REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE Mathématique et Informatique
DEPARTEMENT D' Informatique
N° : 53

DOMAINE : Mathématique et Informatique
FILIERE : INFORMATIQUE
OPTION : INFORMATIQUE DECISIONNELLE ET OPTIMISATION

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par: Adjiri Hadjer

Intitulé

Les problèmes d'ordonnancement d'atelier :

M-Machine identique en parallèle

Soutenu devant le jury composé de :

Gasmi Abdelkader	Université de M'sila	Président
Bounif Mohamed El Hadi	Université de M'sila	Co-encadreur
Mouhoub Nacer Eddine	Université de M'sila	Rapporteur
Debbi Aimad-Eddine	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

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 ***Mr. MOUHOUB NACER EDDINE*** *et co-encadreur* ***Mr. BOUNIF Mohamed El hadi*** *pour m'avoir assister 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.*

Enfin je tiens aussi à remercier tous mes amis et collègues d'étude qui mon aider de près ou de loin d'avoir réalisé ce travail.

Dédicace

Je dédie ce modeste travail à :

- ·***Mes parents*** *: qui ont la cause de la lumière de mes jours, la source de mes forces, la flamme de mon cœur, ma vie et mon bonheur.*
- ***A mon très chèr père MABROUK ADJIRI,*** *qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de vous. Papa j'adore.*
- ***A ma très chère mère ZOUBIDA BENSEDEK****, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude. Maman j'adore.*
- ***A mes beaux-frères et ma sœur : Abdou, Bahi, Saber, Dounia, Mohamed et son épouse Aicha et leurs petite fille Houria*** *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 mes chère collègues d'études et frères de cœur Ali Zitouni Mohammed Abdelmalik, Bahache Anwar, Messeguem Chahrazed, Bernis Imane, Rahenbi Asma, Zaiter Meriem, Bouhali Maroua, Selam Maroua, Khenache Bouchra, Gahgah Hadjer :*** *qui m'ont toujours à mes côtés, et qui m'ont accompagnaient durant mon chemin d'études et ont été témoins de ce processus et ils m'ont soutenu dans toute ma carrière. En souvenir de notre sincère et profonde amitié et des moments agréables que nous avons passés ensemble. 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.je vous aime en dieu.*
- ***A tous mes proches la famille ADJIRI et BENSEDEK*** *avec tous mes sentiments de respect et d'amour* **.**
- ***A*** *mon encadreur* ***Mr. Mouhoub Nacer Eddin et*** *co-encadreur* ***Mr.Bounif Mohamed Elhadi,*** *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

Liste des Figures

Liste des Tables

Liste des algorithmes

Liste des abréviations

ACM *Les Ateliers Cheminements Multiples*

PPC *Programmation Par Contrainte*

CP *Constraint Programming*

CSP *Constraint Satisfaction Problem*

FIFO *First In First Out*

SPT *Shortest Processing Time*

LPT *Longest Processing Time*

EDD *Earliest Due Date*

AG *Algorithme Génétique*

ACO *Ant Colony Optimization*

TSP *Traveling Salesman Problem*

PL *Programmation Linéaire*

NSGA-II *Non-dominated Sorting Genetic Algorithm II*

NGPM *NsGa-II Program in Matlab*

INTRODUCTION GENERALE

La concurrence industrielle nécessite d'utiliser au mieux toutes les ressources potentielles de l'entreprise. Cette optimisation doit traiter les problèmes plus spécifiques à tous les niveaux, et notamment au niveau de la production et la gestion de ressources.

Dans un tel contexte, rester toujours performant, passe obligatoirement par une gestion de production plus fiable. En effet, l'amélioration de la gestion de production vise à organiser le fonctionnement du système de production, et à mieux gérer ses différentes opérations. Ainsi, l'amélioration de la production dépend étroitement de l'ordonnancement d'ateliers.

L'ordonnancement d'ateliers consiste à prévoir l'enchaînement de toutes les opérations élémentaires nécessaires à la réalisation des ordres de fabrication sur les ressources de production tout en tenant compte des contraintes internes et externes.

