REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Informatique Décisionnel et Optimisation

Par : MEGUIRECHE SOUMIA

SUJET

Ordonnancement des systèmes de production flexible

Soutenu publiquement le : / /2019 devant le jury composé de :

Nom et prénom Enseignant		
……	**Université de M'sila**	**Président**
Dr. BOUNIF Mohamed Elhad	**Université de M'sila**	**Rapporteur**
……	**Université de M'sila**	**Examinateur**

Promotion : 2018 /2019

Remerciements

Je tiens tout d'abord à remercier Dieu le tout-puissant et Le Miséricordieux, qui m'a donné la vie et la force ainsi que le courage et l'audace pour dépasser toutes les difficultés, d'avoir suivis et accomplis toutes les étapes de mes études.

En second lieux je tiens à remercier mon encadreur ***Dr. BOUNIF Mohamed El Hadi*** *pour m'avoir assisté et m'aides par leurs précieux conseils et de m'avoir offerts toutes les possibilités telle que suivent et prestation de documentations nécessaires pour la réalisation de mon mémoire de fin d'étude.*

Et je remercier aussi tous les professeurs qui aide moi de les deux années de master au Mohamed BOUDIAF M'sila.

Dédicace

Toutes les lettres ne sauraient trouver les mots qu'il faut… Tous les mots ne sauraient exprimer la gratitude, l'amour, Le respect, la reconnaissance… Aussi, c'est tout simplement que

Je dédie ce modeste travail à :

A MA TRÈS CHÈRE MÈRE : *Belouadeh Malika*

Autant des phrases aussi expressives soient-elles ne sauraient montrer le degré d'amour et d'affection que j'éprouve pour toi. Tu m'as comblé avec ta tendresse et affection tout au long de mon parcours. Tu n'as cessé de me soutenir et de m'encourager durant toutes les années de mes études. En ce jour mémorable, pour moi ainsi que pour toi, reçoit ce travail en signe de ma vive reconnaissance et mon profond estime. Puisse le tout puissant te donner santé, bonheur et longue vie afin que je puisse te combler à mon tour.

A MON TRÈS CHER PÈRE : Nour eddine

Autant de phrases et d'expressions aussi éloquentes sont-elles ne sauraient exprimer ma gratitude et ma reconnaissance. Tu as su m'inculquer le sens de la responsabilité, de l'optimisme et de la confiance en soi face aux difficultés de la vie. Tes conseils ont toujours guidé mes pas vers la réussite. Ta patience sans fin, ta compréhension et ton encouragement sont pour moi le soutien indispensable que tu as toujours su m'apporter. Je te dois ce que je suis aujourd'hui et ce que je serai demain et je ferai toujours de mon mieux pour rester ta fierté et ne jamais te décevoir. que Dieu le tout puissant te préserve, t'accorde santé, bonheur, quiétude Del esprit et te protège de tout mal.

- ***A mes beaux-frères et ma sœur :*** *dans tous mes moments d'examens par leur soutien moral et ses belles surprises sucrées. Je vous souhaite un avenir plein de joie, de bonheur, de réussite et de sérénité. Je t'exprime à travers ce travail mes sentiments de fraternité et d'amour.*

- ***A mon cher Youcef*** *qui m'a toujours à mes côtés. Veuillez trouver dans ce travail l'expression de mon respect le plus profond et mon affectation la plus sincère, que Dieu vous garde et vous garde ainsi que vos soins.*

- ***A tous mes proches la famille Meguireche et Belouadeh*** *avec tous mes sentiments de respect et d'amour.*

- ***A*** *mon encadreur* ***Dr. BOUNIF Mohamed El hadi -*** *qui m'ont aidés pour réaliser ce travail, dont, je leur doit un grand remercîment , grand respect et une parfaite appréciation et grande considération*

Table des matières

Chapitre 2 : L'ordonnancement des systèmes de production flexible

Liste des Figures

Liste des Tables

Liste des algorithmes

INTRODUCTION GENERALE

De manière générale dans le domaine de la production où la rivalité est grande, pour cela les unités de production laissant place à des nouvelles structures industrielles. Connu sous le nom de Système Flexible de Production qui permet d'unifier la gestion de la production, de la qualité et des matériaux. Ce mémoire concerne l'évaluation des performances et l'ordonnancement dans ces systèmes. L'ordonnancement de la production est défini comme un ensemble des jobs (taches) à réaliser sur un ensemble des ressources, de sorte qu'une fonction objective soit optimisée.

Pour augmenter la flexibilité des systèmes de production, il est possible de multiplier le nombre des machines qui peuvent réaliser une même opération. Ces machines sont identiques et sont regroupés on étage. Les modèles résultant sont appelé Flow Shop Hybride.

Notre travail s'articule autour de cette problématique dans le but de minimiser le temps totale d'exécution pour terminer toutes les tâches (makespan) et minimiser la somme des retards on utilise le problème d'ordonnancement de type flow shop hybride pour diminuer les coûts de production et améliorer la qualité des produits.

Le mémoire est organisé en quatre chapitres.

Le premier chapitre de ce mémoire propose, une vue d'ensemble sur les concepts d'une chaine logistique et de sa gestion avec les différents niveaux décisionnels respectivement stratégique, tactique et opérationnel. Et dernière section, on abordera de définir d'un côté les besoins méthodologiques spécifiques à la gestion des chaînes logistiques.

Le deuxième chapitre nous introduisons les concepts, les caractéristiques générales des problèmes d'ordonnancement, ainsi que les différents types d'ateliers. Ensuite, nous présentons la flexibilité dans les systèmes de production.

Le chapitre trois nous présentons les problèmes d'optimisation en général. Ensuite nous présentons en particulier les problèmes d'optimisation mono et multi objectifs. Ainsi, nous exposons aussi une présentation des différentes méthodes de résolution d'un problème d'optimisation combinatoire développées dans la littérature, classées en méthodes exactes et en méthodes approchées.

Le quatrième chapitre présente notre implémentation d'un problème d'ordonnancement de type flow shop hybride dans laquelle on va utiliser l'algorithme génétique et particulièrement l'algorithme de NSGA II avec quelques exemples.

A la fin du ce travail nous finirons par une conclusion générale.

CHAPITRE 1

LES CHAINES LOGISTIQUES

Introduction

La chaine logistique est une fonction importante au cœur des activités industrielles, elle utilisé pour fournir des produits et des services depuis les matières premières jusqu'aux clients finaux.

Dans ce premier chapitre, nous définissons en premier abord la notion de la chaine logistique. Ensuite nous intéressons aux spécificités les flux, les fonctions. Par suit nous nous définissons la gestion et le rôle d'ordonnancement de la chaine, et dernière section, on abordera de définir d'un côté les besoins méthodologiques spécifiques à la gestion des chaînes logistiques.

1. La logistique

L'origine du mot vient de l'ancienne langue grecque et vient du mot "logos" qui signifie "proportion, compte, raison, discours".

Dans leur ouvrage, « La logistique au service de l'entreprise », Colin et al. [01] ont proposé la définition suivante : « *La logistique est le processus stratégique par lequel l'entreprise organise et soutient son activité. À ce titre, on peut déterminer et gérer les flux matériels et informationnels afférents, tant internes qu'externes, en amont qu'en aval.* »

2. La chaîne logistique

Le terme « chaîne logistique » vient de l'anglais *Supply Chain* qui signifie littéralement « chaîne d'approvisionnement ».

Christopher [02] définit la chaîne logistique comme étant «*le réseau d'entreprises qui participent, en amont et en aval, aux différents processus et activités qui créent de la valeur sous forme de produits et de services apportés au consommateur final. En d'autres termes, une chaîne logistique est composée de plusieurs entreprises, en amont (fourniture de matières et composants) et en aval (distribution), et du client final* ».

2.1 Les flux de la chaîne logistique

Le bon fonctionnement d'une entreprise repose sur la circulation efficace de certains flux. On peut les classer en trois catégories : les flux d'information, les flux physique et les flux financier. Comme le montre la figure (1.1).

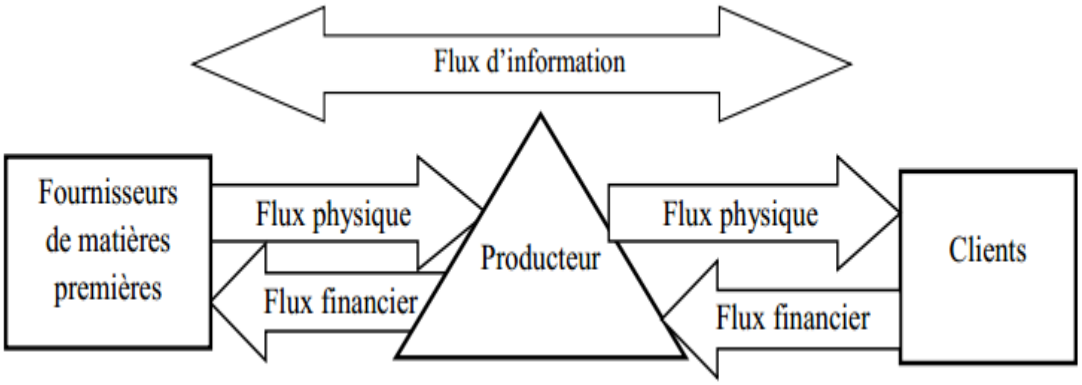


Figure 1.1 Modélisation des flux d'une chaine logistique [02].

2.1.1 Le flux d'information

Lorsque le client passe une commande, il déclenche un flux d'information. Ce flux est composé d'un flux de donnée et de décision qui sont essentiels pour un bon fonctionnement d'une chaîne logistique.

2.1.2 Le flux physique

Appelés également flux produit, le flux physique est constitué par le déplacement et le stockage des marchandises transportées depuis les matières premières jusqu'aux produits finis.

Ces flux constituent le cœur d'une chaîne logistique, sans lesquels les autres flux n'existeraient pas. Ils peuvent être regroupés en trois étapes : produire, stocker et transporter.

2.1.3 Le flux financier

Le flux financier concerne toute la gestion pécuniaire des entreprises : ventes des produits, achats de composants ou des matières premières, mais aussi des outils de production, de la location d'entrepôts,...et bien sûr du salaire des employés. Le flux financier est généralement géré de façon centralisée dans l'entreprise dans le service financier ou comptabilité, en liaison toutefois avec la fonction production par les services achats et le service commercial [03].

2.2 Les fonctions de la chaîne logistique

Ganeshan et Hrisson [4] définissent une chaîne logistique comme étant: « *le réseau des moyens de production et de distribution qui assurent les tâches d'approvisionnement en matières premières, la transformation de ces matières premières en produits semi finis et en produits finis, et la distribution de ces produits finis aux clients* ». De la définition ci-dessus on distingue, que les fonctions d'une chaîne logistique vont de la gestion des commandes, l'approvisionnement, la production, le stockage et la vente**.**

2.2.1 La gestion des commandes

Cette activité concerne la navigation de commande du client par l'attribution d'inventaire grâce à l'allocation des stocks, la cueillette, l'emballage, l'expédition, et le cycle de réapprovisionnement. L'objectif fondamental de performance est l'expédition des produits en basée sur délai de livraison, les quantités commandé, et des spécifications sur la qualité. [05].

2.2.2 L'approvisionnement

Pour l'activité d'approvisionnement, Ding dit que [06], le décideur a besoin d'identifier les fournisseurs potentiels à choisir pour alimenter les différentes usines en matières premières, en composants et en produits semi-finis.

2.2.3 La production

La fonction de production est au cœur de la chaîne logistique, il s'agit là des compétences que détient l'entreprise pour fabriquer, développer ou transformer les matières premières en produits ou services. Elle donne la capacité à la chaîne logistique pour produire et donne ainsi un indice sur sa réactivité aux demandes fluctuantes du marché **[0**7**].** La production représente l'ensemble des activités nécessaires pour réaliser le produit, le fabriquer et le stocker. Il se base essentiellement sur la conception du produit et la gestion de la production et des services.

2.2.4 Le stockage

Le stockage inclut toutes les quantités stockées tout au long du processus en commençant par le stock des matières premières, le stock des composants, le stock des en-cours et finalement le stock des produits finis. Les stocks sont donc partagés entre les différents acteurs : les fournisseurs, les producteurs et les distributeurs. Ici aussi se pose la question de

l'équilibre à trouver entre une meilleure réactivité et la réduction des coûts. La gestion des stocks est l'une des clés de la réussite et l'optimisation de toute une chaîne logistique [08].

2.2.5 La vente

Une vente est l'opération par laquelle un bien ou un droit détenu par un vendeur est cédé à un acheteur en échange d'une contrepartie, généralement la remise d'une somme d'argent.

3. La gestion de la chaîne logistique

La gestion de la chaîne logistique est une stratégie qui vise à la fois la réduction des frais globaux, permettant une position plus concurrentielle à toutes les différentes parties de la chaîne logistique, et l'optimisation de la satisfaction du client final par une plus grande adaptabilité des systèmes de production et de distribution [09].

3.1 Les niveaux décisionnels du La gestion de la chaine logistique

Pour n'importe quel type de chaîne logistique, les niveaux décisionnels peuvent être regroupés en trois niveaux: stratégique, tactique et opérationnelle.

3.1.1 Décisions stratégiques

Les décisions stratégiques sont des orientations générales configurent la chaîne sur le long terme. Ces décisions couvrant:

- La définition de mission de l'entreprise.
- La définition des stratégies concurrentielles.

3.1.2 Décisions tactiques

Les décisions tactiques s'intéressent aux décisions à moyen terme, qui devront être mis en application pour déployer la stratégie décidée par l'entreprise. Ces décisions couvrent:

- La planification agrégée de la production.
- La gestion de projet.

3.1.3 Décisions opérationnelles

Les décisions opérationnelles sont prises pour un horizon à court terme. Ce niveau, elles génèrent un plan détaillé de production ou d'ordonnancement.

3.2 Les mesures de la performance de la chaîne logistique

La gestion de la chaîne logistique cherche à améliorer le système global de production. Pour atteindre cet objectif, il faut connaître sa performance effective et déterminer une cible ou un objectif à atteindre. La première étape de suivi les performances consiste donc à « mesurer la performance ». Ensuite, il faut prendre des décisions de réingénierie et agir sur le système à travers des variables de décision afin de tendre vers la cible choisie [36]. Nous devons donc définir un ensemble d'indicateurs pour mesurer cette performance.

Chopra et Meindil [37] identifient six indicateurs de performance :

- Les infrastructures : ce sont les sites où l'on produit, assemble et stocke les composants. La performance de la chaîne logistique est influée par leur rôle, localisation, flexibilité et capacité.
- Les stocks : la réactivité d'une chaîne logistique peut être influée par tout changement de politique de stockage.
- Les transports : les transports, représentant une lourde charge pour la chaîne logistique, influent sur la performance, la réactivité et l'efficacité de celle-ci.
- L'information : est l'indicateur le plus important car elle influe directement sur l'ensemble des autres indicateurs et sur l'efficacité de la chaîne.
- Le Sourcing : définit le travail que chacun doit effectuer à travers la chaîne. c'est aussi la répartition des activités de l'entreprise mère.
- Les prix: donner une valeur aux biens et services (produits) en fonction de plusieurs paramètres car leurs répercussions influencent le comportement (décisions) des clients et les performances de l'entreprise.

