des solutions globales de qualité élevée. (LARIBI, 2018)

Il y a plusieurs métaheuristiques qui ont une solution unique, voici quelques exemples : la recherche taboue et recuit simuler

2.2.2.1.1 La recherche taboue

L'algorithme de recherche Tabou est l'un des algorithmes qui repose sur la recherche locale et qui utilise une liste pour stocker les solutions améliorées. Cette méthode a été proposée par Glover en 1986. Le principe de base de cette méthode consiste à utiliser une liste pour éviter de choisir une solution précédemment visitée ou de boucler en passant par une solution déjà visitée, afin d'échapper, dans une certaine mesure, aux optima locaux. Cette liste, appelée "liste Tabou", peut avoir une taille fixe ou variable. La taille de la liste Tabou est un facteur important : si elle est trop petite, l'algorithme ne pourra pas s'échapper de certaines zones d'attraction d'optima locaux, tandis que si elle est trop grande, la qualité des solutions obtenues sera affectée, car le voisinage en chaque point sera très clairsemé. Dans les premières implémentations, les

chercheurs ont préconisé une taille de 7 éléments. (Hanset, Fei, Roux , Duvivier , & Meskens, 2007)

L'algorithme de recherche tabou repose sur deux principes : l'utilisation du concept de voisinage et l'utilisation de la mémoire directionnelle. Il ne s'arrête pas à la limite S, mais recherche dans le voisinage la solution la plus optimale et conserve toujours la meilleure solution. Il examine un échantillon de solutions voisines de l'état actuel et donne au processus de recherche d'autres opportunités d'explorer l'espace de recherche pour trouver la solution globale. Les solutions de qualité peuvent être faibles, mais à côté d'elles, il y a de bonnes solutions. Par conséquent, la recherche est orientée vers le meilleur, Cependant, il est possible de revisiter les solutions déjà explorées, et pour surmonter ce problème, les exemples de recherche tabou proposent l'utilisation d'une mémoire appelée liste tabou pour stocker les dernières solutions les plus efficaces et ne pas les revisiter.

Algorithme 2: 1 Algorithme de recherche tabou

1-Inisialisation

-Solution initiale $s_0. s^* \leftarrow s_0, c^* \leftarrow f(s_0)$.

-k ← 0 , Liste Tabou = Ø

2-Répéter (critère ¬vérifié)

-Voisins $V(s_k)$de s_k

-Choisir une solution, le meilleur qui minimise f et qui n'appartient pas à liste tabou, meilleur (s_k).

-$s_{k+1} \leftarrow$ meilleur (s_k)

-Si $(c(s_{k+1}) < c^*)$ alors $s^* \leftarrow s_{k+1}$, $c^*(s_{k+1})$.

-Mise à jour de liste tabou

2.2.2.1.2 Recuit simulé

Recuit simulé est un algorithme basé sur l'approximation de la limite optimale global de la fonction objective. Cette méthode a été dérivée de la technologie de fusion des métaux, qui permet de réduire les défauts des matériaux en augmentant leur

température, puis en contrôlant le refroidissement à l'aide de la chaleur extérieure (BERKANI, 2013).

Le refroidissement naturel ne peut pas atteindre la configuration optimale des atomes pour obtenir la meilleure stabilité à chaque cycle de fusion simulée. L'algorithme remplace la solution libre par une autre solution générée par la mutation. La solution est acceptée avec un taux de probabilité qui dépend de la différence de coût entre les deux solutions et du coefficient de température.

La température diminue progressivement au fil du temps selon une loi prédéfinie. La probabilité de détérioration de la solution au cours du cycle est souvent élevée. Cette stratégie évite de tomber dans la solution locale la plus basse. (BERKANI, 2013)

L'algorithme du Recuit Simulé consiste en plusieurs étapes qui se concentrent sur le principe de la technique d'analyse froide aléatoire afin de générer une séquence de configurations qui convergent vers l'équilibre thermique. Les étapes principales sont les suivantes :

- Choisissez une température de départ T=T0 et une solution initiale S=S0
- Générez une solution aléatoire dans le voisinage de la solution actuelle
- Comparez les deux solutions selon le critère de Metropolis
- Répétez les étapes 2 et 3 jusqu'à ce que l'équilibre statistique soit atteint
- Réduisez la température et répétez les étapes 2 à 4 jusqu'à ce que le système soit gelé.

Dans l'algorithme du Recuit Simulé, nous commençons par une configuration donnée et nous appliquons des modifications élémentaires dessus. Si cette perturbation a pour effet de diminuer la fonction objective du système, elle est acceptée. Sinon Il est accepté avec une probabilité exp (ΔE/T) lors de l'application répétée de cette règle (Baptiste , 2006)

Algorithme 2: 2 Algorithme de recuit simulé

1-s ← Initialiser solution initial
2- T← T_{max}
3- Répéter {Processus de refroidissement}
4- Répéter {répéter pour chaque valeur de T}
5- $s' \leftarrow$ Générer voisin (s)
6- $\Delta E \leftarrow f(s') - f(s)$
7- **si** $\Delta E \leq 0$ **alors**
8- $s \leftarrow s'$
9- **sinon**
10- $P \leftarrow e^{\frac{-\Delta E}{T}}$
11- $s \leftarrow s'$ avec probabilité P
12- **fin si**
13- **jusqu'à** condition d'arrêt {équilibre thermique}
14- T ← g(T) {Baisser la température (refroidissement)}
15- **jusqu'à** $T < T_{min}$

2.2.2.2 Les métaheuristiques à population de solution

Les algorithmes à population de solutions sont des méthodes d'amélioration de performances qui se basent sur le développement simultané de plusieurs solutions. Ces algorithmes travaillent sur un ensemble de solutions possibles appelé génération et les améliorent de manière répétitive. La plupart des algorithmes à population de solutions sont inspirés de la nature, comme les algorithmes génétiques. Ils peuvent également être construits sur des modèles mathématiques tels que les algorithmes de simulation de particules. Le but de l'utilisation des mécanismes de sélection, de reproduction et de mutation est d'améliorer les solutions.

Parmi ces algorithmes, nous mentionnons ces trois-ci en détail : les algorithmes génétiques, algorithme de chameau et la recherche dispersée

2.2.2.2.1 Algorithme génétique

Les algorithmes génétiques font partie de la famille des algorithmes évolutionnaires. Ils s'inspirent de l'évolution naturelle des espèces. Avec ce type de

méthodes, il ne s'agit pas de trouver une solution analytique exacte, mais de trouver une bonne solution satisfaisante dans un temps de calcul raisonnable. (Chaari, 2010)

L'algorithme génétique repose sur plusieurs étapes, la première consiste à créer une génération initiale composée d'un groupe d'individus. Ces individus sont manipulés à l'aide de l'opérateur de croisement, qui transforme les gènes des parents en gènes des enfants, ainsi qu'à l'aide de l'opérateur de mutation, qui modifie les résultats avec une probabilité de mutation P mut. La création d'une nouvelle génération d'individus par les phases de sélection et de recombinaison (croisement et mutation) permet de générer des individus plus forts que ceux de la génération précédente. Les individus issus de la phase de recombinaison sont insérés dans une nouvelle population en utilisant une méthode d'insertion, et la valeur de la fonction objective est évaluée pour chaque individu. La force des individus de la population augmente de génération en génération et un test d'arrêt est effectué pour décider quand arrêter l'algorithme. La figure 2.1 présente un schéma de fonctionnement général de l'algorithme génétique. Les différentes étapes de ce dernier sont présentées en détail dans les sections suivantes. (Chaari, 2010)

Algorithme 2: 3 Algorithme génétique

1. **Initialiser** une population (***n*** individus)
2. **Calculer** le degré d'adaptation de chaque individu
3. **Tant que** non fini ou non convergence
    i. **Sélectionner** les $c \times n$ meilleurs individus, les apparier et effectuer un **croisement** pour obtenir les nouveaux individus
    ii. Chacun des ***l*** caractères des nouveaux individus **mute** avec une probabilité ***m***
4. Ne conserver que les ***n*** meilleurs individus
5. **Calculer** le degré d'adaptation de chaque individu
6. **Fin Tant que**

Avec

n = taille de la population

c = taux de croisement

l = nombre de caractères de chaque individu

m = taux de mutation

2.2.2.2.2 La recherche dispersée

La méthode de La recherche dispersée diffère des autres algorithmes évolutifs tels que les algorithmes génétiques en utilisant une conception stratégique plutôt que l'aléatoire. Cette méthode fonctionne en utilisant un ensemble des solutions connues sous le nom d'ensemble de retour, qui sont combinées pour créer de nouvelles solutions. Contrairement aux "populations" dans les algorithmes génétiques, l'ensemble de retour dans la méthode de La recherche dispersée est relativement petit. (Jourdan, 2003)

L'algorithme de La recherche dispersée comprend les cinq méthodes suivantes : (Jourdan, 2003)

- La méthode de génération de diversité pour produire une variété de solutions en utilisant une solution de test arbitraire (ou la première solution) comme entrée.
- La méthode d'amélioration pour transformer la première solution en une ou plusieurs solutions améliorées.
- La méthode de mise à jour de l'ensemble de retour pour construire et maintenir un ensemble de retour contenant les meilleures solutions disponibles, organisé pour fournir un accès efficace à d'autres parties de l'algorithme. Les solutions sont introduites dans l'ensemble de retour en fonction de leur qualité ou de leur diversité.
- La méthode de génération de sous-ensemble pour travailler sur l'ensemble de retour, produisant un sous-ensemble de ces solutions comme base pour créer des solutions combinées.

- La méthode de combinaison des solutions pour transformer un sous-ensemble spécifique de solutions produites par la méthode de génération de sous-ensemble en un ou plusieurs vecteurs combinés de solutions.

Le principe de la méthode de la recherche dispersée

L'algorithme de recherche dispersée repose sur le principe de diviser l'ensemble des solutions en deux groupes : l'ensemble de référence où sont placées les solutions préférées, et l'ensemble des solutions aléatoires où se trouvent les solutions moins préférées. Chaque individu est amélioré en appliquant une recherche locale, et à la fin, des solutions sont régénérées en remplaçant les solutions éloignées par les solutions préférées. Le processus se poursuit jusqu'à ce qu'un critère d'arrêt soit satisfait. L'un des points importants est la mesure de la diversité des solutions, qui doit être effectuée dans l'espace des solutions (et non dans l'espace des objectifs) et doit refléter la différence entre deux solutions. (HOUARI , 2012)

Algorithme 2: 4 Algorithme de recherche dispersé

1. Initialiser : générer une population initiale P de solutions
2. Mettre les meilleures solutions trouvées dans une population de référence R
3. Tant que le critère d'arrêt non satisfait (nombre d'itérations) faire
    4.A=R → prendre la population de référence
    5.Tant que A≠ faire
        6.Combiner les solutions (B←R×A)
        7.Améliorer les solutions B
        8.Mettre à jour les solutions R
        9.Garder les meilleures solutions à partir de R B
        10.A←B−R
    11.Fin tant que
    12.Supprimer mauvaise solution dans R
    13. Ajouter de nouvelles solutions diverses dans R

14. Fin tant que

2.2.2.2.3 Algorithme de chameau

L'état de l'art : En 2016 (Khalid Ibrahim & Ramzy , 2016) à publier le premier article sur un nouvel algorithme inspiré du comportement des chameaux.

L'algorithme du chameau est une nouvelle métaheuristique inspirée du mouvement et du comportement des chameaux lorsqu'ils se déplacent, de sorte que plusieurs facteurs sont considérés pour déterminer les actions de l'algorithme, notamment : (Khalid Ibrahim & Ramzy , 2016)

- L'effet de la température.
- L'approvisionnement (eau et nourriture).
- L'endurance du chameau.
- La portée de la visibilité (et/ou de l'ouïe) du chameau.
- La marche aléatoire.
- L'effet de groupe (multi-solution).
- La condition de terminaison (mort ou retour).
- Les conditions du terrain (oasis, sables mouvants, tempêtes, etc.).
- Les limitations (vitesse maximale, âge et poids supportable).

En 2017, (Hasanen & Zied , 2017) ont publié un article sur leur recherche sur l'application des algorithmes de chameaux dans les problèmes d'ordonnancement de production. Ils ont expliqué le principe de cet algorithme, qui consiste à déterminer le nombre de troupeaux de chameaux, chaque troupeau représentant une solution. Un troupeau est composé d'un groupe de chameaux et un leader est choisi pour chaque troupeau afin de diriger les autres chameaux dans la recherche d'une solution dans le désert. Chaque leader commence à partir d'un point différent dans le désert, ce qui donne une diversité des solutions au problème.

Après l'initialisation des paramètres, l'algorithme lance les troupeaux dans l'espace du problème. Le chef commence par son état initial, envoie les autres chameaux pour trouver les voisins. Le chef examine chaque voisin donné par les chameaux et traite l'humidité élevée à travers la fonction du meilleur voisin qui dépend du facteur d'humidité. (Hasanen & Zied , 2017)

En 2019, (Ramzy, Falih M. , & Abdulkareem , 2019) a publié un nouvel article dans lequel il a décrit certaines modifications qu'il a apportés aux algorithmes de chameaux. Ramzy a présenté la version modifiée de l'algorithme de voyage du chameau (MCA) avec une structure simple qui améliore considérablement sa convergence et sa vitesse de calcul.

En 2021 (Ramzy , Jawad , & Hussein, 2021) ont publié une nouvelle version modifiée de l'algorithme du chameau. Ils ont validé la méthode proposée en utilisant des défis de systèmes de distribution d'énergie Dans cette section, une nouvelle version de l'algorithme MCA (NMCA) est présentée. Le NMCA est utilisé pour trouver la configuration optimale (taille, position et nombre) des unités PV dans le réseau de distribution pour résoudre le problème de congestion, améliorer le profil de tension et réduire les pertes de puissance. Le NMCA fonctionne de la même manière que le MCA, avec les modifications suivantes :

- Modification de l'étape de mutation.
- Modification de l'équation de localisation.
- Ajout de l'équation de vitesse.
- Ajout des limites de vitesse.
- Ajout des limites de localisation.

En 2022, (Utama, Safitri, & Garside, 2022) ont appliqué l'algorithme de chameau modifiée dans le domaine de la logistique pour améliorer les itinéraires de résolution du problème de la consommation de carburant. L'objectif des chercheurs était de résoudre le problème de la distribution de véhicules verts en améliorant l'itinéraire, et

les véhicules verts ont été utilisés comme objectif dans cet algorithme. Le problème était de réduire le coût total de la distribution, y compris le coût de la consommation de carburant et le coût du retard de livraison. L'expérience a été menée en utilisant l'algorithme de chameau pour déterminer l'impact de ces variables sur le coût de la distribution, et la performance de l'algorithme de chameau a été comparée à celle de l'algorithme de recherche locale, de l'amélioration de l'essaim de particules et de l'amélioration des colonies. Les résultats de ces comparaisons ont montré que l'algorithme de chameau a donné des meilleurs résultats que les autres algorithmes.

Principe d'algorithme : (Khalid Ibrahim & Ramzy , 2016) L'algorithme de chameau est basé sur des étapes principales

La première étape consiste à préparer et définir les paramètres de base de l'algorithme tels que $Tmin$ et $Tmax$, le nombre de chameaux dans le convoi et les limites de vision des chameaux, ainsi que la position de chaque chameau qui est déterminée au hasard.

La deuxième étape est la transformation et l'évaluation, où la position de chaque chameau est convertie en une séquence de voyage en utilisant la technique LRV. Ensuite, la qualité de chaque chameau est évaluée en utilisant une fonction de forme physique et la meilleure position est déterminée.

La troisième étape est la recherche et l'amélioration, où les mouvements des chameaux sont simulés en fonction de la température et de leur capacité à avancer. Il est déterminé si les conditions permettent ou non le mouvement des chameaux dans leur position actuelle. En cas d'impossibilité de se déplacer, le mouvement est effectué au hasard. En cas de capacité à se déplacer, le mouvement est effectué vers la meilleure position disponible.

Pour éviter toute confusion, nous utilisons dans notre travail les abréviations T, S et E pour se référer respectivement à la température, à l'alimentation et à l'endurance. La température est considérée comme le principal facteur aléatoire qui affecte le voyage du chameau itinérant et à un impact sur son endurance. De plus, ce facteur peut varier d'un

chameau à l'autre car chaque animal se déplace et explore une zone différente dans le désert. Ainsi, pour chaque chameau j, la température instantanée peut être exprimée par T_{actuel}. (Khalid Ibrahim & Ramzy , 2016)

La température instantanée T(actuel) peut être exprimé comme :

$$T_j\ actuel = (T_{max} - T_{min}) * Rand(0,1) + T_{min}$$

- T_{max} : La température maximale
- T_{min} : La température minimale

La disponibilité d'eau et la nourriture est un facteur critique qui varie de manière inversement proportionnelle à la durée du voyage. L'approvisionnement restant S_{actuel} peut-être décrit comme une fonction décroissante pour chaque chameau j :

$$S_{j\ actuel} = S_{j\ précedent} * (1 - w * \frac{étape\ journeé}{toutes\ les\ étapse\ de\ jours})$$

Avec w le facteur de charge ∈ (0,1)

L'endurance des chameaux est affectée par la température et la durée du voyage, ce qui peut être représenté pour chaque chameau j par une fonction décroissante.

$$E_{j\ actuel} = E_{j\ précedent} * (1 - \frac{T_{actuel}}{T_{max}}) * (1 - \frac{étape\ journeé}{toutes\ les\ étapse\ de\ jours})$$

À la première étape du voyage, le E précédentes égal à E initiale qui indique l'endurance totale initiale et Après chaque étape du voyage, l'endurance sera régulièrement mise à jour comme :

E ancien j= E actuel j

Algorithme 2: 5 Algorithme des chameaux

Créer une première caravane de chameaux (multi-solutions).

Calculer la forme physique individuelle de la caravane de chameaux et trouvez le meilleur actuel.

Tant que (compteur <nombre total d'étapes du trajet)

Pour i = 1 : Caravane de chameaux

Calculer $Tactuel(j)$, $Sactuel(j)$, $Eactuel(j)$ dans la Caravane.

Mettre à jour les emplacements des chameaux

Décider de l'acceptation de nouveaux chameaux emplacements (dépendent de la qualité de la solution mesurée par la fonction et la plage de fitness limitation).