3.3 Le rôle d'ordonnancement en gestion de la chaine logistique

L'ordonnancement occupe une place particulière et joue un rôle privilégie dans la gestion informatisée des flux de la chaine logistique, s'inscrivant dans des niveaux de décision à la fois tactique et opérationnel.

En pratique, l'occurrence des évènements aléatoires endogènes et exogènes est fréquente et fortement probable. En effet, la mise à jour de l'ordonnancement devient nécessaire, par conséquent les problèmes d'ordonnancement sont traites aux niveaux inférieurs. L'ordonnancement traduit l'ensemble des décisions de fabrication définies par le programme

directeur de production en instructions d'exécution détaillées destinées au lancement au contrôle et au pilotage à court terme de l'activité des postes de travail. A l'issu de la fonction ordonnancement, on obtient un calendrier qui assure une affectation optimale des taches sur les ressources disponibles, en précisant la durée et la date d'exécution de chacune d'elles, tout en respectant certaines contraintes et en optimisant un ou plusieurs fonctions objectifs [10].

4. Besoins méthodologiques pour les chaînes logistiques

Dans cette section particulièrement, nous exposons deux grands axes de recherche en termes de méthodologies et techniques utilisées au service de la gestion des chaînes logistiques, à savoir l'optimisation et l'aide à la décision.

4.1 Optimisation

L'optimalité de la qualité de service en logistique sous-entend une satisfaction des utilisateurs du système proposé par l'intégration des fonctionnalités nécessaires à la bonne gestion de la chaîne logistique. Dans ce sens, L'optimisation a été introduite dans un souci d'amélioration des services fournis peu importe le domaine auquel ils s'appliquent [11].

L'optimisation de la chaîne logistique consiste en la recherche des méthodes spécifiques permettant un optimum, en maximisant ou minimisant un objectif, dite fonction objectif.

4.2 Aide à la décision

Les techniques d'aide à la décision ont pour but de modéliser de manière la plus fidèle possible les préférences d'un expert. Cette modélisation va permettre alors de concevoir et de construire des outils adaptés et capables d'assister ou de remplacer un décideur sur des problématiques complexes. L'aide à la décision formalise l'expertise obtenue après un entretien avec le décideur et les interactions entre le décideur et son environnement. Étant donné que nous nous intéressons principalement à l'expertise des décideurs dans le domaine de la logistique, le modèle utilisé se doit d'être souple et suffisamment élaboré afin de permettre la représentation des différents comportements décisionnels les plus communément rencontrés en gestion de chaînes logistiques de gestion de crise [11].

Conclusion

Dans ce chapitre, on a vu une brève introduction sur les concepts d'une chaine logistique et de sa gestion. Nous avons décrit les niveaux de décisions et les besoins méthodologiques de la chaine logistique.

Dans le chapitre suivant nous allons apprendre l'ordonnancement des systèmes de production flexible.

CHAPITRE 2

L'ORDONNANCEMENT DES SYSTEMES DE PRODUCTION FLEXIBLE

Introduction

L'ordonnancement est une fonction essentielle en gestion de la production. Les problèmes d'ordonnancement apparaissant dont tous les domaines de l'organisation, la planification, la gestion… etc.

Dans ce chapitre, nous présentons quelques notions générales d'ordonnancement. Ainsi, nous définissons la notion « flexibilité ». Nous terminons ce chapitre par la description de l'atelier de production de type Flow Shop Hybride.

1. Les problèmes d'ordonnancement

Un problème d'ordonnancement peut être considéré comme un sous problème de planification dans lequel il consiste à organiser dans le temps la réalisation des tâches, et ainsi d'établir leur planning d'exécution et leur allouer des ressources visant à satisfaire un ou plusieurs objectifs sous une ou plusieurs contraintes. En se basant sur les concepts de tâche, ressource, contrainte et objectif, l'ordonnancement peut également être déterminé comme.

1.1 Les taches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début et une date de fin d'exécution et qui consomme des ressources avec des quantités déterminées. Un coût est attribué à une tâche pour estimer sa priorité. On distingue deux types des tâches :
- les tâches morcelables (préemptives) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution des certains problèmes.
- les tâches non morcelables (indivisibles) qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

On note en général $N= \{J_1, J_2,\ldots, J_n\}$ l'ensemble des tâches, chaque tâche est caractérisée par [12] :

- La durée opératoire de la tâche i sur la machine j : (p_{ij}).
- La date de disponibilité de la tâche i : (r_i).
- La date de début d'exécution de la tâche i :(s_i).
- La date de fin d'exécution de la tâche i :(c_i).
- La date d'achèvement souhaitée de la tâche i :(d_i).
- Le facteur de priorité ou poids de la tâche i : (w_i).
- Le retard algébrique de la tâche i : ($L_i = c_i - d_i$).
- Le retard vrai de la tâche i : (T_i = max ($c_i - d_i$, 0)).
- L'indicateur de retard de la tâche i : (U_i = 1, si T_i>0, U_i= 0, sinon.

1.2 Les ressources

L'exécution des différentes tâches nécessite la mise en œuvre d'un ensemble des moyens techniques et d'opérateurs humains. La capacité d'une ressource est en réalité limitée. On distingue deux types de ressources [13]: les ressources renouvelables et celles non renouvelables.

- Les ressources renouvelables : une ressource est dite renouvelable si, après avoir été allouée à une tâche, elle redevient disponible à la fin de l'exécution de cette dernière. Les ressources renouvelables usuelles sont les machines, les processeurs, les fichiers, le personnel,... etc.
- Les ressources non renouvelables (consommables) : ces ressources ne sont plus disponibles après la fin de la tâche. Elles sont en fait épuisées par la réalisation de cette opération. C'est le cas pour l'argent, la matière première, ... etc.

1.3 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décision. Leur prise en compte permet d'avoir un ordonnancement faisable et réalisable.

Parmi ces contraintes on distingue [13]: les contraintes potentielles ou contraintes de précédence, les contraintes de concurrence, les contraintes intérieures et les contraintes extérieures.

- Contraintes potentielles

Elles lient les tâches entre elles, c'est-à-dire que telle opération ne peut être exécutée avant telle autre. Elles peuvent être temporelles (quand il faut attendre la fin d'une tâche avant de lancer la suivante). Ce sont des contraintes qui vont définir les gammes de production.

- Contraintes de concurrence

Ce type de contraintes est lié aux ressources utilisées par les tâches. Ces contraintes apparaîtront lorsque deux tâches, utilisant la même machine, ne pourront pas s'exécuter simultanément.

- Contraintes intérieures

Ce sont des conditions directement liées au système de production et à ses performances. Nous pouvons citer:
(a) les capacités des machines et des moyens de transport.
(b) les dates de disponibilités de la matière première des produits et des moyens de transport.

- Contraintes extérieures

Ce sont des conditions imposées extérieurement. Elles sont indépendantes du système de production, telles que:
(a) les dates de besoin de produits, imposées généralement par les commandes du client.
(b) les niveaux de priorité et d'urgence de quelques commandes et de quelques clients.
(c) les retards accordés pour certains produits.

1.4 Les critères d'ordonnancement

Il s'agit ici d'optimiser (maximiser ou minimiser) une fonction d'évaluation en respectant un certain nombre de contraintes. Les trois critères d'un ordonnancement sont : le coût, la qualité et le délai.

On distingue deux types de critères : les critères liés au temps et ceux liés aux coûts.

- liés au temps : le temps totale d'exécution ou le temps moyen d'achèvement d'un ensemble de taches ou encore les retards par a pour aux dates limite fixés.

- Liés aux couts : ce sont des critères qui représentent un coût d'utilisation des machines et ceux qui représentent un coût lié à l'attente des opérations avant et/ou après leur traitement.

2. Caractéristiques générales d'ordonnancement

2.1 Ordonnancement statique et dynamique

- Si l'ensemble des informations à la résolution d'un ordonnancement est fixé au départ, alors le problème d'ordonnancement est dit « statique ».
- Si une ou plusieurs informations à la résolution d'un ordonnancement n'est pas fixé au départ, alors le problème est « dynamique ».

2.2 Ordonnancement préemptif et non- préemptif

- Si les tâches d'un ordonnancement sont exécutées par morceaux, alors le problème est «préemptif».
- Si les tâches d'un ordonnancement sont exécutées sans interruption, alors le problème est «non- préemptif».

2.3 Ordonnancement admissible

Un ordonnancement est dite «admissible» s'il respect toute les contraintes du problème (date limite, précédence, limitations des ressource… etc.).

2.4 Ordonnancement actif et semi-actif.

- un ordonnancement «semi-actif» est un ordonnancement ou en peut faire un glissement à gauche locale sans toucher au contrainte.
- un ordonnancement «actif» est un ordonnancement ou aucun glissement à faire.

2.5 Ordonnancement sans retard

Dans un ordonnancement «sans-retard» on doit exécuter la tâche qui est en attente immédiatement à condition que la ressource doit disponible.

2.6 Ordonnancement multi périodes

L'ordonnancement consiste à déterminé l'affectation des différentes opérations sur les ressources et leurs dates d'exécution.

3. Classification de problème d'ordonnancement

3.1 La complexité

C'est la mesure abstraite permettent d'évaluer les algorithmes, dans la complexité d'un problème d'optimisation combinatoire en trouve :

- Un problème est dite facile si l'en résout par un algorithme efficace. Un algorithme est dite efficace est un algorithme qui permet de résoudre un problème en un temps d'exécution qui ne sont pas trop important.
- Un problème est dite difficile si en ne connut pas un algorithme efficace pour un problème.

La théorie de la complexité permet de classer les problèmes en deux classes *P* et *NP*.

3.1.1 La complexité d'un algorithme

L'objectif de la théorie de complexité est d'analyser les couts de résolutions surtout en termes de temps de calcul. Elle vise aussi à classer les problèmes en plusieurs niveaux de difficulté. Une étude a prouvé que les problèmes d'ordonnancement sont des problèmes difficiles.
En général, la complexité algorithmique se mesure par rapport à deux paramètres :

- Le temps alloué pour l'exécution de l'algorithme : il est relatif au nombre d'instructions à exécuter ainsi qu'à la taille des données manipulées.
- Espace mémoire requis : associé à la taille d'instance d'un problème donné.

3.1.2 La complexité problématique

- La classe NP :

C'est la classe des problèmes qui peuvent être décidés sur une machine non déterministe en temps polynomial.

Elle est décomposée en 3 parties :

- La classe P :

C'est l'ensemble de tous les problèmes qui ont des algorithmes de résolution de complexité polynomiale.

- La classe NP-Complet :

Les problèmes les plus difficiles de NP définissent la classe des problèmes NP-complets. Ainsi, les problèmes NP-complets sont des problèmes combinatoires dans le sens où leur résolution implique l'examen d'un nombre exponentiel de cas. Cette classe est basée sur la notion de réduction polynomiale.

- La classe NP-Difficile :

Est un problème vers lequel on peut ramener tout problème de la classe NP par une réduction polynomiale. [14]

3.2 Les notations

Nous suivons les schémas de classification proposée par Graham et al, 1979.
Classification à trois champs **α, β, γ**:

- α : d'écrit la structure de problème et ce décompose en deux sous-champ α1 et α2.
  - α1 **:** Indique le type de problème**.**
  - α2 **:** Désigne le nombre de machine.
- β : Décrit les types de contrainte prise en compte.
  - β1 = pmtn si la préemption des tâches est autorisée, sinon β1 est absent.
  - S'il y a des contraintes de précédence entre les tâches β**2** {prec, chain, tree}, sinon β**2** est vide.
  - β3 =rjsi les dates de début au plus tôt rj (ou date de disponibilité) des tâches ne sont pas forcément identiques, sinon (j, rj=0) β3 est absent.
  - β4 = j si la tâche ou le job possède une date de fin nécessaire.
- **γ :** Indique la fonction objective et le (ou les) critère(s) à optimiser.

4. La flexibilité dans les systèmes de production

4.1 Les systèmes de production

Un système de production est un lieu géométrique dans lequel évoluent quatre populations distinctes : les produits, les moyens de production, les personnes de production et le flux d'information.

4.2 Définition de la flexibilité

La flexibilité d'un objet ou d'un système, est une propriété qui recouvre généralement deux aspects complémentaires mais distincts: Un aspect interne lié à une capacité de changement et de déformation à une variété d'états possibles et un aspect externe lié à une capacité d'adaptation à des modifications de l'environnement. [15]

4.3 Dimensions de flexibilité

La classification différencie la flexibilité dans le système de production en sept dimensions [16] :

- Flexibilité process

Capacité de faire varier les tâches nécessaires à la réalisation d'un travail. Ceci permet de réaliser plusieurs travaux différents dans le système, en utilisant une variété des machines.

- Flexibilité sur le routage

Capacité de changer la séquence de passage des pièces sur les machines et de continuer à produire un ensemble donné des pièces même lorsqu'une machine tombe en panne. (Existe seulement s'il y a plusieurs séquences de production possible ou lorsque chaque opération peut être réalisée sur plus d'une machine).

- Flexibilité sur les opérations

Capacité de changer l'ordre des certaines opérations dans une gamme de production.

- Flexibilité machine

Capacité de changer d'outils et d'assembler ou monter les fixations nécessaires sans interférer avec la production et sans temps de reconfiguration long. C'est la facilité du système à faire les changements nécessaires sur les machines pour produire un ensemble donné des pièces.

- Flexibilité produit

Capacité de mettre en œuvre, rapidement et de manière économique, les changements nécessaires à l'intégration des nouvelles pièces dans le plan actuel.

- Flexibilité sur le volume

Capacité de faire fonctionner un atelier à des niveaux de production différents tout en restant à un niveau de profit acceptable.

- Flexibilité sur la production

Capacité de faire varier rapidement et de manière économique la gamme des pièces qu'un atelier peut produire. Cette flexibilité ne peut être atteinte que lorsque les autres dimensions sont satisfaites.

- Flexibilité sur l'expansion

Capacité de construire un système ou de l'agrandir facilement et de manière séparable.

5. L'ordonnancement d'atelier

Les problèmes d'ordonnancement ont des ressources qui sont des machines, ne peuvent réalisées qu'une tache à la fois.

Chaque travaille concerne une entité physique indivisible appelé produit où lorsque plusieurs produit identique sans regroupés.

Une même travaille ne peut être exécuté qu'arisant d'une opération à la fois. Pour chaque travaille il y'a un ordre stricte entre les opérations qui le composant.

Une classification simple basée sur l'organisation de l'atelier classifie les problèmes d'ordonnancement en trois classes (le problème à une machine, les problèmes à machines parallèles et les problèmes d'atelier multi-machines)

5.1 Les problèmes d'atelier à machine unique

C'est le cas le plus simple dans les problèmes d'ordonnancement où on ne dispose qu'une seule ressource pour réaliser un ensemble des travaux. Ces derniers sont constitués d'une seule tâche pour chacun d'eux. Dans ce type de problème la résolution consiste à trouver l'ordre optimal d'exécution de ces tâches.

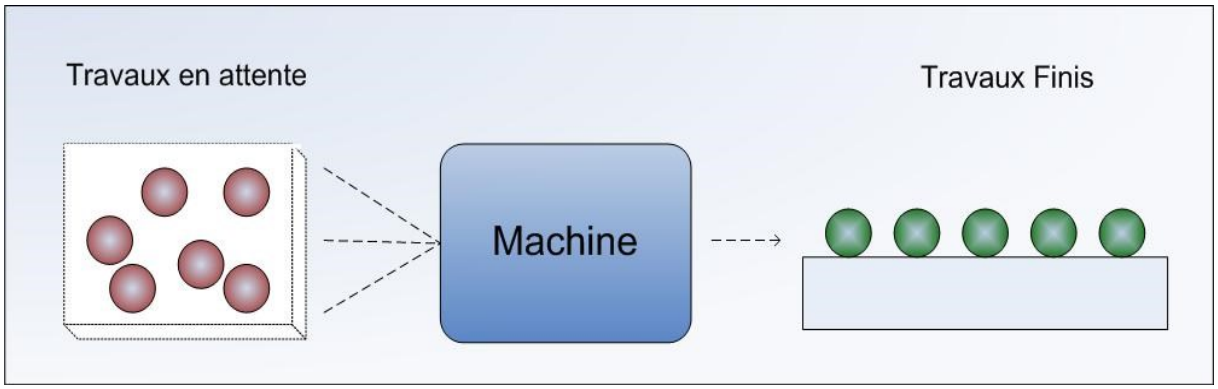


Figure 2.1 la représentation d'une machine unique [17].

5.2 Les problèmes d'atelier à machines parallèles

Les problèmes d'ordonnancement à machines parallèles proposent plusieurs machines pour l'exécution d'un travail. Elles représentent un cas particulier de problèmes multi-machines où les machines sont disposées en parallèle. On peut distinguer trois types des problèmes à machines parallèles:

- Les problèmes à machines identiques : les durées opératoires sont égales et ne dépendent donc pas des machines.
- Les problèmes à machines uniformes : la durée d'une opération varie uniformément en fonction de la performance de la machine choisie.
- Les problèmes à machines indépendantes (non liées) : les durées opératoires dépendent complètement des machines utilisées.

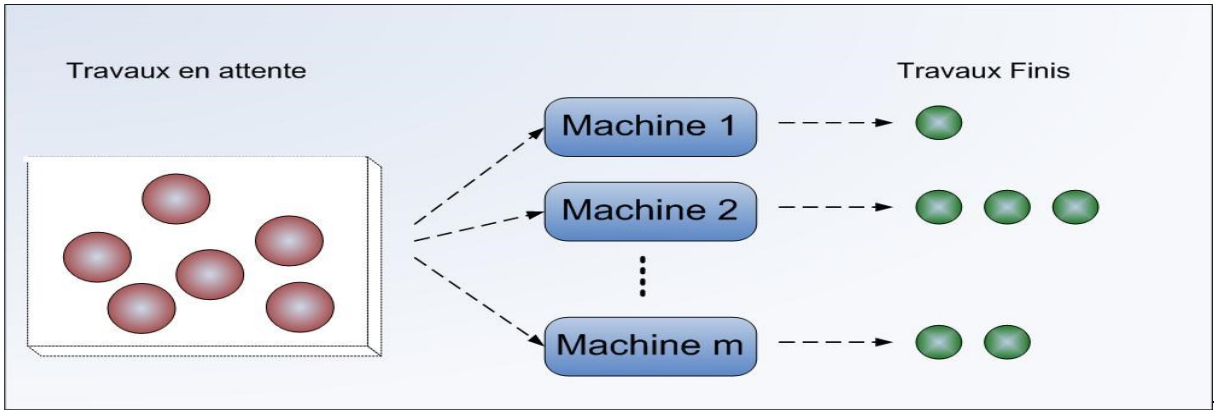


Figure 2.2 La représentation des machines parallèles [17].

5.3 Les problèmes d'atelier multi-machines

- **Atelier a cheminement unique (Flow Shop)**

Les produits sont procédés dans une direction unique selon la même gamme de fabrication. Un travail est constitué d'opération visitant les différentes machines et on chinée de manière linéaire suivant une chaine, tout travail visite chaque machine de l'atelier et l'ordre de passage d'un travail sur les différents machine et le même pour tous les travails. Les machines consacrées à la réalisation de ces produits sont inévitablement disposées en série (Figure 2.3).

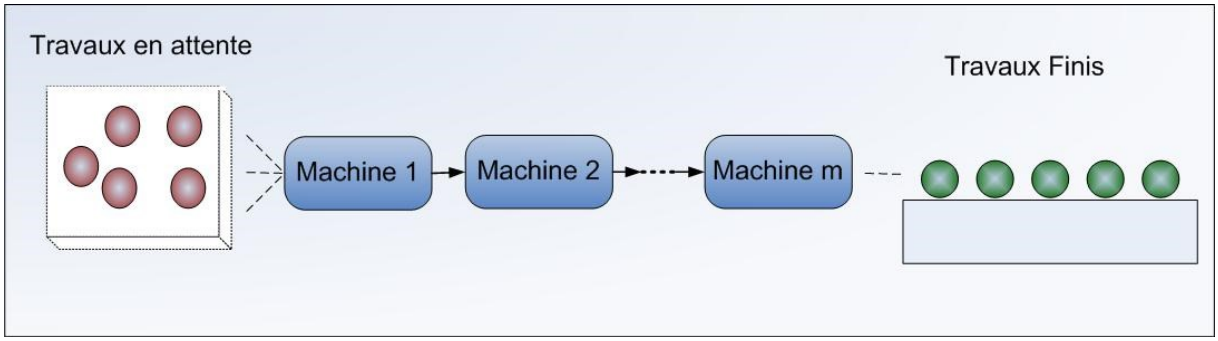


Figure 2.3 La représentation d'ateliers à cheminement unique (flow-shop) [17].

- Flow shop hybride

Le Flow Shop hybride c'est une généralisation du Flow Shop classique au cas où il est possible de multiplier le nombre des machines qui peuvent réaliser une même opération. Ces machines sont regroupées sur un ou plusieurs étages pour exécuter les différentes tâches du Flow Shop.

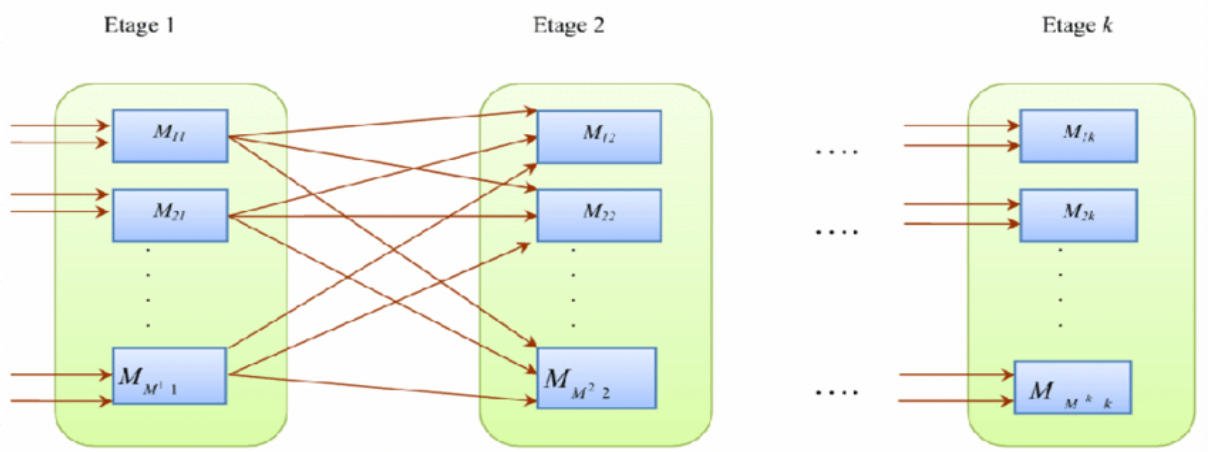


Figure 2.4 La représentation d'un flow show hybride à « k » étages.

- **Atelier à cheminement multiple (Job shop)**

Le problème de Job Shop consiste à réaliser un ensemble de *n* Job (travail) sur un ensemble de *m* machines en cherchant d'atteindre certain objectifs. Chaque travail passe sur des machines dans un ordre fixé, mais cette ordre d'exécution peut être diffèrent pour chaque travail. Les travaux ne s'exécutent pas sur toutes les machines.

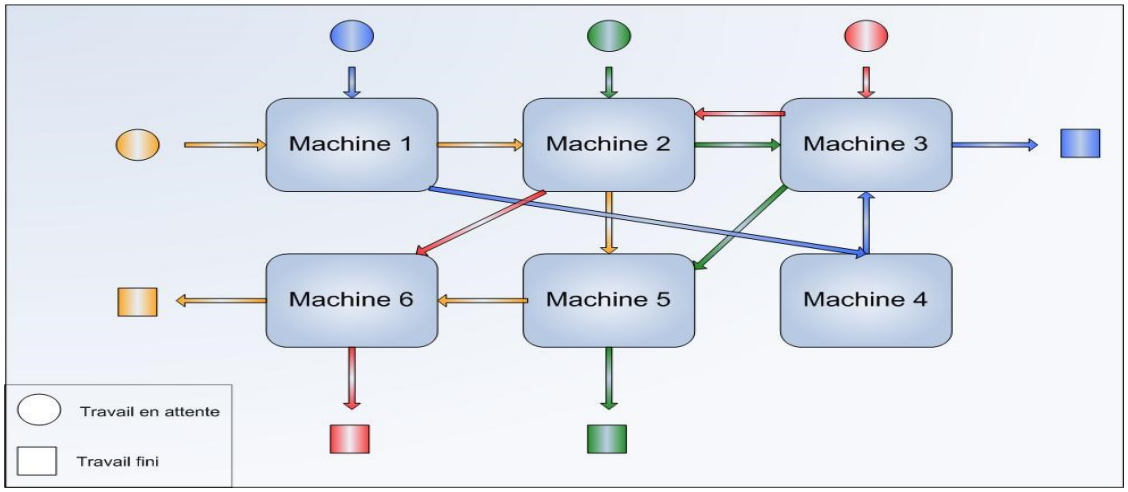


Figure 2.5 La représentation d'ateliers à cheminement multiple (job-shop) [17].

- Job shop hybride

Dans ce modèle, les machines qui effectué la même opération sont regroupées dans un même étage de façon a augmenté la productivité d'atelier.

C'est un problème FJSP, contenant deux sous-problèmes :

- Le problème de Job shop classique.
- Le problème de machine parallèle.

- **Atelier a cheminement Libres (Open shop)**

Dans le problème de type open shop l'ordre d'exécution des tâches d'un travail est totalement libre, c'est-à-dire aucune contrainte est remarquée sur la précédence entre tâches. Les produits passent alors dans n'importe quelle direction (les gammes sont libres).

6. Le problème d'ordonnancement de type Flow-shop

6.1 Etat de l'art

Le problème d'ordonnancement du flow shop hybride est l'un des problèmes les plus difficiles et également les plus courants dans le domaine industriel. Gupta, en 1988, a ouvert le champ d'études des problèmes d'ordonnancement du flow shop hybride et prouvé que le cas le plus simple de ce problème est NP-complet (le cas où il y a une seule machine dans le premier étage et deux machines dans le deuxième étage). Depuis, de nombreuses recherches ont été menées et les méthodes et techniques utilisées s'étendent des heuristiques aux méta-heuristiques (comme le recuit simulé, la recherche tabou, les algorithmes génétiques, etc.) Et aux méthodes hybrides, en passant par des méthodes exactes, avec comme objectif la minimisation du makespan ou le temps de séjour total (ou moyen) des travaux ou du nombre des travaux en retard, etc. [18]

6.2 Définition formelle de problème d'ordonnancement Flow-Shop

6.2.1 Flow-Shop classique

Dans le problème d'ordonnancement de type Flow-Shop, un ensemble de N jobs $J = \{J_1, J_2, \ldots, J_n\}$, doit être traités sur un ensemble de M machines $M = \{M_1, M_2, \ldots, M_m\}$.

Tous les jobs passent par toutes les machines sur le même ordre, ils sont donc composés de M opérations $O_i = \{O_{i1}, O_{i2}, \ldots, O_{im}\}$ .L'opération O_{ik} a besoin d'un temps d'exécution P_{ik} sur la machine $M_k \in M$ .Chaque machine ne peut effectuer qu'une seule opération à la fois et chaque job ne peut avoir qu'une seule opération en cours de réalisation simultanément. La capacité de stockage inter-machines est définie et la préemption d'opérations n'est pas autorisée. [19]

6.2.2 Flow-Shop hybride

Le Flow-Shop hybride est donc une généralisation du Flow-Shop classique au cas où plusieurs machines sont disponibles sur un ou plusieurs étages pour exécuter les différentes tâches du Flow-Shop.

Dans un problème d'ordonnancement de type Flow-Shop hybride, un ensemble de N jobs, $J = \{J_1, J_2, \ldots, J_n\}$, doit être traité dans un atelier de production composé de K étages, $E = \{E_1, E_2, \ldots, E_k\}$. Chaque étage E_k contient M_k machines parallèles identiques, avec (k = 1, 2, ..., K).

Tous les jobs exigent le même ordre des opérations, Oi = {Oi1, Oi2, . . ., Oik}, qui doivent être exécutées selon le même processus de fabrication. L'opération Oik a besoin d'un temps d'exécution Pik sur l'étage Ek ∈ E. Une machine ne peut appartenir qu'à un seul étage et ne peut effectuer qu'une seule opération à la fois. Chaque job ne peut avoir qu'une seule opération en cours de réalisation simultanément et doit être traité par une seule machine de l'étage Ek, sans interruption. [19]

6.3 Les modèles mathématiques

Les modèles mathématiques visent à trouver des équations mathématiques pour décrire les données, les contraintes ainsi que les critères utilisés pour l'optimisation des solutions.

- **Makespan** (C_{max})

Le makespan représente le temps de fin d'exécution du dernier job dans une séquence. Il est l'un des critères les plus utilisés pour évaluer le coût d'un ordonnancement. En minimisant ce critère, on peut améliorer le rendement et réduire le temps moyen d'inactivité des machines.

- **Le problème Flow Shop**

Les données du problème et le modèle qui s'expriment de la manière suivante

- n : le nombre de tâches.
- m : le nombre de Machine.
- j : l'indice de la tâches, où j =1….., n.
- k : l'indice de la machine, où k =1….., m.
- r : la position de la tâche dans une machine, où r =1….., nk.
- r_{ij} : date de début.
- d_{jk} : date de fin la tâche j.
- S_{ij} : temps de préparation de la tâche j effectuées immédiatement après la tâche i sur une machine.
- P_{jk} : temps opératoire de la tâche j.
- C_{jk} : date de fin d'exécution de la tâche j.
- T_{jk} : retard réel de la tâche j.
- C_{max} : makespan.