Si (une condition d'oasis se produit)

Mettre à jour l'endurance

Fin si

Classer les individus Caravane de chameaux et trouver la meilleure solution

Fin pour i

Fin tant que

Garder la meilleure solution

3 Conclusion

Dans ce chapitre, nous avons défini les problèmes d'optimisation et leurs méthodes de résolution. De tout ce que nous avons abordé, nous pouvons conclure qu'il y a beaucoup des méthodes d'optimisation et des algorithmes qui permettent de résoudre les problèmes d'optimisation avec grande difficulté tels que les problèmes d'ordonnancement

de la production. Nous avons ainsi cité quelques algorithmes à solution unique, tels que l'algorithme de recuit simulé et l'algorithme de recherche tabou. Nous avons également appris les algorithmes à population des solutions, tels que l'algorithme génétique, l'algorithme de recherche dispersée et l'algorithme des chameaux. Tout cela nous a aidés à trouver une méthode pour résoudre le problème posé à savoir l'ordonnancement de la production dans un atelier flow shop.

III. Chapitre 3 : Adaptation des métaheuristiques sur le problème d'atelier flow shop

1. Introduction

Grâce à ce que nous avons abordé dans les chapitres précédents, la définition de l'ordonnancement de la production et leurs problèmes, ainsi que la définition des méthodes de résolution des problèmes d'optimisation, y compris la définition des métaheuristiques, il est clair que les problèmes complexes de l'ordonnancement peuvent être résolus en appliquant certains algorithmes qui peuvent nous donner des solutions améliorées.

À travers tout cela, il est clair que l'application des métaheuristiques aux problèmes d'ordonnancement nécessite une adaptation spécifique à chaque problème.

Cela comprend une formulation claire du problème, une représentation appropriée des solutions et la définition d'une fonction d'évaluation adéquate, en ajustant les paramètres de la métaheuristique et en répétant les processus, il est possible d'obtenir des solutions optimales pour les problèmes complexes de l'ordonnancement.

Dans ce chapitre, nous aborderons la résolution du problème d'ordonnancement de la production en appliquant plusieurs algorithmes tels que l'algorithme des chameaux, l'algorithme génétique et l'algorithme hybride combinant l'algorithme des chameaux et l'algorithme de recherche dispersée. Enfin, nous comparerons les résultats obtenus et choisi les solutions optimales.

2. Présentation de problème

2.1. Problématique

Nous avons étudié dans notre travail le problème d'ordonnancement concernent la minimisation de makespan $Cmax$ dans un atelier flow shop, (Cmax ou makspan est le temps de sortie de la dernière pièce de la dernière machine), le nombre des machines et des jobs plus de cinq ou la complexité de problème NP-difficile.

Dans un atelier flow shop on a des machines on série et l'exécution des tâches dans ce système un par un et le temps opératoire de tous les jobs sont connus auparavant, et chaque job ne peut être traité par une machine que lorsque celle-ci fini le traitement du job précédent

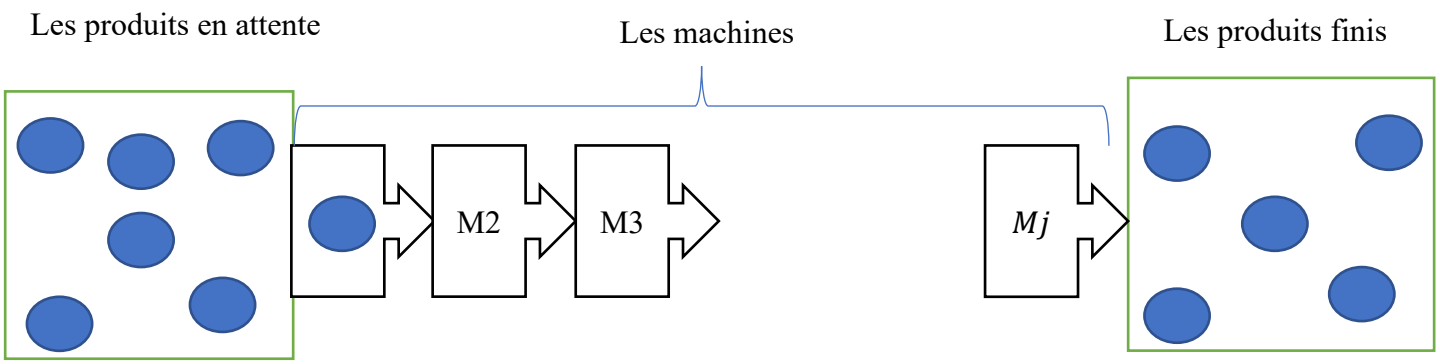

Figure 3: 1 Cheminement des produits dans un atelier flow shop

m : les nombres des machines

n :les nombres des jobs

Cij : la date de fin de traitement de job i dans la machine j

Pij : le temp de traitement de job i dans la machine j

Fm: les type atelier flow shop

Sij : le temp de début de traitement de job i dans la machine j

La fonction objective : $Fm|\ |Cmax \qquad f = min(Cmax)$

$$Cmax = max(Cij)\ \forall\ i = 1 \dots n\ et\ j = 1 \dots m$$

$$Cij = Sij + Pij\ \forall\ i = 1 \dots n\ et\ j = 1 \dots m$$

3. Adaptation et application de l'algorithme génétique

Dans notre travail nous appliquons l'algorithme génétique pour résoudre le problème de minimisation de $Cmax$ dans un atelier flow shop et pour ce faire il faut tout d'abord adapter l'algorithme à notre problème.

3.1. Le principe de l'algorithme génétique

L'algorithme génétique repose sur trois principes :

- Le principe de variation : Chaque individu au sein d'une population est unique, que ces différences soient grandes ou petites, elles seront décisives dans le processus de sélection. (nadir, 2019)
- Le principe d'adaptation : Les individus les plus adaptés à leur environnement ont plus de chances de survie et de reproduction. Ce sont ceux qui sont le mieux adaptés à leur environnement qui ont les meilleures chances de survie et de reproduction. (nadir, 2019)
- Le principe d'hérédité : Ce principe repose sur le fait que les traits des individus doivent être héritables à travers les générations pour partager les caractéristiques bénéfiques à la survie. (nadir, 2019)

3.2. Les opérateurs d'algorithme génétique (nadir, 2019)

- La sélection consiste à choisir les individus les mieux adaptés afin d'avoir une population de solution la plus proche de convergence vers l'optimum global,
- Le croisement : Le croissement est le résultat obtenu lorsque deux chromosomes partage leurs particularités.
- La mutation : La mutation consiste à altérer un gène dans un chromosome selon un facteur de mutation

3.3. Le fonctionnement de l'algorithme génétique (TAMRABET, 2018)

- Initialisation : Au départ, on tire aléatoirement une population initiale de N chromosomes.
- Evaluation : Après avoir sélectionné la population initiale, chaque chromosome est soumis à un processus de décodage suivi d'une évaluation
- Sélection : Une nouvelle population de chromosomes est créée en utilisant une méthode de sélection appropriée, où les chromosomes sont regroupés par paires.
- Croisement et mutation : Au sein de la nouvelle population, il est possible d'effectuer des croisements et des mutations
- Retour : On retourne à la phase d'évaluation (étape 2) tant que la condition du problème n'est pas satisfaite, ou tant que le critère d'arrêt n'est pas atteint

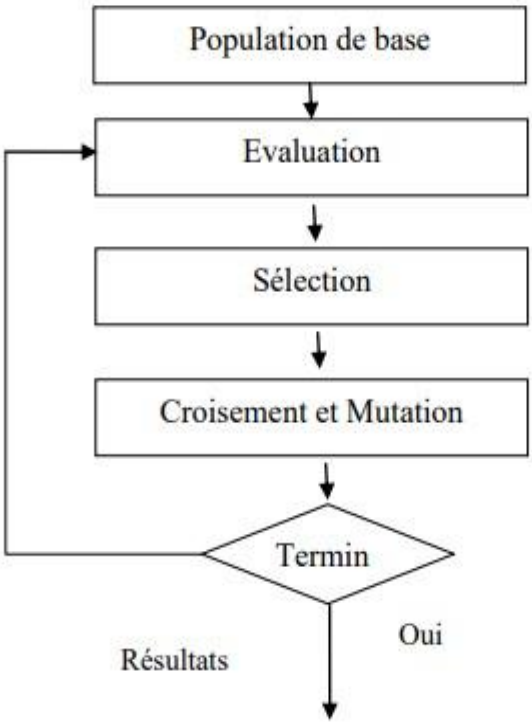

Figure 3 : 2 Schéma présenter le fonctionnement de AG

3.4. Adaptation de l'algorithme génétique

Dans notre système nous supposons que les jobs sont des gènes, la séquence sont des individus et Cmax représente la fonction d'évaluation.

- Population initiale : par rapporte le système étude
- Nombre des individue : n=100
- La sélection : méthode de sélection par roulette
- Facture de croisement : c=0,5
- Facture de mutation : m= 0,05

Algorithme 3 : 1 Algorithme génétique

1 **Initialiser** une population (***n*** individus)**2**

2 **Calculer** le degré d'adaptation de chaque individu

3 **Tant que** non fini ou non convergence

   i. **Sélectionner** les $c \times n$ meilleurs individus, les apparier et effectuer un **croisement** pour obtenir les nouveaux individus

   ii. Chacun des ***l*** caractères des nouveaux individus **mute** avec une probabilité ***m***

4 Ne conserver que les ***n*** meilleurs individus

5 **Calculer** le degré d'adaptation de chaque individu

6 **Fin Tant que**

4. Résultats de simulation de l'algorithme génétique pour les problèmes étudiés (20 machines /10 jobs)

	Cmax optimale	La séquence	Cmax AG	La séquence
Exemple 1	539	[6, 2, 9, 4, 1, 7, 3, 8, 5, 10]	**562**	[6, 1, 9, 8, 1, 7, 3, 2, 10, 5]
Exemple 2	549	[9, 10, 4, 7, 3, 5, 8, 1, 2, 6]	**593**	[10, 1, 2, 7, 4, 5, 3, 8, 16, 9]
Exemple 3	529	[6, 8, 2, 5, 1, 10, 7, 4, 9, 3]	**564**	[10, 2, 6, 5, 3, 7, 8, 4, 9, 1]
Exemple 4	528	[5, 7, 4, 9, 1, 8, 10, 6, 2, 3]	**566**	[5, 10, 1, 9, 8, 4, 6, 7, 3, 2]
Exemple 5	541	[9, 2, 3, 1, 4, 5, 7, 8, 10, 6]	**581**	[2, 5, 3, 1, 7, 8,10, 9, 4, 6]

Tableau 3 : 1 Résultats de simulation avec l'algorithme génétique

5. Adaptation et application de l'algorithme des chameaux

5.1. Le principe de l'algorithme

Le fonctionnement de l'algorithme des chameaux repose sur le principe de recherche locale de l'eau et la nourriture ou des oasis les plus proches dans le désert. Les chameaux se déplacent sous forme de caravanes réparties de manière aléatoire. Dans cet algorithme, nous appuyons sur plusieurs facteurs qui influencent la recherche des chameaux. Parmi ces facteurs, il y a la température, l'endurance et l'approvisionnement de oasis.

5.2. Le fonctionnement de l'algorithme

- Crée les caravanes à partir des chameaux
- Déclaration des paramétré $T\ initiale$, $E\ initiale, S\ initiale$ et la température minimale et maximale
- Généré les caravane initiale aléatoire
- Calculer la fonction d'évaluation
- Calculer les paramétré $actuel$, $E\ actuel$, $S\ actuel$ en fonction de $Cmax$
- Calculer la nouvelle fonction d'évaluation

5.3. Adaptation de l'algorithme des chameaux

Dans notre système nous supposons que les jobs sont présentés les chameaux et les caravanes présenter la séquence des chameaux dans la caravane et $Cmax$ présenter la durée entre la caravane et la source de oasis ou nourriture.

Les paramètres de l'algorithme sont considérés comme suit :

- Une caravane représente un ensemble de procédures de recherche qui évoluent en parallèle ;
- Le nombre de chameaux : le nombre de chameaux est relatif au système étudié, dans notre adaptation le nombre de chameaux représente le nombre de recherches parallèles effectuées sur une zone de l'espace de recherche ;
- Nombre des caravanes : 100 caravanes ;
- Les pas du jour =100 pas sachant qu'un jour représente le nombre d'itérations de l'algorithme. Être proche à la nourriture pour un chameau peut être équivalent à être proche à une solution meilleure d'une itération à une autre ;
- Les pas des tous les jours =1000 pas
- La température maximal ($T_{\max}$) = 75℃
- La température minimal (T_{min}) = 25℃
- L'endurance initial (E) = 01
- L'approvisionnement initial (S) = 0, en effet l'approvisionnement indique à quel point un chameau est loin ou proche d'une solution meilleure. En effet, plus l'endurance est petite plus on est proche d'une solution meilleure.

Les formules utilisées pour calculer la température, l'endurance et l'approvisionnement sont données comme suit :

$$T_j\ actuel = (T_{max} - T_{min}) * (\frac{la\ fonction\ fitness}{Les\ pas\ des\ tous\ les\ jours}) + T_{min}$$

$$E_{j\ actuel} = E_{j\ précedent} * (1 - \frac{T_{actuel}}{T_{max}}) * (1 - \frac{Les\ pas\ de\ jour}{Les\ pas\ des\ tous\ les\ jours})$$

$$S_{j\ actuel} = S_{j\ précédent} * \left(1 - (\frac{la\ fonction\ fitness}{Les\ pas\ des\ tous\ les\ jours}) * (\frac{Les\ pas\ de\ jour}{Les\ pas\ des\ tous\ les\ jours})\right)$$

Algorithme 3 : 2 Algorithme des chameaux

Créer une première caravane de chameaux (multi-solutions).

Calculer la forme physique individuelle de la caravane de chameaux et trouvez le meilleur actuel.

Tant que (compteur <nombre total d'étapes du trajet)

Pour i = 1 : Caravane de chameaux

Calculer $Tactuel(j)$, $Sactuel(j)$, $Eactuel(j)$ dans la Caravane.

Mettre à jour les emplacements des chameaux

Si (une condition d'oasis se produit)

Mettre à jour l'endurance $E\ actuel$ et l'approvisionnement $S\ actuel$

Fin si

Classer les chameaux dans le Caravane et trouver la meilleure solution

Fin pour i

Fin tant que

Garder la meilleure solution

6. Résultat de simulation par l'algorithme des chameaux pour les problèmes étudier (20 machines /10 jobs)

	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
Cmax optimale	539	549	529	528	541
La séquence	[6, 2, 9, 4, 1, 7, 3, 8, 5, 10]	[9, 10, 4, 7, 3, 5, 8, 1, 2, 6]	[6, 8, 2, 5, 1, 10, 7, 4, 9, 3]	[5, 7, 4, 9, 1, 8, 10, 6, 2, 3]	[9, 2, 3, 1, 4, 5, 7, 8, 10, 6]
AC(Cmax)	**556**	**569**	**560**	**566**	**575**
La séquence	[9,6,7, 1,4,2,8, 3,5,10]	[9,10,4, 7, 3, 8,5, 1, 2, 6]	[6,8,3,2, 5,9,1,7,4, 10]	[4,9,5,2 ,10,8,1,6,7 ,3]	[9,1,2,8,5, 7,4,6,3 ,10]
E_moy	0.266	0.258	0.261	0.260	0.255
S_moy	0.044	0.043	0.043	0.043	0.042
T_moy	52.800	53.444	53.800	53.300	53.750

Tableau 3 : 2 Résultats de simulation avec l'algorithme des chameaux

7. Interprétation de résultat de simulation

Après simulation, nous constatons que l'algorithme des chameaux, après son application sur notre problème, ne fournit pas une solution optimale, mais il s'en approche considérablement.

De plus, chaque facteur de l'algorithme nous fournit des informations. Plus nous nous rapprochons de la solution optimale, plus les facteurs de l'endurance (E) et l'approvisionnement (S) augmentent. En revanche, le facteur de température (T) nous fournit une autre information : plus ce facteur est élevé, plus le nombre d'itération dans

l'algorithme est important et les chameaux se déplacent sur une grande distance pour atteindre la solution optimale.

8. Hybridation de l'algorithme des chameaux et la recherche dispersée

Dans ce travail, nous avons développé un algorithme hybride entre l'algorithme des chameaux et l'algorithme de recherche dispersée afin d'améliorer la dispersion des caravanes des chameaux dans le désert.

L'algorithme développé considère trois sous-ensembles :

Dans la première étape, nous avons créé deux ensembles : un ensemble de référence appelé "ensemble des solutions de référence 1", cet ensemble contient les 10% des meilleures solutions, et un ensemble de solutions aléatoires composé de 90% individus restants.

Dans la deuxième étape, l'algorithme génère toutes les solutions possibles. Ensuite, les solutions optimisées sont placées dans un ensemble de référence, tandis que les autres solutions potentielles sont placées dans un ensemble de solutions aléatoires.

Dans la troisième étape, nous avons créé un deuxième ensemble de référence appelé "ensemble des solutions de référence 2". Cet ensemble contient les 15% des solutions qui sont proches des solutions optimales trouvées.

8.1. Les paramètres de l'algorithme hybride

- Nombre des chameaux : par rapport le système étudier
- Nombre des caravanes : 100 caravanes
- Les pas de jour =100 pas
- Les pas des tous les jours =1000 pas
- La température maximal ($T_{\max}$) $=$ 75℃
- La température minimal (T_{min}) $=$ 25℃
- L'endurance initial (E) = 01
- L'approvisionnement initial (S) = 0
- Ensemble des solutions de référence 1 : les 10% des meilleures solutions (les solutions optimale)
- Ensemble des solutions de référence 2 : les 15% des solutions proche à la solution optimale
- Ensemble des solutions aléatoires : les 75 % des solutions restante

Algorithme 3 : 3 Algorithme des chameaux

Créer une première caravane de chameaux (multi-solutions).

Créer une population de solutions

Créer un ensemble de solutions de référence 1

Créer un ensemble de solutions de référence 2

Créer un ensemble de solutions aléatoires

Calculer la forme physique individuelle de la caravane de chameaux

Tant que (compteur <nombre total d'étapes du trajet)

Pour i = 1 : Caravane de chameaux

Calculer $Tactuel(j)$, $Sactuel(j)$, $Eactuel(j)$ dans la Caravane.

Mettre à jour les emplacements des chameaux

Si (une condition d'oasis se produit)

Mettre à jour l'endurance $E\ actuel$ et l'approvisionnement $S\ actuel$

Fin si

Classer les chameaux dans le Caravane et trouver la meilleure solution

Si (Len (ensemble des solutions)) =nombre des caravanes :

Classer les solutions en ordre croissent

Ajouter les solutions dans ensemble des solutions

Pour (R1 =10) :

Ajouter les 10 premières solutions dans ensemble _ référence_1

Pour (10 <R2<25) :

Ajouter les solutions du 11 au 25 d'ensemble_ solutions dans ensemble _référance_2

Pour (R3>25) :

Ajouter les solutions du 26 au 100 d'ensemble_ solutions dans l'ensemble_ solut_aléatoire

Fin pour R3

Fin pour R2

Fin pour R1

Fin Si

Fin pour i

Fin tant que

Garder la meilleure solution

9. Comparaison des résultats de simulation entre l'algorithme génétique et l'algorithme des chameaux et l'algorithme hybride

Dans cette partie, nous présenterons les résultats des simulations pour l'algorithme des chameaux, l'algorithme génétique et l'algorithme hybride combinant l'algorithme des chameaux et l'algorithme de recherche dispersée. Nous comparerons également les résultats de ces trois algorithmes et les commenterons.