Minimiser (C_{max} , ∑T) (1)

Sous contraintes :

- $\sum_{j=1}^{n} X_{jkr}$ k=1,2,…., m r=1,2,…, n (2)
- $\sum_{k=1}^{m} \sum_{r=1}^{nk} X$jkr j=1,2,….., n (3)
- $R_{ki} = R_{kj}$ k=1,2,…., m i=1,2,…,m-1 j=1,2,….., m (4)
- $P_{[kr]} = \sum_{j=1}^{n} X_{jkr} P_{jk}\, r_{ij}$ k=1,2,…., r=1.2,….. n (5)
- $S_{[kr]} = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{jkr}\, y_{ij}\, S_{ij}$ k=1,2,….., r=1,2,…. n (6)
- $r_{[kr]} = \sum_{j=1}^{n} X_{jkr}\, r_{jk}$ k=1,2,….., r=1,2,…. n (7)
- $d_{[kr]} = \sum_{j=1}^{n} X_{jkr}\, d_{jk}$ k=1,2,……..,m r = 1,2,……., n (8)
- $C_{[kr]} = \max(C_{[k,r-1]} + s_{[kr]}, r_{[kr]}) + P_{[kr]}$ k=1,2,.,m r = 1,2… n (9)
- $T_{[kr]} = \max(C_{[kr]} + d_{[kr]}, 0)$ k=1,2,.,m r = 1,2… n (10)
- $C_{max} = max_{k=1}^{m}\ max_{r=1}^{m}\ C_{[kr]}$ k=1, 2,., m r = 1,2… n (11)
- $\sum T = \sum_{k=1}^{m} \sum_{r=1}^{m} T_{[kr]}$ k=1, 2,., m r = 1,2… n (12)
- X_{jkr} =0 or 1 k=1, 2,., m r = 1,2… n_k j=1,2,…, n (13)
- y_{ij} =0 or 1 i=1,2,…, n j=1,2,….., n (14)

La fonction (1) exprime directement les objectifs de notre problème, i.e., la minimisation du makespan et la minimisation de la somme des retards.

Les contraintes (2) et (3) sont des contraintes d'unicité des tâches sur la machine *k* et à la position *r*. Elles assurent qu'il y a seulement une tâche sur la machine *k* et à la position *r*, et que chaque tâche est affectée seulement une fois sur ces machines.

La contrainte (4) exprime que la position des taches dans tous les machines soit même.

La contrainte (5) calcule la durée d'opération de la tâche qui est en position *r* sur la machine *k*.

La contrainte (6) définit le temps de préparation de la tâche qui est en position *r* sur la machine *k*.

Les contraintes (7) - (10) concernent respectivement la date de début au plus tôt, la date de fin au plus tard, la date de fin d'exécution et le retard réel de la tâche qui est en position r sur la machine k.

Les contraintes (11) et (12) représentent le calcul du makespan et de la somme des retards.

La variable binaire Y_{ij} contrôle la position relative de deux tâches i et j. Si la tâche i précède

immédiatement la tâche j, alors Y_{ij} est égale à 1, sinon elle est égale à 0. Par contre, si j est la premier tâche (j=1), alors Yij est égale à 1, car aucune tâche ne précède j. [20]

Conclusion

Dans ce chapitre, nous avons définit l'ordonnancement, qui est généralement décrit comme une fonction particulière de décision au sein d'un système de gestion du travail concernant la production et La majorité des problèmes d'ordonnancement sont NP-difficile.

Dans le chapitre suivant nous nous exposons en générale Les méthodes de résolution d'un problème d'optimisation**.**

CHAPITRE 3

LES METHODES DE RESOLUTION D'UN PROBLEME D'OPTIMISATION

Introduction

Les problèmes d'ordonnancement font partie des problèmes d'optimisation combinatoire, les méthodes de résolution sont celles utilisé dans l'optimisation combinatoire.

Dans ce chapitre nous présentons les principes de base de l'optimisation combinatoire, et nous donnerons aussi une présentation des différents méthodes de résolution, ces approches de résolution peuvent être classées en deux catégories : les méthodes exactes et les méthodes approchées.

1. Problème d'optimisation combinatoire

1.1 Définition

Le problème d'optimisation combinatoire c'est un problème qui consiste à calculer une solution optimisée parmi un ensemble des solutions réalisables. Dans un problème d'optimisation combinatoire en cherche un objectif (Max, Min…) cette objectif peut-être mono objectif ou multi objectif.

1.2 Principaux concepts en optimisation

Un problème d'optimisation est défini par :

- Fonction objective

Une fonction objective est une fonction qu'on cherche à optimiser. Elle est notée $F(x)$ de manière générale $F(x)$ est un vecteur : $F(x) = [f_1(x), f_2(x),..., f_k(x)]$.

- Paramètre

Un paramètre du problème d'optimisation, est une variable qui exprime une donnée sur une dimension du problème: coût, temps,...etc. Ces paramètres correspondent aux variables de la fonction de fitness. Ils sont adaptés pendant le processus d'optimisation, pour parvenir à des solutions optimales. On les appelle aussi variables d'optimisation, variables de conception ou décision.

- Variable de décision

Les variables de décision sont des quantités numériques pour les quelles des valeurs sont à choisir. Cet ensemble de n variables est appelé vecteur de décision : ($x_1, x_2, \ldots, x_n$). Les différentes valeurs possibles prises par les variables de décision x_i constituent l'ensemble des solutions potentielles.

- Contraintes

Dans la plupart des problèmes d'optimisation, une contrainte du problème est une condition que doivent respecter les vecteurs de décision du problème. Ces contraintes C_j exprimées sous forme d'équations ou d'inéquations linéaires composées des variables de décision. Comme se suit : $C_j (x_1, x_2, \ldots, x_n) \geq 0$.

1.3 L'optimisation mono objective

Lorsqu'un seul objectif (critère) est donné, le problème d'optimisation est mono-objectif. Dans ce cas la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal, maximal).

1.4 L'optimisation multi objective

On dit qu'un problème est multi objectif s'il conduite plusieurs objectifs. Ces objectifs sont souvent conflictuels (contradictions, concurrents), qualificatif et incontournables.

L'optimisation multi objectif fournie aux décideurs un ensemble des solutions optimales dit Front de Pareto.

Mathématiquement, un problème multi objectif peut s'écrire sous la forme (3.1), où p représente le nombre d'objectifs et X un ensemble combinatoire.

$$\text{Min} \quad z(x) = (z1(x), \ldots, zp(x)) / x \in X \quad (3.1)$$

- La dominance

une solution A domine une solution B pour un problème de minimisation si et seulement si:

$\forall i \in \{1,2,\ldots,n\} : f_i (A) \leq f_i (B)$

$\exists j \in \{1,2,\ldots,n\} : f_i (A) < f_i (B)$.

On dit que B est dominée par A ou entre les deux solutions, A est la solution non dominée**.** [20]

- Pareto optimum

On appelle front de Pareto la représentation des solutions non dominées dans l'espace d'objectif. La figure 3.1 montre un exemple de front de Pareto pour un problème de minimisation à deux objectifs. L'ensemble de points blancs représentent le front de Pareto. [20]

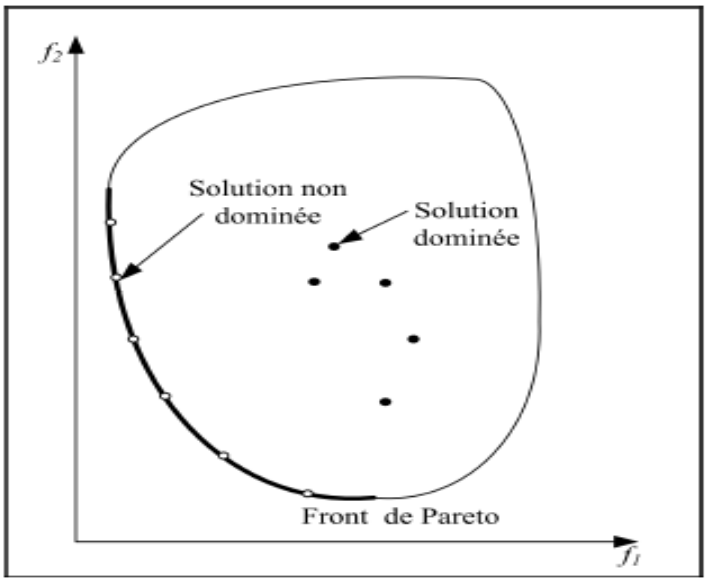


Figure 3.1 Front de Pareto **[21]**

2. Les méthodes de résolution de problème combinatoire

La principale difficulté à laquelle est confronté un décideur, en présence d'un problème d'optimisation est celui du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable.
Dans la littérature, on distingue essentiellement deux classes de méthodes :

Les méthodes exactes et Les méthodes approchées.

2.1 Les méthodes exactes

Ces méthodes sont généralement utilisées pour résoudre des problèmes de petite taille. Dans ce cas, les méthodes exactes garantissent l'optimalité des solutions trouvées si no elle dite approchée.

On distingue quatre sous-classes de méthodes exactes:

- La procédure de séparation et d'évaluation (Branch and Bound)

L'algorithme Branch and Bound consiste à placer progressivement les tâches sur les ressources en explorant un arbre de recherche décrivant toutes les combinaisons possibles. Il s'agit de trouver la meilleure configuration donnée de manière à élaguer les branches de l'arbre qui conduisent à de mauvaises solutions.

Pour appliquer la méthode de branch-and-bound, nous devons être en possession [22] :
1. d'un moyen de calcul d'une borne inférieure d'une une solution partielle
2. d'une stratégie de subdiviser l'espace de recherche pour créer des espace de recherche de plus en plus petits.
3. d'un moyen de calcul d'une borne supérieure pour au moins une solution.

- La programmation dynamique

La Programmation Dynamique est une procédure qui contient une succession d'opérations ou d'instructions en se basant principalement sur la planification et l'ordonnancement permettant de résoudre des problèmes d'optimisation.

L'idée de base de la programmation dynamique est de décomposer un problème en étapes (sous-problèmes) ou chaque étape possède un certain nombre d'états correspondant aux différentes possibilités rencontrées.

A chaque étape, on doit prendre une décision qui permet de projeter l'état courant au début de l'étape suivante. Puis, la décision du résultat optimal obtenu est associée à l'étape suivante et ainsi de suite de telle sorte que tous le problème soit traité. Dans le cas d'un problème d'optimisation, contrairement à la programmation linéaire, il n'y a pas une modélisation mathématique standard. C'est une approche de résolution séquentielle où les équations doivent être désignées selon le problème à résoudre [23].

- La programmation par contrainte

(PPC, ou CP pour constraint programming en anglais) est un paradigme de programmation apparu dans les années 1970 et 1980 permettant de résoudre des problèmes combinatoires de grandes tailles tels que les problèmes de planification et d'ordonnancement. En programmation par contraintes, on sépare la partie modélisation à l'aide de problèmes de

satisfaction de contraintes (ou CSP pour Constraint Satisfaction Problem), de la partie résolution dont la particularité réside dans l'utilisation active des contraintes du problème pour réduire la taille de l'espace des solutions à parcourir (on parle de propagation de contraintes) [24].

- La programmation linéaire

On appelle Programmation Linéaire, le problème mathématique qui consiste à optimiser (maximiser ou minimiser) une fonction linéaire de plusieurs variables qui sont reliées par des relations linéaires appelées contraintes.

2.2 Les méthodes approchées

La résolution d'un problème d'optimisation combinatoire où les problèmes sont NP-difficiles, de grande taille, se heurte à des tailles mémoire et des temps de calcul trop importants.

L'objectif n'est plus alors d'obtenir systématiquement l'optimum mais plutôt d'obtenir une solution proche de l'optimum ou de « bonne qualité » en un temps raisonnable et polynomial.

On distingue deux types de méthodes: les heuristiques et les méta-heuristiques.

2.2.1 Les heuristiques

Les heuristiques sont des méthodes empiriques basées sur des règles simplifiées qui permet d'identifier en temps polynomial au moins une Solution réalisable rapide, pas obligatoirement optimale. On distingue [25]:

- **FIFO** (First In First Out) : la première tâche qui vient est la première tâche ordonnancée.
- **SPT** (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu.
- **LPT** (Longest Processing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu.
- **EDD** (Earliest Due Date) : la tâche ayant le date due la plus petite est la plus prioritaire.
- **SRPT** (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs.

- **ST** (Slack Time) : à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative.

2.2.2 Les méta-heuristiques

Les méta-heuristiques ont connu un essor considérable depuis leur apparition dans les années 1970. Elles sont présentées par Osman et Laporte (1996) [26] comme étant des méthodes d'approximation conçues dans le but d'une heuristique est de réussir a trouvé un optimal globale pour ce là, l'idée est à la fois de parcourir l'espace de recherche et explorer les zones qui paressent prometteuse mais sans être piégée par un optimum locale elles sont inspirées des processus naturels.

En générale, une résolution efficace d'un problème d'optimisation a besoin de tester plusieurs méthodes, le réglage de paramétrage de la méta-heuristique utilisée.

Les enjeux des méta-heuristiques groupées par Meignan [28] peuvent être présentés comme suit :

- La performance : tout comme les heuristiques, l'objectif d'une méta-heuristique est de fournir une méthode de résolution performante en termes de qualité de résultat et de temps de résolution.
- La simplicité de mise en œuvre : l'utilisation d'une méta-heuristique doit permettre de simplifier le développement par rapport à la mise en œuvre d'une heuristique spécifique au problème traité.
- La flexibilité : elle se définit par la capacité d'une méta-heuristique de traiter différents problèmes en conservant ses performances.

Donc on peut classer les méta-heuristiques en deux grandes familles: celles à population de solutions et les autres à base de solution unique.

2.2.2.1 Les méta-heuristiques à solution unique

Les méta-heuristiques à base de solution unique reposent sur la notion de recherche de voisinage. Elles démarrent d'une solution initiale (générée aléatoirement ou par une précédente méthode d'optimisation) puis, de manière itérative, substituent la solution courante par une autre solution, généralement meilleure, de son voisinage candidat. Le processus de recherche cesse lorsqu'un certain critère dit de continuation n'est plus avéré [22].

Nous présenterons ici les méthodes les plus utilisées et leur utilisation en extraction de connaissances : Les méthodes de descente, le recuit simulé et la recherche tabou.

- Les méthodes de descentes

Ces méthodes s'articulent toutes autour d'un principe simple. Partir d'une solution existante, chercher une solution dans le voisinage et accepter cette solution si elle améliore la solution courante. [27]

Initialise : trouver une solution initiale x
répéter
neighbourhood search : trouver une solution x' $\in$ N(x)
If f(x') < f(x) alors x' $\leftarrow$ x
End if
jusqu'à ce que f(y) >= f(x) ; v y $\in$ N(x)

Algorithme 3.1 l'algorithme de la méthode de descente.

L'algorithme 3.1 présente le squelette d'une méthode de descente simple. A partir d'une solution initiale x, on choisit une solution x' dans le voisinage N(x) de x. Si cette solution est meilleure que x, (f (x') < f(x)), alors on accepte cette solution comme nouvelle solution x et on recommence le processus jusqu'à ce qu'il n'y ait plus aucune solution améliorante dans le voisinage de x [27].

- le recuit simulé

Est un algorithme itératif qui apporte une solution au problème de l'attraction par les optimums locaux. Une solution voisine meilleure que la solution actuelle sera toujours acceptée. Cette méthode est transposée en optimisation pour trouver les extrêmes d'une fonction.

Le principe de cet algorithme est simple. Il commence par une température initiale élevée T et une solution initiale qui peut être choisie aléatoirement. A chaque itération de l'algorithme on génère une autre solution par une modification élémentaire de manière aléatoire de la solution courante, ce qui entraîne une variation de l'énergie (ΔE) du système donc de la fonction Objectif [29].
Selon ces mouvements, deux scénarios peuvent être considérés

- S'il y' a une amélioration de la fonction objectif, cette dernière solution est automatiquement acceptée.
- Si ce mouvement dégrade la fonction objectif, la nouvelle solution n'est pas forcément rejetée, et peut être acceptée avec une probabilité dépendante de la variation de la fonction objectif et de la température. Cette particularité d'acceptation de ces solutions permet de mieux explorer l'espace de recherche et ainsi la probabilité de trouver des solutions satisfaisantes [29].