	La solution optimale	AG(Cmax)	AC(Cmax)	AC_ hybride avec la recherche dispersée un ensemble de référence (Cmax)	AC_ hybride avec la recherche dispersée deux ensembles des références (Cmax)
Exemple N°1	539	562	556	553	579
Exemple N°2	549	593	569	575	575
Exemple N°3	529	564	560	569	584
Exemple N°4	528	566	566	573	608
Exemple N°5	541	581	575	581	609

Tableau 3 : 3 Résultat de simulation pour l'algorithme des chameaux, l'algorithme génétique et l'algorithme hybride pour un problème flow shop 20 machines 10 job

	Algorithme génétique				Algorithme des chameaux				Algorithme des chameaux hybride avec la recherche dispersée avec un ensemble de référence				Algorithme des chameaux hybride avec la recherche dispersée avec deux ensembles des références			
	10 jobs 20 machines	20 jobs 20 machines	20 jobs 30machine	50 jobs 20 machines	10 jobs 20 machines	20 jobs 20 machines	20 jobs 30 machines	50jobs 20 machines	10 jobs 20 machines	20 jobs 20 machines	20 jobs 30 machines	50 jobs 20 machines	10 jobs 20 machines	20 jobs 20 machines	20 jobs 30 machines	50 jobs 20 machines
Example 1	563	808	1047	1426	556	791	1031	1410	553	787	1034	1424	609	788	1057	1420
Example 2	589	825	1054	1458	569	821	1045	1457	575	815	1052	1441	608	816	1052	1452
Example 3	554	792	1024	1454	560	809	1018	1417	563	811	1003	1428	584	812	1025	1459
Example 4	592	795	998	1422	566	796	1022	1420	563	809	1031	1402	575	788	1012	1437
Example 5	575	812	1050	1422	575	825	1062	1411	563	790	1076	1398	597	795	1052	1419

Tableau 3 : 4 Comparaison des résultats de l'algorithme génétique et de l'algorithme des chameaux et l'algorithme hybride avec un ensemble de référence et avec deux ensembles des références

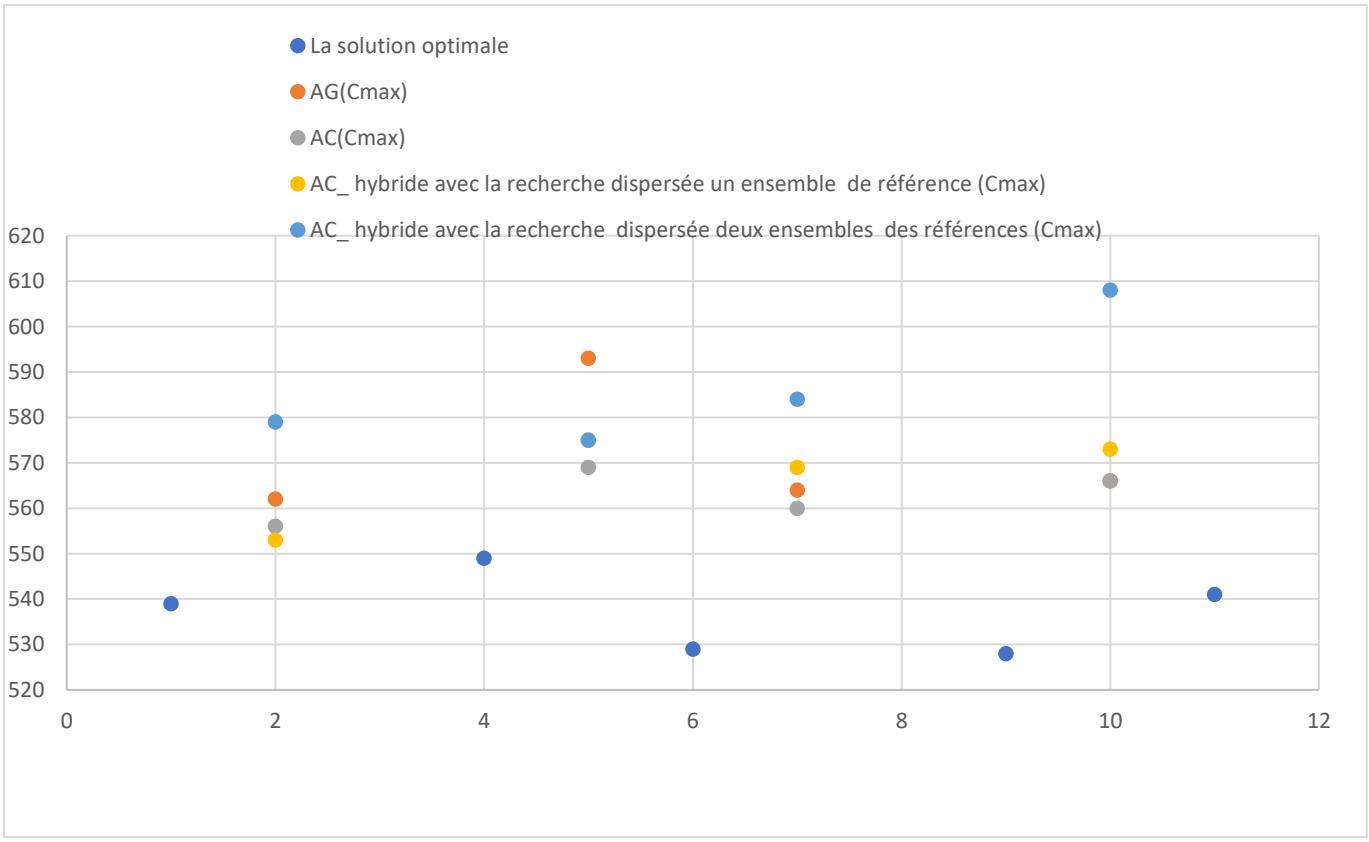

figure 3. 2:Représentation de la dispérsion des solutions

Après les simulations et l'analyse des résultats présentés dans les tableaux 3 et 4, nous constatons que l'algorithme des chameaux nous a donné les meilleures solutions, qui sont proches de la solution optimale.

En revanche, nous remarquons que l'algorithme génétique nous a fourni de bonnes solutions, mais celles-ci sont loin des solutions optimales.

En ce qui concerne l'algorithme hybride à une ensemble de référence, il nous a donné des solutions proches de celles fournies par l'algorithme génétique.

Quant aux solutions fournies par l'algorithme hybride à deux ensembles de référence, elles sont très éloignées des solutions optimales.

10.Conclusion

Dans ce chapitre, nous avons abordé l'application des certains algorithmes pour résoudre le problème de l'ordonnancement de la production. Nous avons ajusté l'algorithme génétique et l'algorithme des chameaux pour notre problème spécifique.

Après les simulations de ces deux algorithmes, nous avons analysé les résultats obtenus. Finalement, nous avons développé un algorithme hybride entre l'algorithme des chameaux et l'algorithme de recherche dispersée.

À partir de cet algorithme hybride, nous avons créé deux variantes : une utilisant un seul ensemble de référence et l'autre utilisant deux ensembles de référence. Enfin, nous avons analysé les résultats. À partir de ces résultats, il est apparu que l'algorithme des chameaux nous a donné de bons résultats par rapport à l'algorithme génétique. De même, l'algorithme hybride entre l'algorithme des chameaux et l'algorithme de recherche dispersée avec une seule ensemble de référence a donné de bons résultats. Cependant, l'algorithme hybride avec deux ensembles de référence a donné des résultats éloignés des solutions optimales.

De tout cela, nous concluons que l'algorithme des chameaux peut être appliqué aux problèmes d'ordonnancement de la production dans une atelier flow shop, et pour améliorer les solutions, nous pouvons hybride cet algorithme avec de recherche dispersée.

Conclusion générale

Conclusion générale

Les problèmes d'ordonnancement de la production sont considérés comme des problèmes d'optimisation en recherche opérationnelle. L'un des domaines spécialisés dans ce domaine est le développement des algorithmes pour résoudre les problèmes d'ordonnancement. Ces problèmes sont classés comme des problèmes NP-difficiles. Pour résoudre ces problèmes, il y a plusieurs approches sont utilisées, notamment les métaheuristiques. Les métaheuristiques sont principalement inspirées de la nature et du comportement des organismes vivants. Ces algorithmes peuvent s'adapter au problème posé et peuvent être hybrides, en combinant des méthodes pratiques et étudiées pour améliorer les solutions proposées.

Effectivement, il existe un grand nombre de métaheuristiques qui nous permettent d'obtenir des solutions optimales pour nos problèmes. Cela dépend de la complexité de ces problèmes. Parmi les problèmes couramment rencontrés dans l'ordonnancement de production, on trouve les problèmes d'atelier flow shop.

Les problèmes d'atelier flow shop sont des problèmes d'ordonnancement où une seule séquence d'opérations doit être déterminée pour l'ensemble des tâches. L'objectif est de minimiser le temps total de production, le temps d'attente ou tout autre critère défini.