- la recherche tabou

La recherche tabou a été formalisée par Glover en 1986. Le principe de base de la méthode tabou est simple, partant d'une solution initiale, son processus itératif, tente de converger vers la solution optimale en remplaçant à chaque itération la solution courante par la meilleure solution trouvée dans son voisinage.

De même, plusieurs stratégies peuvent être introduites afin d'améliorer l'efficacité de la méthode tabou, comme les deux concepts complémentaires l'intensification et la diversification. [29]

L'organigramme de cet algorithme est représenté sur la figure suivante :

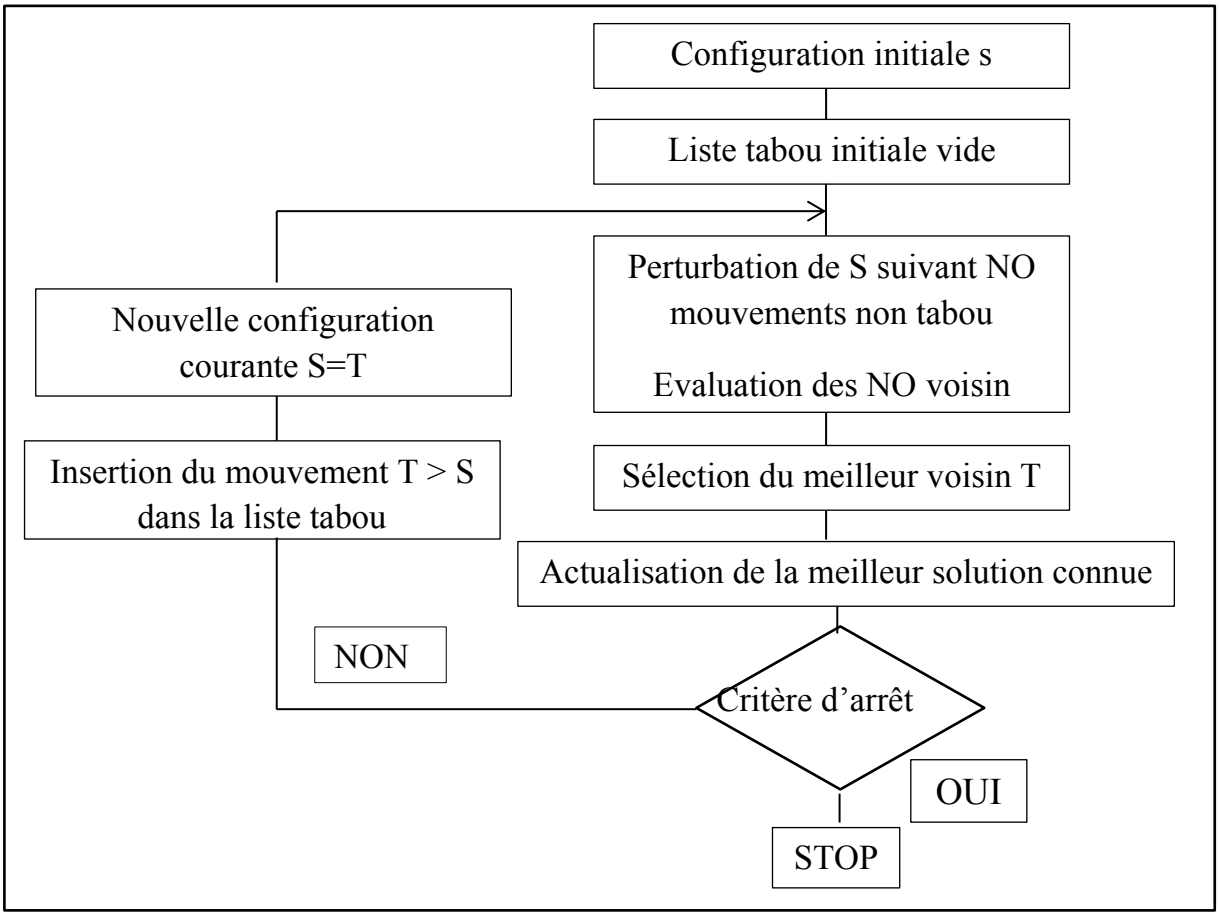


Figure 3.2 : L'organigramme de la recherche tabou [30].

2.2.2.2 Les méta-heuristiques à population de solutions

Les méthodes d'optimisation à population de solutions améliorent utilisent la population comme facteur de diversité. On distingue ici les méthodes les plus utilisées :

L'optimisation par colonie de fourmis et l'algorithme d'essaims particulaires, Les algorithmes génétiques.

- L'optimisation par colonie de fourmis

Un algorithme de colonies de fourmis est un algorithme itératif à population où tous les individus partagent un savoir commun qui leur permet de guider leurs futurs choix et d'indiquer aux autres individus des directions à suivre ou, au contraire, à éviter.

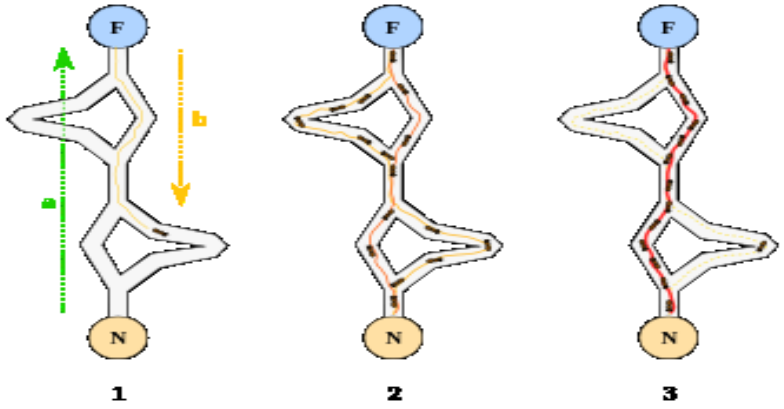


Figure 3.3 l'influence de l'expérience sur le choix des fourmis [31].

L'algorithme de colonies de fourmis pour un problème de voyageur de commerce (TSP) :

Tant que le critère d'arrêt n'est pas atteint faire
Pour k=1 à m faire
Choisir une ville au hasard
Pour chaque ville non visitée i faire
Choisir une ville j, dans la liste des villes restantes selon (P1)
Fin Pour
Déposer une piste sur le trajet (t) conformément à (P2)
Fin Pour
Évaporer les pistes selon (P3)
Fin Tant que

Algorithme 3.2 L'optimisation par colonie de fourmis [31].

Le principe de l'algorithme de colonies de fourmis est [31]:

P1) la première fourmi trouve la source de nourriture (F), via un chemin quelconque(a), puis revient au nid (N) en laissant derrière elle une piste de phéromone (b).
P2) les fourmis empruntent indifféremment les quatre chemins possibles, mais le renforcement de la piste rend plus attractif le chemin le plus court.
P3) les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromones.

- Essaims particulaires

L'optimisation par essaims particulaire, Initialement conçue pour résoudre des problèmes d'optimisation combinatoire continus. PSO est une méthode à population dont les individus, que nous appellerons particules, cherchent à reproduire le comportement de nuées d'oiseaux ou de bancs de poissons [32]. Le pseudo algorithme de PSO est le suivant :

n : nombre d'individus
Initialiser la population
Initialiser les paramètre
Tant que (critère d'arrêt est non atteint)
Pour *i* = 1 à *n* (pour chaque particule)
Évaluer fonction objectif
Si $f(x_i) > f(P_i)$ alors : (mise à jour du best local)
$Pi = xi$
Finsi
Si $f(xi) > f(P_g)$ alors : (mise à jour du best global)
$P_g = xi$
Finsi
Fin pour
Pour *i* = 1 à *n*
On met à jour la particule à l'aide des formules (3.2) et (3.3)
Fin pour
Fin tant que.

Algorithme 3.3 L'optimisation par essaims particulaire [32].

À chaque itération de l'algorithme, chaque particule possède une valeur de fitness (distance par rapport à la solution) qui peut être évaluée afin de mettre à jour les meilleures positions connues et se déplace dans l'espace de recherche selon sa vélocité qui représente la vitesse utilisée pour guider le déplacement de la particule et sa position dans l'espace de recherche. La structure de base de ce mouvement est illustrée par le schéma de l'algorithme 3.3. Chaque itération fait bouger chaque particule, en fonction de sa vitesse actuelle; sa meilleure solution *Pi*; la meilleure solution obtenue dans son voisinage *Pg*. En effet, en plus de la vitesse actuelle de la particule, le déplacement de la particule est influencé par deux autres

forces best local et global. La première l'attire à la position qui a donné la meilleure fitness pour la particule et l'autre c'est la meilleure position trouvée par la particule et ses voisins. A chaque instant *t*, la vélocité *Vi(t)* et la position *Xi(t)* sont mises à jour comme suit [29] :

$$V_i(t) = V_i\ (t\text{-}1) + C_1 r_1\ (P_i + X_i\ (t\text{-}1)) + C_2 r_2\ (P_g + X_i\ (t\text{-}1)) \qquad (3.2)$$

$$X_i\ (t) = X_i\ (t\text{-}1) + V_i(t) \qquad (3.3)$$

- Les algorithmes génétiques

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature : croisement, mutation, sélection.

Le principe d'un algorithme génétique est simple. On part d'une population de solutions du problème représentées par des individus. Cette population choisie aléatoirement constitue la population initiale. Chaque individu *x* de la population qui est codé sous forme d'une représentation génétique du problème, possède une fonction coût *f(x)* utilisée pour exprimer son degré d'adaptation à l'environnement ou à l'objectif visé, on dit qu'un individu est d'autant mieux adapté à son environnement ou plus performant que le coût de la solution qu'il représente est plus faible [29].

Les algorithmes génétiques sont appliqués sur une population d'individus, chacun de ces derniers est codé par un chromosome. Donc, une population est présentée par un ensemble de chromosomes. Le processus de codage d'individus consiste à trouver une structure de données pour les individus.

On distingue plusieurs types de codage, tels que : le codage binaire, le codage par permutation de valeurs entières et le codage par valeur, etc.

- Le codage binaire

Les premiers algorithmes génétiques ont utilisé le codage binaire. Un chromosome est représenté par une chaine binaire (chaine de bits) précisant l'information nécessaire à la description d'un individu [38].

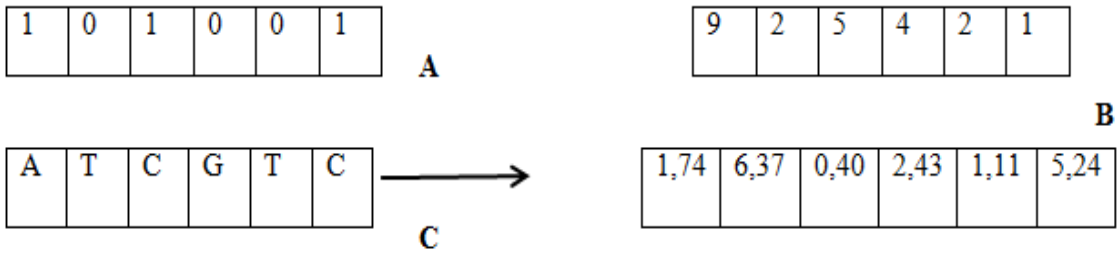
- Le codage par valeurs entières

Chaque gène est codé par une valeur entière. Donc, un chromosome est représenté sous forme d'une chaine d'entiers.

Ce codage est bien adapté à l'optimisation des problèmes industriels réels [38].

- Le codage par valeur

Dans ce type de codage, chaque gène est codé par une valeur qui appartient à un ensemble fini ou infini. Ces valeurs sont liées au problème à résoudre [38]



(A) : codage binaire,
(B) : codage par permutation de valeurs entières,
(C) : codage par valeur.

Figure 3.4 : Exemples de codage par valeurs [38]

Les opérateurs jouent un rôle prépondérant dans la possible réussite d'un AG. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement et de mutation. Le schéma d'un algorithme génétique est présenté dans la figure 3.5.

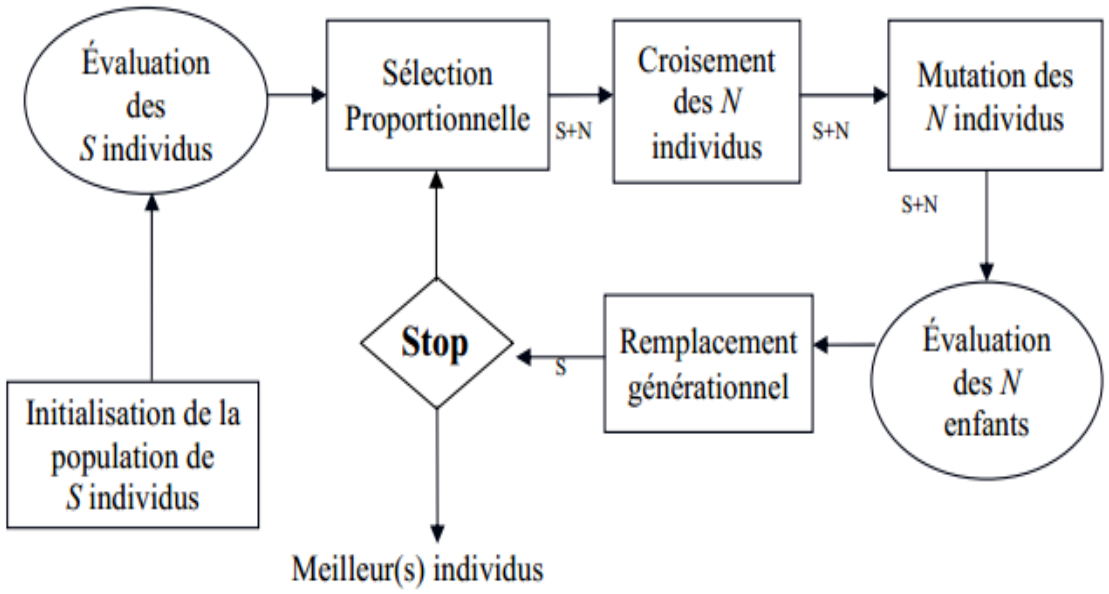


Figure 3.5 Schéma d'un algorithme génétique [30].

- Opérateur de Sélection

Le choix est de choisir les meilleurs individus à adapter pour avoir un certain nombre de solutions les plus proches de la convergence optimale globale. Cet opérateur applique le principe d'adaptation de la théorie de Darwin. Il existe plusieurs techniques de sélection.

Voici les principales utilisées:

- **Sélection par rang :** Cette technique de sélection sélectionne toujours les individus ayant le meilleur degré d'adaptation.
- **Probabilité de sélection proportionnelle à l'adaptation :** La technique de la roulette ou la roue de la chance pour chaque individu, la probabilité de la choisir est proportionnelle à son adaptation au problème.
- **Sélection par tournoi :** Cette technique est utilisée de manière proactive pour des paires d'individus, puis choisit parmi ces paires l'individu ayant le plus haut degré d'adaptation.
- **Sélection uniforme :** La sélection est aléatoire et uniforme et sans interférence avec la valeur d'adaptation.

- Opérateur de Croisement

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus). Tout d'abord, deux individus, qui forment alors un couple, sont tirés au sein de la nouvelle population issue de la reproduction. Puis un (potentiellement plusieurs) site de croisement est tiré aléatoirement (chiffre entre 1 et l - 1). Enfin, selon une probabilité pc que le croisement s'effectue, les segments finaux (dans le cas d'un seul site de croisement) des deux parents sont alors échangés autour de ce site (voir figure 3.6) [33].

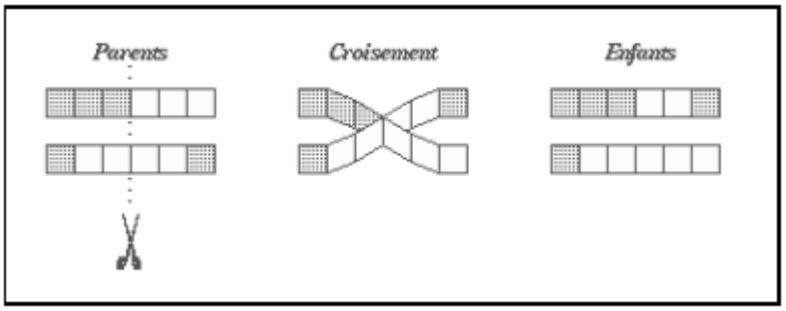


Figure 3.6 Le croisement en codage binaire [33].

- Opérateur de Mutation

Le rôle de cet opérateur est de modifier aléatoirement, avec une certaine probabilité, la valeur d'un composant de l'individu. Il s'agit de la modification aléatoire de la valeur d'un caractère de la chaîne Pour un codage binaire, elle consiste simplement à changer un 0 en un 1 (et réciproquement).

Le taux de mutation est généralement choisi très faible (≈ 0.001)
↳ pour chaque caractère des descendants, probabilité de 1/1000 qu'il mute.

Si la mutation joue un rôle secondaire (dû au taux faible), elle permet l'exploration de dimensions (éventuellement utiles), abandonnées (à tort) par le processus de sélection ou absentes de la population initiale.

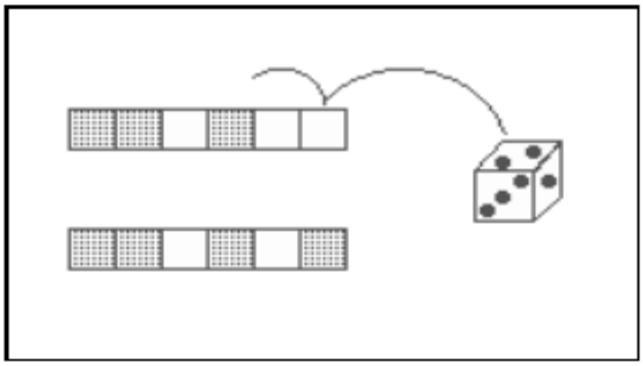


Figure 3.7 La mutation en codage binaire [33].

Les avantages des algorithmes génétiques : ont plusieurs avantages citons.

- Faculté d'adaptation aux différents problèmes.
- Facile à implémenter.
- Hybridation facile.
- Bonne approximation.
- Les AGs, à l'inverse des méthodes déterministes. recherchent d'une population de solutions à l'autre, plutôt que d'une solution à l'autre.

3. La comparaison de la méthode exacte et la méthode approchée

Exploration de S	**Systématique (intelligente)**	**Partielle (astucieuse)**
Méthode / solution	Exacte	Approchée
Caractéristique	Optimalité	Approximation
Preuve	Formelle	Empirique
Aspect	Déterministe	Stochastique (guidée)
Complexité	$O(2^n)$	$O(n^k)$
Handicap	Explosion combinatoire	Convergence précoce
Adaptée aux POC	De petites tailles	NP-difficiles de grandes tailles

Tableau 3.1 La déférence entre méthode exacte et méthode approchée.

Conclusion

Dans ce chapitre il existe deux types de méthodes d'optimisation, la première est l'optimisation mono-objectif, qui se base sur la minimisation (ou la maximisation) d'une seule fonction objectif. D'autre part, l'optimisation multi objectif optimise simultanément plusieurs fonctions objectives et on a étudié les différentes méthodes de résolution des problèmes d'ordonnancement. On distingue deux algorithmes : méthode exactes qui contient différentes méthodes comme PL, programmation dynamique et méthode approchés comme les heuristiques et Méta-heuristiques. Parmi ces méthodes on a choisi les algorithmes génétiques pour résoudre le problème d'ordonnancement d'un système flexible de production.

Le chapitre suivant sera consacré à l'implémentation et la mise en œuvre de l'algorithme génétique, ainsi l'étude et les résultats obtenus.

CHAPITRE 4

IMPLEMENTATION DE PROBLEME

Introduction

Dans ce chapitre, nous allons passer à l'implémentation du problème par l'algorithme NSGAII (Non-dominated Sorting Genetic Algorithm II). Dans un premier temps, nous présentons langage et l'environnement de développement que nous avons utilisé. Ensuite, on a présenté l'application. Après Nous avons procédé quelque exemples pour tester.

1. Langage et environnement de développement

Nous avons implémenté notre modèle en utilisant le langage JAVA car, il est considéré parmi les meilleurs langages qui prennent en charge la véritable programmation orientée objet (OOP).En plus, il se caractérise par une vaste collection de bibliothèque de classes. Mais, la particularité de ce langage par rapport aux autres réside sur tous les ordinateurs actuellement disponibles. L'environnement de développement utilisé est NETBEANS.

1.1 Langage de programmation Java

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C++. Il a été mis au point en 1991 par la firme Sun Microsystems. Il s'agissait de concevoir un langage bien adapté aux environnements de travail en réseau et capable de gérer des informations de nature variées (données numériques, informations sonores et graphiques). Java est devenue aujourd'hui une direction incontournable dans le monde de la programmation, parmi les différentes caractéristiques qui sont attribuées à son succès [34] :

- L'indépendance de toute plate-forme : le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine.
- Java est également portable, permettant à la simulation d'être distribuée facilement sans avoir à recompiler le code pour les différents systèmes.
- Le code est structuré dans plusieurs classes, dont chacune traite une partie différente de la simulation.
- Il assure la gestion de la mémoire.

- Java est multitâche : il permet l'utilisation de Threads qui sont des unités d'exécution isolées.

1.2 Environnement de développement

Netbeans est l'environnement de Développement Intégré (EDI) supporté par SUN. Il est particulièrement bien adapté pour le développement d'applications WEB. Il remplace l'IDE Java Studio Creator.

C'est un IDE moderne offrant un éditeur avec des codes couleurs et un ensemble de signes, des modèles de projets multi-langage et de différents types (application indépendante, distribuée, plugin, mobiles, ...), le refactoring, l'éditeur graphique d'interfaces et de pages web pour supporter le programmeur dans son travail. Il permet d'accéder rapidement à la documentation détaillée, de naviguer dans les sources et de faire des recherches d'usage des classes, méthodes et propriétés. Netbeans indique à l'utilisateur les erreurs et fait des propositions pour y remédier. Un débogueur permet l'exécution pas à pas. Un suivi des ressources utilisées (cpu, mémoire) par le logiciel développé peut être fait via un profiler. Un framework de test unitaire tel que Junit Fiche Junit peut être utilisé. [34]

2. Algorithme génétique de tri par non-dominance II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) est un algorithme de recherche couramment utilisées qui tente de trouver des solutions Pareto-optimale d'un problème d'optimisation multi-objectif. NSGA-II a les trois caractéristiques suivantes :

1. Il utilise un principe élitiste.

2. Il insiste sur les solutions non-dominées.

3. Il utilise un opérateur de comparaison basé sur un calcul de la distance de crowding.

Dans cet algorithme, à chaque génération t, une population de parents (P_t) de taille (N) et une population d'enfants (Q_t) de taille de même taille (N) sont mélangées pour produire une population ($R_t = P_t \cup Q_t$), comme le montre la figure 4.1. Cet assemblage permet d'assurer l'élitisme. La population R_t de taille ($2N$) est ensuite répartie par une procédure de tri selon un critère de non-dominance pour identifier plusieurs fronts $F1$, $F2$, etc. Les meilleurs individus vont se retrouver dans le ou les premiers fronts. Une nouvelle population parent (P_t+1) est formée en ajoutant les fronts au complet (premier front $F1$, second front $F2$, etc.) tant que ceux-ci ne dépassent pas N. Si le nombre d'individus présents dans ($Pt+1$) est inférieur à (N),

une procédure de crowding est appliquée sur le premier front suivant, (F_i), non inclus dans ($Pt+1$) [20].

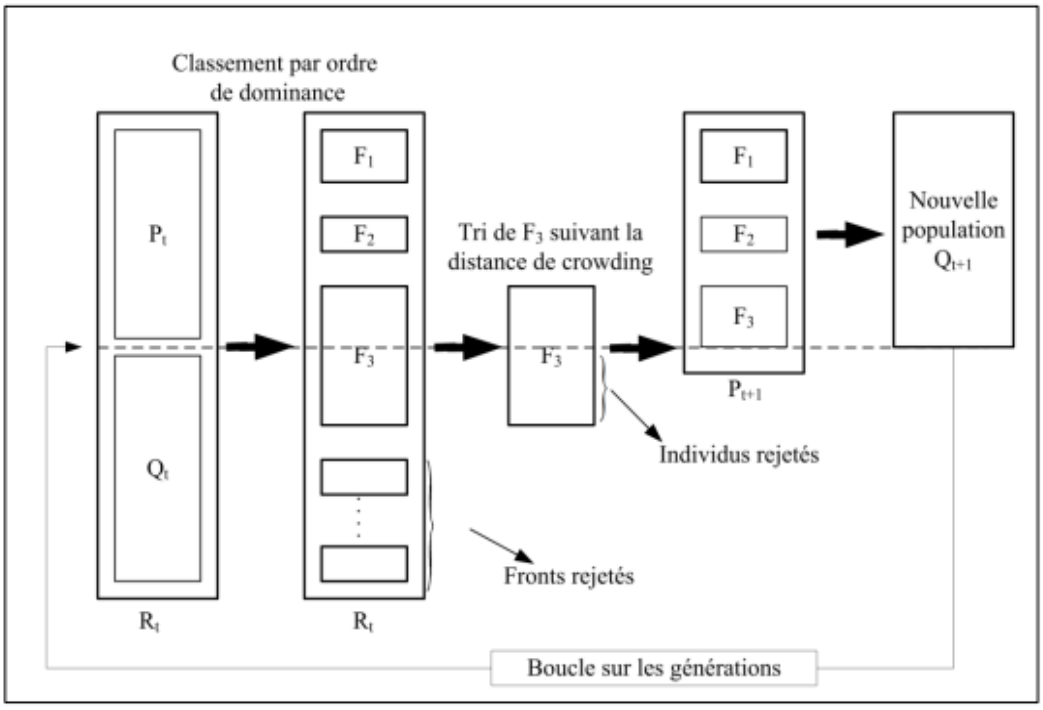


Figure 4.1 Représentation schématique de l'algorithme NSGA-II [21].

Le but de cet opérateur est d'insérer les (N - | P_{t+1}|) meilleurs individus de Fi qui manquent dans la population (P_{t+1}). Les individus de ce front sont utilisés pour calculer la distance de crowding entre deux solutions voisines. Une fois que les individus de la population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1}) est créée par sélection, croisement et mutation. La sélection par tournoi est utilisée mais le critère de sélection est basé sur l'opérateur de comparaison défini ci-dessous. Le processus continu, d'une génération à l'autre, jusqu'à satisfaction d'un critère d'arrêt. Une itération résume les différentes étapes décrites ci-dessus de l'algorithme *NSGA*-II :

- **Une itération de NSGA-II**

Après initialisation aléatoire de la population initiale P_0, une itération de NSGA-II se déroule comme suit [20]:

1. Créer Q_t à partir de P_t en utilisant le tournoi et en appliquant des opérateurs de variation génétique aux individus gagnants.

2. Réunir les populations des parents et des enfants $R_t = P_t \cup Q_t$. Trier l'ensemble résultant R_t en sous-ensemble F_i.

3. Soit une nouvelle population $P_{t+1} =$. Soit le compteur des sous-ensembles non-dominés i=1.

4. Tant que $|P_{t+1}|+|F_i| < N$, $P_{t+1} \leftarrow P_{t+1} \cup F_i$ et $i \leftarrow (i+1)$.

5. Ordonner l'ensemble F_i selon les ''distance de surpeuplement'' (crowding distance) et inclure $N - |P_{t+1}|$ solutions ayant les valeurs de distance les plus grandes dans la population P_{t+1}.

- **La distance de crowding :**

Pour une solution i on appelle « crowding distance » la distance moyenne entre les deux points (i-1) et (i+1) encastrant le plus large cuboïde contenant uniquement la solution i.
La figure suivante illustre la notion du crowding distance [35].

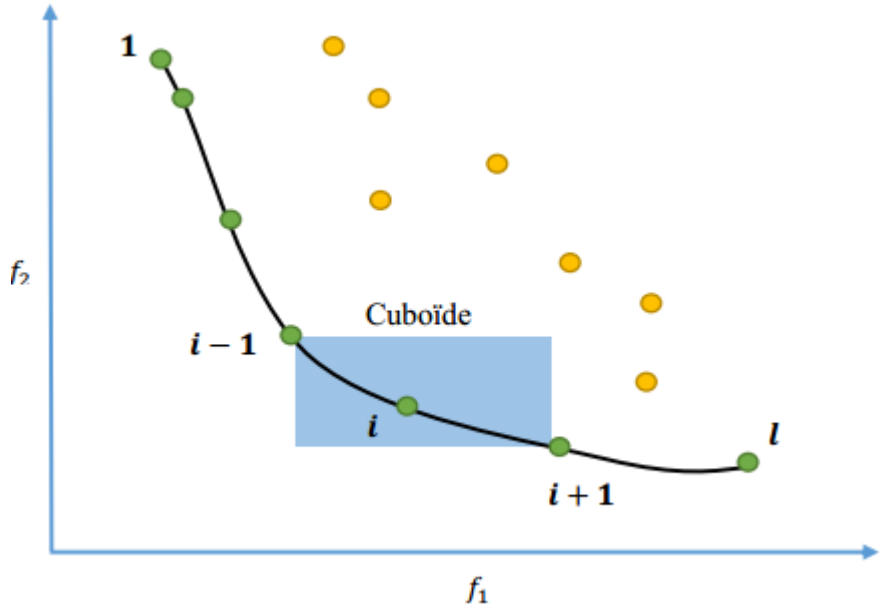


Figure 4.2 La distance de crowding [35].

3. Diagramme de Gantt

Le diagramme de Gantt est un outil utilisé en ordonnancement et gestion de projet et l'un des outils les plus efficaces pour présenter visuellement l'état d'avancement de différentes taches (toute tâche doit avoir une liaison entrante et sortante excepté le jalon de début du projet et celui de fin du projet), value (chaque tâche à une durée non nulle ou nulle).

Cet outil répond à deux objectifs: planifier de façon optimale et communiquer sur le planning établi et les choix qu'il impose.

4. Illustration de l'application

Notre application se compose de deux fenêtres d'affichage, la première représente l'interface principale qui permet de choisir des exemples existent qui contient par exemple 4 machines et 7 jobs en différents étages et la deuxième représente l'interface de l'implémentation de problème.