Pour résoudre ces problèmes d'atelier flow shop, il y a plusieurs métaheuristiques peuvent être utilisées, telles que les algorithmes génétiques, l'algorithme des chameaux et les algorithmes de recherche tabou, etc.

Ces métaheuristiques offrent des approches flexibles et adaptatives pour trouver des solutions de qualité dans des problèmes complexes.

Il est important de choisir la métaheuristique la mieux adaptée en fonction des caractéristiques spécifiques du problème d'ordonnancement et de ses contraintes. Cela peut être déterminé par des expérimentations et des études comparatives pour identifier la métaheuristique la plus performante dans chaque cas

Nous avons abordé dans notre sujet la résolution du problème posé, qui est celui d'ordonnancement dans un atelier flow shop. Nous avons appliqué plusieurs algorithmes et comparé les résultats entre eux.

Parmi ces algorithmes, nous avons utilisé l'algorithme génétique et l'algorithme des chameaux. Finalement, nous avons développé un nouvel algorithme hybride combinant l'algorithme des chameaux et l'algorithme de la recherche dispersée.

Après avoir adapté chaque algorithme à notre problème, nous avons réalisé des simulations de ces algorithmes en utilisant le langage Python. Nous avons obtenu plusieurs résultats, que nous pouvons résumer comme suit :

- L'algorithme génétique a montré de bonnes performances en termes d'exploration de l'espace de recherche et de recherche de solutions globales. Il a réussi à trouver des solutions de qualité, mais avec un temps d'exécution relativement long.
- L'algorithme des chameaux a également donné des bons résultats, en particulier en termes d'exploitation des solutions locales et de convergence rapide vers des solutions optimales. Il a affiché des temps d'exécution plus courts que l'algorithme génétique.
- L'algorithme hybride développé en combinant l'algorithme des chameaux et l'algorithme de recherche dispersée a montré des performances améliorées par rapport aux deux autres algorithmes. Il a réussi à exploiter les avantages des deux approches et à trouver des solutions de haute qualité dans un temps raisonnable.

Après l'hybridation, nous avons créé un algorithme hybride avec une seule ensemble de référence comprenant des solutions optimisées et un ensemble de solutions aléatoires. Cet ensemble comprend des solutions éloignées des solutions optimisées.

De plus, nous avons créé un autre algorithme qui contient deux ensembles de référence et un ensemble aléatoire, Après avoir comparé les résultats, nous avons remarqué ce qui suit :

- L’algorithme hybride qui comprend un seul ensemble de référence nous donne de bonnes solutions optimisées. En revanche, le deuxième algorithme hybride, qui comprend deux ensembles de référence, nous donne des solutions qui sont quelque peu éloignées des solutions optimisées.

Cela suggère que l'utilisation de plusieurs ensembles de référence peut avoir un impact sur la qualité des solutions générées. Dans le cas de l'algorithme hybride avec une

seule ensemble de référence, il est possible que la combinaison des solutions optimisées et aléatoires dans une seule ensemble ait permis une meilleure exploration de l'espace de recherche.

En revanche, l'algorithme hybride avec deux ensembles de référence pourrait avoir introduit une certaine redondance ou confusion dans la sélection des solutions. Les deux ensembles de référence peuvent se chevaucher ou contenir des informations contradictoires, ce qui peut conduire à des solutions moins optimales.

ANNEXES

Exemple (10 jobs, 20 machines)

P (i, j) i=jobs j=machine	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	22	30	17	20	3	3	31	2	14	17	6	12	25	25	6	21	15	29	25	28
2	16	31	16	9	1	4	10	10	30	11	20	3	18	10	30	25	30	20	11	23
3	26	18	20	22	16	19	6	15	4	31	27	31	24	9	6	7	31	13	23	16
4	13	9	22	12	9	29	6	7	25	15	4	25	8	12	7	20	19	12	16	14
5	14	21	3	7	21	10	5	12	24	16	29	31	16	19	18	10	9	15	21	5
6	9	20	9	2	26	6	24	20	9	16	2	2	15	16	7	23	5	21	8	27
7	20	28	7	13	24	26	3	14	6	9	23	30	24	17	24	6	25	19	28	8
8	13	22	11	26	26	11	1	18	31	12	17	24	8	31	15	26	13	15	24	19
9	20	11	7	5	15	7	21	17	25	19	17	15	26	15	15	12	31	12	9	30
10	18	27	12	20	13	30	8	2	25	29	12	14	27	13	4	31	11	9	4	9

Tableau 1 : Tableau présenter les temps opératoire p(i,j) entre les machines j et les jobs i pour système de 20 machines et 10 jobs dans un atelier flow shop

Exemple : (20 jobs, 20 machines)

P (i, j) i=jobs j=machine	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	13	1	22	19	8	11	19	27	7	15	2	23	21	10	13	22	19	1	20	31
2	24	3	30	1	3	8	4	4	7	6	5	30	23	23	17	25	7	31	20	2
3	22	31	31	4	17	30	16	5	19	29	2	16	8	11	12	2	3	3	30	28
4	8	4	8	31	25	11	30	22	21	10	22	23	2	1	19	18	2	27	26	14
5	6	3	2	28	19	25	26	17	9	22	8	21	7	6	22	28	2	11	28	31
6	27	12	27	3	28	8	3	6	11	3	11	31	1	25	28	1	14	6	24	6
7	4	25	20	3	4	31	11	25	22	12	29	9	14	10	6	24	17	15	14	11
8	20	27	2	30	7	26	15	7	12	26	15	16	22	1	3	23	1	29	7	20
9	24	5	14	8	11	3	11	19	3	18	16	9	9	1	2	1	25	20	13	13
10	24	24	11	16	28	13	31	12	27	15	27	22	22	23	4	13	4	14	19	31
11	8	25	21	17	12	10	8	7	9	16	5	16	23	14	23	24	5	26	4	17
12	18	8	27	26	14	25	14	22	21	23	4	26	15	1	5	5	24	4	31	26
13	2	8	14	19	27	6	19	26	17	29	15	13	21	4	11	8	8	12	25	15
14	1	22	5	18	11	29	14	28	8	5	13	24	4	9	25	8	10	27	1	27
15	19	23	30	30	26	28	18	22	30	20	3	28	1	30	29	22	10	30	4	3
16	29	16	20	12	15	15	18	16	28	7	23	12	26	31	19	14	20	1	14	11
17	11	20	7	2	15	3	15	29	4	21	27	17	26	25	24	12	1	29	13	21
18	20	25	3	1	27	23	5	5	15	7	23	1	11	20	12	28	26	16	18	13
19	24	8	9	8	24	14	14	5	30	25	24	28	5	25	2	16	15	25	7	20
20	13	29	1	31	26	6	22	10	11	1	3	6	19	2	7	10	4	10	12	2

Tableau 2 : Tableau présenter les temps opératoire p(i,j) entre les machines j et les jobs i pour système de 20 machines et 20 jobs dans un atelier flow shop