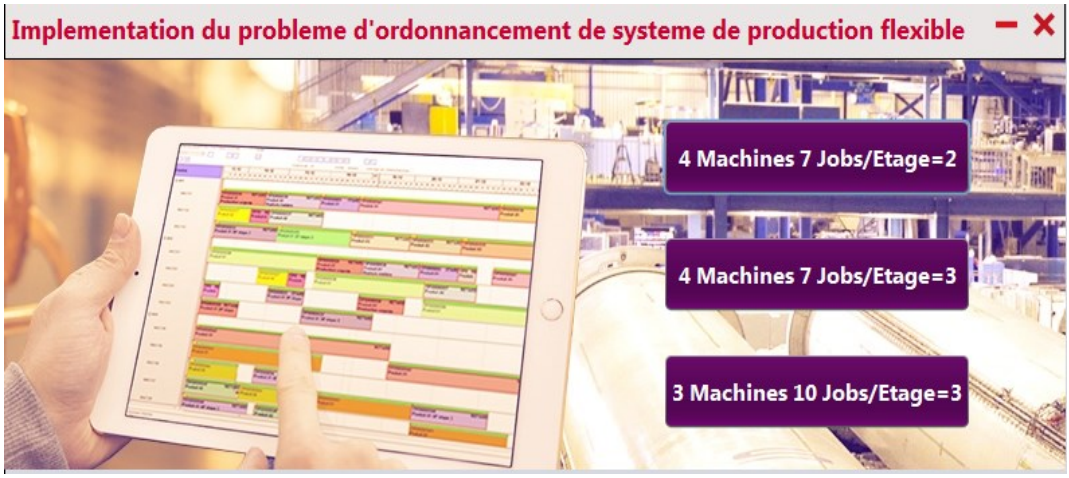


Figure 4.3 Interface principale.

A partir de cette interface, on pourra choisir l'exemple traité dans cette application pour afficher l'implémentation de problème.

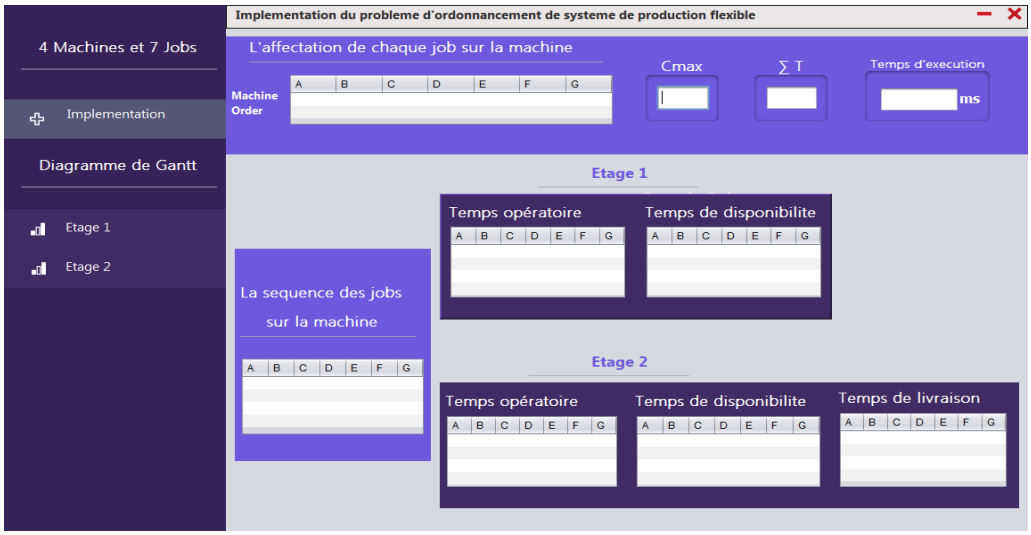


Figure 4.4 l'interface d'implémentation de l'exemple de 4 machines et 7 jobs en 2 étages.

Cette interface se compose en deux partie la première permet l'implémentation d'exemple et d'affiche le diagramme de Gantt et la seconde consiste d'afficher les résultats obtenus à partir d'optimisation de problème.

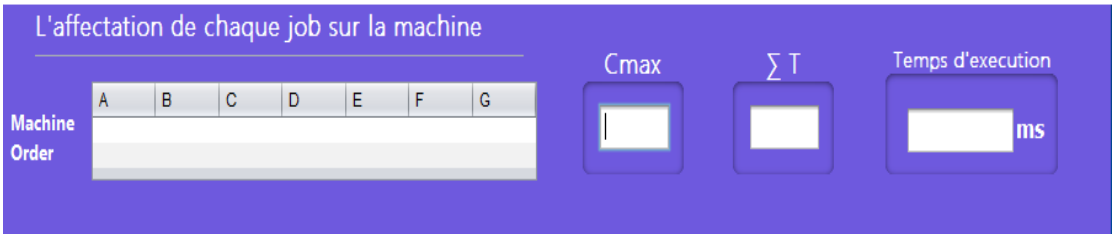


Figure 4.5 L'affichage des résultats obtenus à partir d'optimisation d'algorithme.

Dans la première case en trouve le tableau des variables de décisions obtenus à partir de l'optimisation de l'algorithme. La deuxième case représente le Cmax, la troisième représente la somme des retards et la dernière case représente le temps d'exécution de l'algorithme.

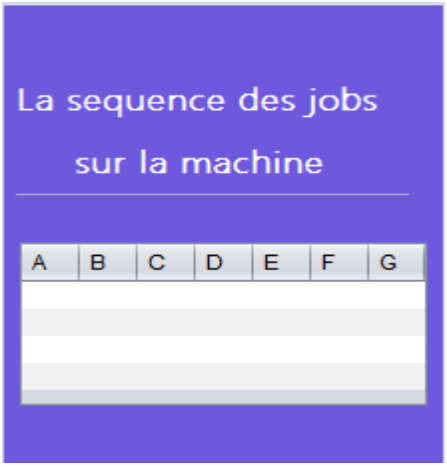


Figure 4.6 Matrice d'affichage de la séquence des jobs sur la machine.

Cette figure représente la séquence détaille des jobs sur chaque machine.

Figure 4.7 les paramètres du Flow shop hybride

Le Flow-shop hybride est un atelier composé de *N* étage tout dépends de l'industrie étudiée, donc nous proposons de fixer le nombre d'étage à N=2.

Les différentes matrices représentent le temps opératoire, le temps de disponibilité et le temps de livraison de la tâche.

5. Quelque exemple sur l'implémentation

Exemple1 : Dans un problème de 4 machines et 7 jobs avec 2 étages tels que les données sont:

- **Etage 1**

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P (temps de traitement)	4	8	10	6	7	6	3
R (temps de disponibilité)	0	3	2	5	6	8	10

Tableau 4.1 La table des taches (étage 1, exemple1).

Avec notre implémentation nous avons trouvé plusieurs solutions optimales d'ordre de l'ordonnancement on a choisi cette solutions optimale comme suite :

• La solution donne l'ordre de l'ordonnancement suivant :

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
N (indice de machine)	4	2	4	1	2	3	1
K (indice de position sur la machine)	1	1	2	1	2	1	2

Tableau 4.2 La table d'affectation de job sur la machine (exemple1).

au début temps on a n=1, k=1 les matrices Pr1, R1,C1 comme suite :

Alors que Pr1 c'est la matrice de temps opératoire de la tâche *j* , R1 c'est la matrice de date de début, C1 c'est la matrice de makespan.

$$Pr1=\begin{pmatrix} 6&3&0&0&0&0&0\\ 8&7&0&0&0&0&0\\ 6&0&0&0&0&0&0\\ 4&10&0&0&0&0&0 \end{pmatrix} \quad R1=\begin{pmatrix} 5&10&0&0&0&0&0\\ 3&6&0&0&0&0&0\\ 8&0&0&0&0&0&0\\ 0&2&0&0&0&0&0 \end{pmatrix} \quad C1=\begin{pmatrix} 11&14&0&0&0&0&0\\ 11&18&0&0&0&0&0\\ 14&0&0&0&0&0&0\\ 4&12&0&0&0&0&0 \end{pmatrix}$$

- Diagramme de Gantt :

Figure 4.8 Diagramme de Gantt de 1ere exemple de l'étage 1.

- **Etage 2**

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P2 (temps de traitement)	6	5	8	6	4	7	4
D (temps de livraison)	22	30	26	24	28	26	26

Tableau 4.3 La table des taches (étage 2, exemple1).

Au début temps on a n=1, k=1 les matrices Pr2, R2, D, comme suite:

D c'est la matrice de date de fin la tâche j.

$$Pr2=\begin{pmatrix}6&4&0&0&0&0&0\\5&4&0&0&0&0&0\\7&0&0&0&0&0&0\\3&8&0&0&0&0&0\end{pmatrix}\quad R2=\begin{pmatrix}11&14&0&0&0&0&0\\11&18&0&0&0&0&0\\14&0&0&0&0&0&0\\4&12&0&0&0&0&0\end{pmatrix}\quad D=\begin{pmatrix}24&26&0&0&0&0&0\\30&28&0&0&0&0&0\\26&0&0&0&0&0&0\\22&26&0&0&0&0&0\end{pmatrix}$$

Alors on a C_{max} = 22, ∑T=0.

- Diagramme de Gantt :

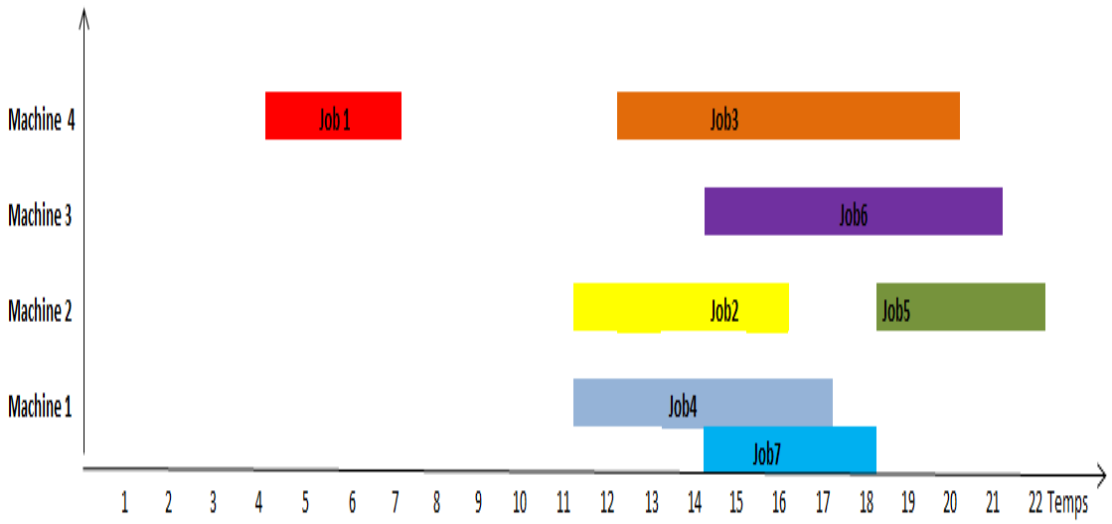


Figure 4.9 Diagramme de Gantt de 1ere exemple de l'étage 2.

Exemple2 : Dans un problème de 4 machines et 7 jobs avec 3 étages tels que les données sont :

- **Etage 1**

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P (temps de traitement)	6	10	8	5	7	6	3
R (temps de disponibilité)	0	2	2	4	6	8	10

Tableau 4.4 La table des taches (étage 1, exemple2).

Avec notre implémentation nous avons trouvé plusieurs solutions optimales d'ordre de l'ordonnancement on a choisi cette solutions optimale comme suite :

- La solution donne l'ordre de l'ordonnancement suivant :

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
N (indice de machine)	4	2	4	1	2	3	1
K (indice de position sur la machine)	1	1	2	1	2	1	2

Tableau 4.5 La table d'affectation de job sur la machine (exemple2).

au début temps on a n=1,k=1 les matrices Pr1, R1,C1 comme suite :

$$Pr1=\begin{pmatrix} 8&7&0&0&0&0&0\\ 5&3&0&0&0&0&0\\ 6&6&0&0&0&0&0\\ 10&0&0&0&0&0&0 \end{pmatrix} \quad R1=\begin{pmatrix} 2&6&0&0&0&0&0\\ 4&10&0&0&0&0&0\\ 0&8&0&0&0&0&0\\ 2&0&0&0&0&0&0 \end{pmatrix} \quad C1=\begin{pmatrix} 10&17&0&0&0&0&0\\ 9&13&0&0&0&0&0\\ 6&14&0&0&0&0&0\\ 12&0&0&0&0&0&0 \end{pmatrix}$$

- Diagramme de Gantt :

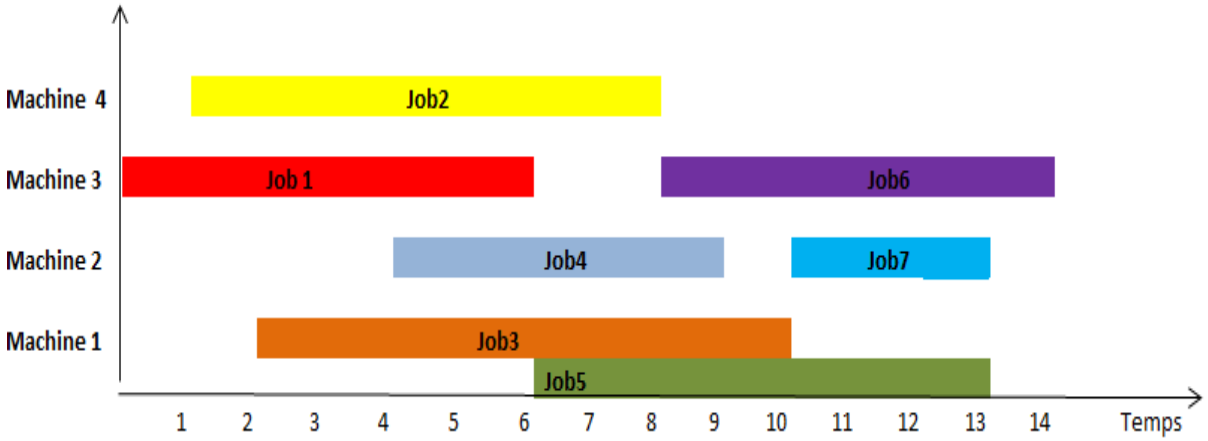


Figure 4.10 Diagramme de Gantt de 2eme exemple de l'étage 1.

- **Etage 2**

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P2 (temps de traitement)	6	5	8	6	4	7	4

Tableau 4.6 La table des taches (étage 2, exemple2).

au début temps on a n=1, k=1 les matrices Pr2, R2, C2 comme suite:

$$Pr2=\begin{pmatrix} 8&3&0&0&0&0&0\\ 6&4&0&0&0&0&0\\ 5&7&0&0&0&0&0\\ 4&0&0&0&0&0&0 \end{pmatrix} \quad R2=\begin{pmatrix} 10&17&0&0&0&0&0\\ 9&13&0&0&0&0&0\\ 6&14&0&0&0&0&0\\ 12&0&0&0&0&0&0 \end{pmatrix} \quad C2=\begin{pmatrix} 18&21&0&0&0&0&0\\ 15&19&0&0&0&0&0\\ 11&21&0&0&0&0&0\\ 16&0&0&0&0&0&0 \end{pmatrix}$$

- Diagramme de Gantt :

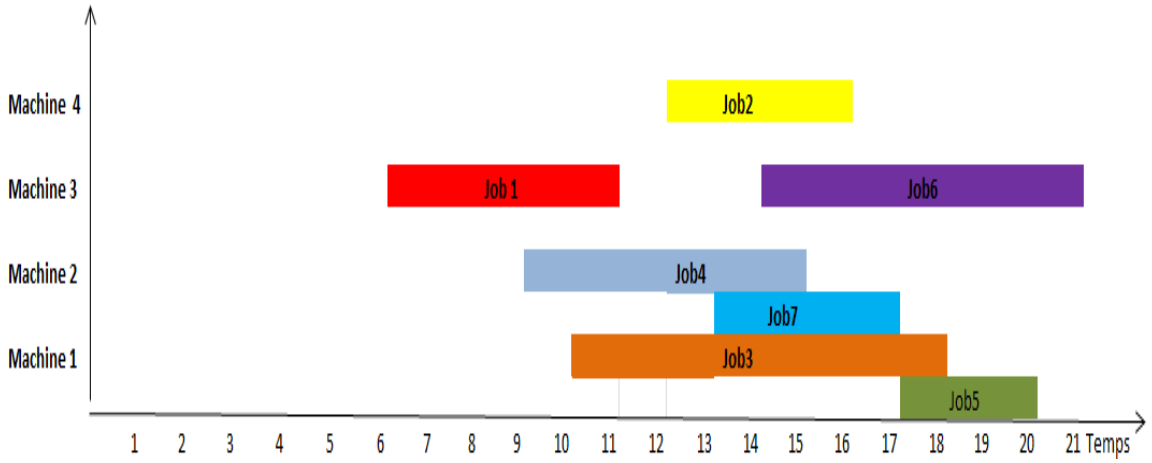


Figure 4.11 Diagramme de Gantt de 2eme exemple de l'étage 2.

- **Etage 3**

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P3 (temps de traitement)	6	5	8	6	4	7	4
D (temps de livraison)	22	30	26	24	28	26	26

Tableau 4.7 La table des taches (étage 3, exemple2).

Au début temps on a n=1, k=1 les matrices Pr2, R2, D, comme suite:

D c'est la matrice date de fin la tâche *j*.

$$Pr3=\begin{pmatrix}5&3&0&0&0&0&0\\6&4&0&0&0&0&0\\3&5&0&0&0&0&0\\7&0&0&0&0&0&0\end{pmatrix}\quad R3=\begin{pmatrix}18&21&0&0&0&0&0\\15&19&0&0&0&0&0\\11&21&0&0&0&0&0\\16&0&0&0&0&0&0\end{pmatrix}\quad D=\begin{pmatrix}26&28&0&0&0&0&0\\24&26&0&0&0&0&0\\22&26&0&0&0&0&0\\30&0&0&0&0&0&0\end{pmatrix}$$

Alors on a C_{max} = 26, ∑T=0.

- Diagramme de Gantt :

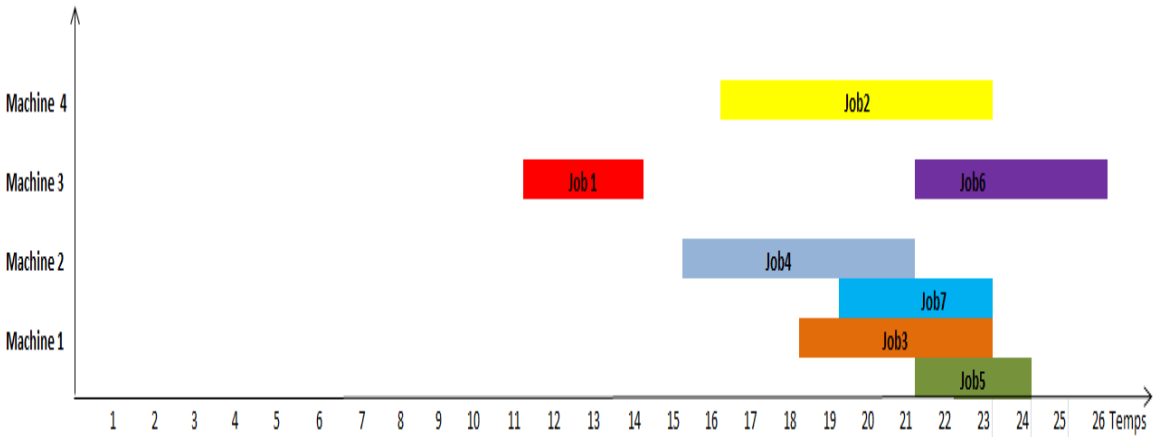


Figure 4.12 Diagramme de Gantt de 2eme exemple de l'étage 3.

Conclusion

Dans ce chapitre, nous avons parlé d'algorithme NSGA-II, le langage et son outil que nous avons utilisé pour résoudre le problème que nous avons abordé dans cette mémoire. Ensuite, nous avons présenté l'implémentation de notre application et nous avons pris des exemples ainsi que les résultats obtenues.

CONCLUSION GENERALE

La production est un facteur commun qui mène à la concurrence entre les entreprises. Pour cela, Il est devenu indéniable que la qualité d'un produit et ses coûts parmi les critères nécessaire de performances et dans un cadre d'amélioration permanente.

Le problème traité dans ce travail est le problème d'ordonnancement de type flow shop hybride dans les systèmes de production flexibles. Pour résoudre le problème, nous avons proposé une méthodologie de résolution du problème par l'optimisation multi objectif basée sur un algorithme génétique et nous avons utilisé NSGA II comme méthode de résolution à cause des caractéristiques qu'elle présente en particulier la rapidité de tri et l'élitisme, pour développer notre application qui minimise le makespan et la somme des retards.

Dans le premier chapitre, nous avons présenté tout d'abord une présentation générale sur la chaine logistique. Ainsi, nous avons présenté les fonctions et les classe de base de la chaine logistique. Ensuite, nous avons défini la gestion de la chaîne logistique et les niveaux de décisions et leurs optimisations.

Dans le deuxième chapitre nous avons donné une généralité sur les problèmes d'ordonnancement avec les différents types d'ateliers.

Dans le troisième chapitre, nous avons présenté les problèmes d'optimisation combinatoire avec une présentation plus détaillés sur les méthodes de résolution combinatoire.

Dans la quatrième et le dernier chapitre, nous avons vue langage et l'environnement de développement qui ont été utilisé. Ainsi, l'implémentation de l'algorithme NSGA II (Non-dominated Sorting Genetic Algorithm II) qui ont été appliquée avec succès à la résolution de problèmes d'ordonnancement de système de production flexible pour différents critères.

Comme perspectives, nous proposons d'examiner une nouvelle approche d'évaluation multicritères basée sur le principe de la logique flou. Cette approche est établit sur une hybridation des algorithmes génétiques et la logique floue et de faire une comparaison du NSGA II actuellement utilisé par autres modèles d'algorithmes génétiques multicritères.

BIBLIOGRAPHIES

[1] Colin. J, Math. H, Rixier. D, la logistique au service de l'entreprise. Dunod, Entreprise, 1983.

[2] Christopher, M.L, Logistics and supply chain management. Pitman Publishing, London,

[3] S. Lamia, Recherche opérationnelle, Université Aboubakr Belkaïd Tlemcen, P 52, 2016.

[4] R. Ganeshan, P. Terry. Harrison, An Introduction to Supply Chain management. Departement of Management Sciences and Information Systems, 303 Beam Business Building, Penn State University, University Park, PA, USA, (1995).

[5] Z. Hadjila, I.Mansouri, La conception d'une chaine logistique pour la distribution des produits pharmaceutique Application sur la ville de Tlemcen, Université Aboubakr Belkaïd Tlemcen, 2016.

[6] H. Ding, Une approche d'optimisation basée sur la simulation pour la conception des chaînes logistiques : Applications dans les industries automobile et textile. Thèse de Doctorat, université de Metz (2004).

[7] Z. Mouloua, Ordonnancements coopératifs pour les chaînes logistiques, Ecole Supérieure des Mines de Nancy (2011).

[8] D. Zahéra, Impact du conditionnement intelligent sur la planification et la traçabilité d'une chaine logistique Agroalimentaire.

[9] Simchi-Levi, Kaminsky, Supply chain simulation and analysis with simflex, 2003.

[10] Y. Bahmani, Optimisation multicritère de l'ordonnancement des activités de la production et de la maintenance intégrées dans un atelier Job Shop, Thèse Doctorat en science, Université de Batna-II, Algérie 2017.

[11] A. Kaddoussi, Optimisation des flux logistiques : vers une gestion avancée de la situation de crise, P 25, 2013.

[12] C. Abderrahman, D. Abd Naceur, Ordonnancement d'un flow-shop par méta-heuristique hybride, Pour l'obtention du diplôme de Master en Génie industriel, Université Abou Bekr Belkaid – Tlemcen, 2017.

[13] I.Nasri, Développement d'une méthodologie d'ordonnancement/optimisation adaptée aux systèmes industriels de type HVLV (HighVariety, Low-Volume).Thèse de Doctorat, université de Grenoble, P 60,2013.

[14] Wikipédia, https://fr.wikipedia.org/wiki/NP-difficile , consulté le 06 mai 2019.

[15] Erschler. J, De Terssac G., « Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production », Rapport laas no 88137, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse, France, 1988.

[16] Thomas.C, « Analyse de la flexibilité : le cas d'une unité de production d'aluminium », Thèse de doctorat, Institut National Polytechnique de Grenoble, 3003.

[17] Wikipédia, www.wikipédia.ordonnancement_d'atelier.com, consulté le 06 mai 2019.

[18] T. Chaari. « Un algorithme génétique pour l'ordonnancement robuste : application au problème du flow shop hybride, thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis », P37, 2010.

[19] W. Trabelsi. « Ordonnancement des systèmes de production flexibles soumis à différents types de contraintes de blocage, thèse de doctorat, Université de Loraine ». P35-36, 2012.

[20] M. Bounif, Optimisation à base de simulation pour le développement des systèmes décisionnels, Thèse Doctorat 3éme cycle en informatique, université de M'sila, 2014/2015

[21] O. Guenounou, Méthodologie de conception de contrôleurs intelligents par l'approche génétique- application à un bioprocédé, Thèse de doctorat, L'Université de Toulouse, (2009)

[22] Z. Deffas, Méta-heuristiques parallèles à solution unique pour la résolution du problème du Q3AP sur grille de calcul, Thèse de Doctorat, université d'OUM EL BOUAGHUI, P 27, 2010.

[23] S. Triqui L, Recherche opérationnelle, Université Aboubakr Belkaïd Tlemcen, P 52, 2016

[24] Wikipédia, www.wikipédia.programmation_par_contrainte.com, consulté le 23 Décembre 2018.

[25] C.Chu, J .Proth, l 'ordonnancement et ses applications. Sciences de l'ingénieur, Edition Masson, Paris, 1993.

[26] I.H.Osman, G.Laporte. Metaheuristics: a bibliography. Annals of Operations Research 63, P.513-623, 1996.

[27] A. CHERGUI, A. DAHMANI, Ordonnancement d'un flow-shop par méta-heuristique hybride, Pour l'obtention du diplôme de Master en Génie industriel, Université Abou Bekr Belkaid – Tlemcen, 2017.

[28] Meignan. D, Une approche organisationnelle et multi-agent pour la modélisation et l'implantation de méta-heuristiques, Application aux problèmes d'optimisation de réseaux de transports. Thèse de doctorat, Université de Technologie de Belfort-Montbeliard, France, 2008.

[29] S. Mehdi, Investigations sur la sélection de routages alternatifs en temps réel basées sur les méta-heuristiques-les essaims particulaires, Thèse de Doctorat en Sciences en Productique, Université de Tlemcen, 2012

[30] Dréo. J, Pétrowski. A, Siarry, P. et Thaillard, E. (2003). Méta-heuristiques pour l'optimisation difficile. Eyrolles, 2003.

[31] Wikipédia, https://fr.wikipedia.org/wiki/Algorithme_de_colonies_de_fourmis, consulté le 28 Décembre 2018

[32] L. Deroussi, Méta-heuristiques pour la logistique, Livre, P 69, ISTE Editions, 2016

[33] T. Vallée, M. Yıldızoglu, Présentation des algorithmes génétiques et de leurs applications en économie, Université de Nantes, LEA-CIL

[34] K. Abderrahimi hidayet, S. Benali Amar, Algorithme génétique parallèle applique à un problème d'optimisation combinatoire, Mémoire de master en RISR, université de Dr. Tahar Moulay Saida, PP 69-70 ,2017.

[35] B. Abdelouhab, B.Yacin, Etude comparative des algorithmes génétiques multi-objectifs, Mémoire de master en automatique, université d'Abderrahmane MIRA de Bejaia, PP 25 ,2015

[36] Chopra S., Meindil P., *Supply chain management: Strategy, planning, and Operations, third edition*, Pearson Education, Inc. New Jersey (2007)

[37] Pitchot. L, Stratégie de déploiement d'outils de pilotage de chaines logistiques : Apport

de la classification, thèse de doctorat, L'institut national des sciences appliquées de Lyon, 2006

[38] M. Meziane, Optimisation par phases pour le probleme d'ordonnancement des ateliers de type job shop totalement flexibles, Magister, ORAN, 2011.

الملخص

في ورشة تصنيع متخصصة، تسعى إدارة الإنتاج إلى تحسين تدفق المنتجات من خلال الجدولة. لحل مشكلة جدولة نظام الإنتاج المرنة، استخدمنا جدولة ورشة عمل مختلطة أحادية المسار حيث يعبر كل أمر إنتاج المراحل بنفس الترتيب. هدفنا هو تقليل الوقت الإجمالي ومجموع التأخير. بحيث أن هذه المعايير تسمح بتحسين الإنتاجية. لحل هذا النوع من المشكلات، اقترحنا طرق تحسين متعددة المعايير حيث أنها تسمح بحل المعايير التي حددناها. من بين هذه الطرق ، هنالك "الخوارزميات الجينية".

الكلمات المفتاحية: سلسلة التوريد , الجدولة , ورشة عمل مختلطة أحادية المسار , الخوارزمية الجينية.

Résume

Dans un atelier spécialisé de fabrication, la gestion de la production cherche à améliorer le flux des produits à travers l'ordonnancement. Pour la résolution de problème d'ordonnancement de système de production flexible, nous avons utilisé l'ordonnancement d'atelier de type Flow-Shop hybride où chaque ordre de production traverse les étages dans le même ordre. Notre objectif est toujours de minimiser le temps total d'exécution (makespan) et la somme des retards. Ces critères permettent de maximiser la productivité. Pour résoudre ce type de problème, nous avons proposé les méthodes d'optimisation multi critère qui sont des méthodes permettant de résoudre les deux critères d'optimisation qui nous avons spécifié. Parmi ces méthodes, apparaissent celles dites «les algorithmes génétique ».

Mots clé: chaine logistique, Ordonnancement, Flow shop Hybride, Algorithme génétique.

Abstract

In a specialized manufacturing workshop, production management seeks to improve the flow of products through scheduling. In order to solve the flexible production system scheduling problem, we used hybrid Flow-Shop type workshop scheduling where each production order crosses floors in the same order. Our objective is always to minimize the total execution time (makespan) and the sum of delays. These criteria help to maximize productivity. To solve this type of problem, we have proposed multi-criteria optimization methods that are methods to solve the two optimization criteria that we have specified. Among these methods, there are those called "genetic algorithms".

Key words: supply chain, Scheduling, Flow shop Hybrid, Genetic algorithms.