P (i, j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	25	15	19	29	30	3	31	16	10	2	28	12	17	13	2	20	4	8	15	21
2	11	17	26	13	18	14	15	7	5	10	13	27	3	15	15	2	5	19	29	6
3	12	7	5	27	28	15	2	14	18	25	5	9	26	3	5	7	7	22	6	24
4	10	1	2	13	17	4	1	1	30	14	28	16	30	3	19	22	9	22	15	10
5	16	10	15	10	1	4	5	1	23	2	13	6	18	22	29	26	6	11	30	20
6	20	25	30	12	14	20	9	23	9	31	31	19	24	17	29	7	17	28	20	29
7	20	11	1	3	8	7	31	2	26	16	2	13	17	6	21	13	30	1	4	2
8	10	16	1	26	2	22	30	30	31	27	11	7	28	12	4	14	21	13	20	18
9	5	18	21	25	23	10	30	12	25	29	16	17	13	6	1	21	3	7	2	27
10	28	11	1	12	12	27	16	19	12	7	28	12	19	12	13	12	10	31	27	1
11	27	24	13	30	15	23	30	8	27	19	9	28	9	3	6	20	28	31	26	23
12	7	14	27	19	26	12	2	24	22	28	27	21	17	15	8	2	18	11	3	2
13	22	7	24	25	10	6	24	15	23	31	17	5	9	11	7	2	14	12	7	4
14	23	12	15	26	24	27	4	11	26	26	2	3	6	12	1	18	15	19	6	17
15	8	6	19	25	15	27	20	16	15	16	13	1	6	30	15	13	14	29	30	8
16	10	9	26	7	15	4	24	8	6	10	27	8	29	21	17	30	1	5	22	25
17	21	19	28	23	20	22	28	2	12	29	21	8	25	10	27	31	25	28	9	4
18	9	3	28	16	4	29	31	25	29	23	1	5	17	9	13	2	4	1	24	17
19	27	16	7	28	28	15	16	9	16	12	19	24	12	1	11	1	22	24	14	15
20	10	7	24	24	30	31	28	19	16	2	7	24	6	6	4	6	4	7	5	28
21	21	5	3	22	14	20	11	23	30	3	30	31	17	5	22	17	3	20	7	6
22	15	24	10	25	13	12	7	3	3	21	4	24	14	28	26	19	29	3	28	25
23	26	23	14	31	22	7	18	31	5	18	24	2	12	25	12	20	14	6	3	21
24	14	25	21	30	5	24	12	19	30	29	17	13	19	9	14	26	6	14	20	11
25	31	26	14	15	24	28	27	1	27	26	8	9	10	10	18	30	13	20	21	9
26	5	13	2	31	16	4	23	18	24	2	29	22	25	11	30	20	1	26	8	18
27	4	29	28	4	5	19	7	4	12	13	2	30	7	2	31	20	19	25	11	26
28	20	14	20	20	29	6	26	3	31	23	31	6	23	1	19	14	14	19	16	4
29	13	23	16	27	16	11	8	24	2	4	26	16	1	1	7	14	14	8	14	28
30	6	15	24	14	28	25	25	7	22	24	29	30	31	15	20	16	2	28	23	29
31	18	9	2	27	17	30	3	7	31	9	6	23	5	22	25	4	28	31	19	7
32	15	3	26	31	31	7	9	16	14	23	28	20	11	29	2	21	20	12	3	14
33	7	20	9	31	16	1	20	9	6	30	23	9	21	15	25	11	2	10	19	9
34	10	13	8	2	12	7	23	16	5	15	15	22	5	16	31	12	25	23	28	24
35	7	30	22	3	6	17	4	30	1	11	25	17	25	27	14	17	2	10	7	1
36	16	10	18	1	2	21	19	24	25	6	18	19	29	28	10	25	9	11	25	29
37	30	21	31	3	20	19	30	15	9	13	22	21	1	18	16	6	15	26	9	27
38	15	19	13	11	23	17	20	8	22	2	2	25	29	21	23	7	24	6	1	14
39	18	2	10	28	7	23	30	21	28	10	11	3	9	22	20	12	25	22	20	11
40	6	2	27	24	5	10	7	21	23	30	24	1	24	25	7	2	11	22	16	2
41	9	12	1	23	5	23	4	28	18	4	22	31	15	31	27	28	9	18	12	14
42	20	21	9	1	31	16	5	8	27	28	12	9	26	15	30	24	29	25	20	1
43	15	4	20	22	11	10	4	29	26	20	25	11	28	18	21	17	21	21	30	23
44	14	10	27	14	28	2	23	25	14	11	4	5	9	3	4	25	5	11	28	24
45	4	28	16	9	10	2	16	8	29	11	26	21	8	16	6	20	12	6	17	3
46	28	7	11	10	27	25	13	28	4	26	22	2	6	19	1	29	21	12	18	11
47	1	13	2	14	16	25	10	22	6	7	9	26	26	21	17	20	2	18	5	22
48	23	6	29	26	25	8	20	16	16	21	24	14	10	13	7	14	8	5	9	3
49	26	11	25	28	24	24	30	18	15	10	31	1	10	2	13	6	10	2	28	12
50	6	28	12	7	3	28	10	16	14	8	6	20	14	4	16	28	3	6	1	17

ANNEXES

P (i, j) I= jobs J=machine	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	21	27	1	22	21	4	3	12	27	5	19	27	29	25	13	24	30	21	8	17	3	18	19	29	29	9	13	29	15	5
2	10	9	24	24	5	25	24	25	28	28	22	27	7	13	8	31	30	12	14	31	7	15	16	2	31	19	27	5	11	14
3	11	21	29	7	3	22	5	5	29	29	26	11	16	20	28	6	21	22	3	16	1	22	3	13	29	18	8	26	26	16
4	30	19	16	24	22	1	9	12	4	18	31	13	21	22	22	7	13	24	17	29	24	23	3	23	11	16	12	20	23	12
5	23	2	12	8	22	26	2	7	22	2	5	21	5	16	7	6	28	18	18	20	20	30	24	1	7	10	7	31	8	25
6	19	23	19	23	13	1	23	26	31	22	18	31	29	17	23	1	30	21	29	15	2	12	16	31	12	22	25	19	15	4
7	10	13	21	24	17	16	5	29	8	8	22	16	13	24	20	8	6	23	22	16	18	15	31	18	31	14	3	5	28	18
8	1	24	20	4	13	3	27	14	17	26	3	8	11	25	29	21	12	11	23	3	15	8	24	31	13	23	12	21	21	1
9	5	21	3	13	3	13	6	3	1	1	27	10	25	28	23	15	9	3	3	20	25	12	16	15	3	25	28	14	28	3
10	12	1	3	5	7	1	3	12	29	16	26	28	13	4	1	18	27	20	27	10	15	22	13	6	26	29	29	4	18	21
11	30	31	29	12	17	30	5	2	12	10	23	14	30	29	22	20	1	4	2	16	7	11	4	13	10	7	12	17	25	5
12	29	16	25	30	23	10	12	17	1	18	18	20	22	17	30	28	17	17	27	15	16	12	27	1	25	25	9	25	9	6
13	30	21	25	15	3	31	5	7	17	11	9	18	4	12	13	16	30	20	5	31	27	11	6	1	9	26	19	7	31	19
14	17	9	1	3	28	3	8	14	1	23	7	20	12	27	21	31	10	4	11	31	25	26	31	14	27	12	30	27	7	10
15	27	2	14	2	22	18	1	5	20	14	31	1	5	1	3	10	26	20	30	5	21	30	30	26	14	21	10	5	6	19
16	9	3	18	24	13	27	2	12	30	21	6	24	4	18	14	9	25	8	28	29	21	7	19	16	27	26	5	28	22	9
17	12	2	22	20	3	25	22	11	21	5	13	23	19	17	30	6	4	25	8	13	10	22	20	29	17	27	17	19	10	28
18	22	13	19	15	10	15	28	17	31	4	2	8	2	28	8	30	18	10	11	11	2	8	4	13	4	17	2	18	9	20
19	27	1	8	8	29	19	30	6	26	21	25	24	26	31	24	26	23	2	12	18	23	4	3	3	4	12	17	29	15	10
20	23	2	10	6	19	26	18	22	10	1	31	26	3	16	21	9	23	30	22	18	30	30	7	8	14	3	28	2	18	1

Tableau 3 tableau présenter les temps opératoire p(i,j) entre les machines j. et les jobs i pour système de 30 machines et 20 jobs dans un atelier flow shop

Les Références

1. A.J, O., & C.N, P. (1997). On the complexity of coupled tasks scheduling. Discrete Applied Mathematics 72. 141-154.
2. ACHOURI , A. (2022, 2 26). Le CSP Pour le problème de gestion de temps.
3. AZEM, S. (2010, Juin 22). ORDONNANCEMENT DES SYSTEMES FLEXIBLES DE PRODUCTION SOUS CONTRAINTES DE DISPONIBILITE DES RESSOURRCES. Thèse de Doctorat, Ecole National Supérieure des Mines Saint-Étienne.
4. BAHMANI, Y. (2017, Avril 17). Optimisation multicritère de l'ordonnancement des activité de la production et de la maintenance intégrées dans un atelier Job Shop. These de Doctorat, université de Batna, Algérie.
5. Baptiste , A. (2006). Les métaheuristiques en optimisation combinatoire. paris.
6. Bastin, F. (2014, 01). Programmation dynamique.
7. BELKAID, F. (2014, Janvier 14). Investigation sur l'ordonnancement des systioèmes à machines parallèles. Thèse de Doctorat en productique , Université de Tlemcen , Algérie.
8. BENBOUZID SITAYEB, F. (2005, Juin 29). CONTRIBUTION A L'ETUDE DE LA PERFORMANCE ET DE LA ROBUSTESSE DES ORDONNANCEMENTS CONJOINTS PRODUCTION / MAINTENANCE-CAS DU FLOW SHOP. Thèse de Doctorat, université de Franche-Comté.
9. BENDAHMANE, A. (2011, 11 25). Le recuit simulé.
10. BENTTALEB, M. (2018, Septembre 24). Gestion de production sous inncertitudes. Thèse de Doctorat, Université de Technologie de Troyes.

11. BERKANI, A. (2013). Métaheuristique Hybride Réseaux de Neurones Artificiels-PSO du Recuit Simulé pour la Commande d'un Procédé Industriel Non-linéaire.

12. BOUKEF BEN OTHMAN, H. (2009, Juillet 3). Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutique Optimisation par algorithmes génétiques et essaims particulaires . Thèse de doctorat soutenue à Université De Tunis , École Nationale D'ingénieurs De Tunis.

13. BOUMEDIENE , F. (2020, Février 13). Ordonnancement des systèmes flexibles de production basé sur les méthaheuristiques hybrides. Thèse de Doctorat, Université de Tlemcen, Algérie.

14. BOUMEDIENE, F. (2020, 2 13). Ordonnancement des systèmesOrdonnancement des systèmes flexibles de production basé sur les méthaheuriqtiques hybrides.

15. Chaari, T. (2010). Un algorithme génétique pour l'ordonnancement robuste: application au problème du flow shop hybride. Université de Valenciennes et du Hainaut-Cambresis. france.

16. CHERGUI, A., & DAHMANI, A. (2017, 7 15). Ordonnancement d'un flow-shop par métaheuristique hybride.

17. CHERGUI, A., & DAHMANI, A. (2017, Juin 15). Ordonnancement d'un flox-shop par métaheuristique hybride. Mémoire de Master, Université de Tlemcen, Algérie.

18. DJERBOUAI , K. (2018). Alignement multiple des séquences protéiques par l'algorithme de recherche tabou.

19. EMBOUAZZA, K., & BETTAHAR, M. (2020, Novembre 8). Résolution d'un problème d'ordonnancement dans un atelier flow-shop avec l'algorithme du chauve-souris. Mémoire de Master en Génie Industriel, Université de Tlemcen , Algérie.

20. Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic Decision Making. The Annual Review of Psychology. Annual Review of Psychology. Récupéré sur Annu. Rev. Psychol. 2011.62:451-482. Downloaded from www.annualreviews.org

21. Gondran, A. (2014). Recherche tabou générique pour problèmes à contraintes binaires. Toulouse, France: hal-01063343v2f.

22. Grainia, S. (2015, 04). L'algorithme de Branch and Price and Cut pour le problème de conception de réseaux avec coûts fixes et sans capacité.

23. HACHIMI, H. (2013, 6 29). HYBRIDATIONS D'ALGORITHMES MÉTAHEURISTIQUES EN. Rbat. Récupéré sur https://tel.archives-ouvertes.fr/tel-00905604

24. HADRI, A. (2012). L'ORDONNANCEMENT PAR INSERTION EN TEMPS REEL DE LA PRODUCTION DANS UN ATELIER FLEXIBLE. Mémoire de Magister en Technologie Génie Industriel et Productique, université Hadj Lakhdhar Batna.

25. Hasanen , S., & Zied , O. (2017). Camel Herds Algorithm: a New Swarm Intelligent Algorithm to Solve Optimization Problems. International Journal on Perceptive and Cognitive Computing, 6-10.

26. Hanset, A., Fei, H., Roux , O., Duvivier , D., & Meskens, N. (2007). Ordonnancement des interventions chirurgicales par une recherche Tabou : Exécutions courtes vs longues.

27. Hao, J.-K., Galinier, P., & Habib, M. (1999). Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. Revue d'Intelligence Artificielle

28. HOUARI , H. (2012). Planification et Ordonnancement en temps réel d'un Job shop en utilisant l'Intelligence Artificielle. Mémoire de Magister en Automatique.

29. HOUARI, H. (2012, Juillet 02). Planification et Ordonnancement en temps réel d'un Job shop en utilisant l'Intelligence Artificielle. Mémoire de Magister en Automatique , Université de Tlemcen.

30. HOUBAD , Y. (2011, Décembre). Modélisation et Ordonnancement temps réel d'un Job shop à l'aide des métaheuristiques. Mémoire de Magister en Automatique, Université Abou Bakr Belkaid –Tlemcen .

31. JOURDAN, L. (2010, 09 15). Métaheuristiques Coopératives du déterministe au stochastique. Habilitation à Diriger les Recherches , Université Lille I.

32. Khalid Ibrahim, m., & Ramzy , S. (2016). Novel Optimization Algorithm Inspired by Camel Traveling Behavior. Iraqi Journal for Electrical and Electronic Engineering, 167-177.

33. L, M., & Pinedo. (2016). Schedling, Theorie Algorithms, and Systems. NYU Stern School of Business, New York, USA: Springer.

34. LARIBI, I. (2018, 12 12). Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables.

35. LARIBI, I. (2018, Décembre 12). Résolution de problèmes d'ordonnancementde type Flow-Shop de permutation en présenece de contraintes de ressources non renouvlables . Thèse de Doctorat , Université de Tlemcen , Algérie.

36. MAMOUNI, A., & OULD MOHAMED , B. (2018, Juin 28). Une métaheuristique basée sur le comportement des lucioles pour la résolution d'un problème d'ordonnancement dans un atelier flow shop. Mémoire de Master en Génie Industriel, Université de Tlemcen.

37. MAMOUNI, A., & OULD MOHAMED, B. (2018, 6 28). Une métaheuristique basée sur le comportement des lucioles.

38. MOUHOUB, N. (2011). Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet. THESE DE DOCTORAT EN SCIENCES , UNIVERSITE FERHAT ABBAS - SETIF.

39. nadir, r. (2019). un probléme d'ordonnacement de type job shob dans un environnement dynamique . univ de msila .

40. NOUIRI, M. (2017, 07 03). Implémentation d'une méta-heuristique embarquée. 12. (Maroua NOUIRI, Éd.) tunis.

41. Ramzy , S., Jawad , R., & Hussein, M. (2021). A New Version of Modified Camel Algorithm for Engineering Applications.

42. Ramzy, a., Falih M. , A., & Abdulkareem , S. (2019). A modified camel travelling behaviour algorithm for engineering applications. Australian Journal of Electrical and Electronics Engineering, 2-11. doi:10.1080/1448837X.2019.1640010

43. Remla, N. (2019). Un problème d'ordonnancement de type Job Shop dans un environnement dynamique. Mémoire de Master, Université Mohamed Boudiaf M'SILA.

44. ResearchGate. (2023, 3 6). Récupéré sur ResearchGate: https://www.researchgate.net/figure/Le-pseudo-code-de-la-methode-du-recuit-Simule_fig1_242254652/download

45. SAOUDI, B., & REMITA, F. (s.d.). Résolution du problème d'ordonnancement conjoint de la production et de la maintenance de type Flow Shop. Mémoire de Master en Génie Industriel , Université de Tlemcen , Algérie.

46. SOUIER, M. (2012, Novembre 27). Investigations sur la sélection de routages alternatifs en temps réel basées sur les métaheuristiques-les essaims particulaires . Thèse de Doctorat en Science en productique, Université de Tlemcen, Algérie.

47. TAMRABET, Y. (2018). Développement de l'approche intelligence artificielle et de la méthode variationnelle des amas (CVM) pour l'étude des propriétés thermodynamiques des alliages métalliques. thése de doctorat, univérsite de batna.

48. TRABELSI, W. (2012, Novembre 14). Ordonnancement des systèmes de production flexibles soumis à différents types de contraintes de blocage. Thèse de Doctorat en automatique traitement de signal et des images , université de Lorraine.

49. Utama, D., Safitri, W., & Garside, A. (2022). A Modified Camel Algorithm for Optimizing Green Vehicle Routing Problem with Time Windows. Jurnal Teknik Industri, 23-35.

50. Xavier, D. (2014). Programmation par contraintes sur les flux de données.

Résumé

Dans ce travail, nous avons abordé la résolution d'un problème d'ordonnancement dans le domaine de la production. L'ordonnancement de la production est une branche dans la recherche opérationnelle. Les problèmes d'ordonnancements sont classés parmi les problèmes NP-difficiles, et pour les résoudre, des méthodes basées sur le développement d'algorithmes sont utilisées en fonction de leur degré de difficulté. Ainsi, plusieurs méta-heuristiques ont été développées dans ce cadre. Dans notre étude, nous avons utilisé une métaheuristique hybride qui combine une nouvelle méta-heuristique appelée "Algorithmes des chameaux » avec la métaheuristique « recherche par dispersion ».

Enfin, nous avons comparé les résultats obtenus par ces méta-heuristiques suite aux simulations avec l'algorithme génétique, et il s'est avéré que l'algorithme des chameaux nous a donné d'excellents résultats et des solutions optimales. Pour améliorer les solutions de cette méta-heuristique, nous avons hybridées avec l'algorithme de recherche dispersée, ce qui nous a donné des bonnes solutions en comparaison avec celles fournies par l'algorithme génétique.

Mots-clés : algorithme des chameaux, ordonnancement, flow shop, algorithme hybride algorithme génétique, la recherche dispersée

الملخص

في هذه الأطروحة، تطرقنا إلى دراسة موضوع الترتيبات في الإنتاج. تُعتبر الترتيبات الإنتاج فرعًا من فروع البحوث العملياتية. كما تُعتبر مشكلات الترتيبات جزءًا من إدارة الإنتاج التي تهدف إلى تطوير أنظمة الإنتاج. تُصنف مشكلات الترتيبات ضمن المشكلات ذات الحدود الصعبة الكثيرة. ولحل هذه المشكلات، نحتاج إلى طرق محددة واستخدام خوارزميات وفقًا لصعوبتها. لذلك، تم تطوير عدة فوقيات استدلال لحل هذه المشكلات. في دراستنا، قمنا بتطبيق فوقيات استدلال جديدة تُسمى "خوارزميات الجمال".

وفي النهاية، قمنا بمقارنة النتائج التي أظهرتها الخوارزميات بعد المحاكاة باستخدام الخوارزمية الجينية، وتبين لنا أن خوارزمية الجمال أعطتنا نتائج جيدة جدًا. ولتحسين حلول هذه الخوارزمية، قمنا بدمجها مع خوارزمية البحث المشتت وأعطتنا حلولًا جيدة مقارنةً بالحلول التي أعطتها لنا الخوارزمية الجينية.

الكلمات المفتاحية: خوارزمية الجمال، ترتيبات الإنتاج، وحيدة المسار، الخوارزمية المهجنة، الخوارزمية الوراثة، خوارزمية البحث المتشتت

Abstract

In this thesis, we addressed the study of scheduling in the field of production. Production scheduling is considered a branch of operations research. Furthermore, scheduling issues are part of production management, which aims to develop production systems. Scheduling problems are classified as NP-hard problems, and to solve these problems, we need specific methods and algorithms based on their difficulty level. As a result, several meta-heuristics have been developed to tackle these issues. In our study, we applied new meta-heuristics called "Camel Algorithms". Finally, we compared the results obtained by these meta-heuristics through simulations with the genetic algorithm, and it was evident that the Camel Algorithm provided excellent results. To enhance the solutions of this meta-heuristic, we hybridized them with the scatter search algorithm, which yielded good solutions compared to those provided by the genetic algorithm.

Keywords: camel Algorithm, scheduling, flow shop, hybrid algorithm, genetic algorithm., scatter search