L'ordonnancement en gestion de production correspond aux concepts et techniques qui règlent les problèmes de priorités et l'organisation des flux de production. Dans un environnement de production complexe, l'ordonnancement peut devenir un problème extrêmement difficile à résoudre. De manière générale, on entend par ordonnancement de la production, la détermination de l'ordre de passage d'un certain nombre de travaux à l'exécution. Cette détermination concerne la planification de l'utilisation des ressources disponibles en homme et en machine afin de mieux contrôler les coûts et de maîtriser les délais de fabrication des productions décidées.

La résolution d'un problème d'ordonnancement passe par une phase d'identification et de modélisation et par une phase de recherche de la méthode de résolution adéquate. Dans la première phase, les contraintes à respecter et les critères à optimiser, lors de la résolution sont identifiés et décrits.

Notre travail s'articule autour de cette problématique dans le but de développer un outil d'ordonnancement qui respecte ces contraintes spécifiques à la gestion de la production et la résolution d'un problème d'ordonnancement, est la recherche et le choix de la méthode de résolution adéquate. Dans cette optique, notre travail propose des solutions pour le problème d'ordonnancement.

Le premier chapitre de ce mémoire propose, une vue d'ensemble sur les problèmes d'ordonnancement. Nous rappelons d'abord les différents éléments qui composent un

problème d'ordonnancement et nous présentons ensuite une typologie des problèmes d'ordonnancement en distinguant les différents types d'ateliers. Dans le deuxième chapitre on parle des modèles mathématiques d'un problème d'ordonnancement. Le chapitre trois est consacré aux différentes méthodes de représentation des problèmes d'ordonnancement. Nous nous sommes intéressés, par ailleurs, à la présentation des différentes méthodes d'optimisation développées dans la littérature, classées en méthodes exactes et en méthodes approchées. Le quatrième chapitre présente notre implémentation qui dans cet implémentation on va utiliser l'algorithme génétique et particulièrement la méthode de NSGA II. Enfin, nous terminons par une conclusion, quelques remarques et perspectives.

CHAPITRE 1
PROBLEME D'ORDONNANCEMENT D'ATELIER

Introduction

L'ordonnancement est la fonction d'exploitation qui gère les planifications de traitements informatique. Son rôle principal est la soumission des taches, ou de séquence des taches, à heures définies lorsque l'ensemble des conditions sont remplies. Ces conditions sont généralement des critères de dates, de statuts de prédécesseurs.

Dans ce chapitre introductif, nous cherchons d'abord à positionner le concept d'ordonnancement pour la gestion de la production, ensuite nous allons représenter les bases de l'ordonnancement dans les ateliers. La section suivante sera consacrée à différents types d'ateliers de production. Ensuite, nous allons expliquer les concepts de complexité algorithmique et problématique.

I- La gestion de production

1- Définition et rôle de la production

- La gestion de la production est « la fonction qui permet de réaliser les opérations de production en respectent les conditions de qualité, délai, cout qui résultent des objectifs de l'entreprise [13].

- La gestion de la production est la mise en application de méthodes et techniques dans le but d'accomplir la transformation des matières en produits fini. Elle se résume en la combinaison de ressources, parmi lesquelles les moyens matériels (les machines), les moyens humains (le personnel par qualification) et les matières (matières premières, matières consommables) dans un planning avec pour but d'assurer la fabrication du produit en qualité et en quantité définies.

- Dans un environnement économique devenu aussi concurrentiel que le nôtre, les enjeux financiers sont cruciaux. Le prix de vente des produits dépend de plus en plus de la demande du marché et reste très influence par la concurrence. Afin de rester compétitive et surtout garantir une marge bénéficiaire convenable sur la vente de leurs produits, les entreprises industrielles ont pour principal recours la réduction

du cout de production. Le champ d'action de la gestion de la production dans l'entreprise est vaste, couvre de nombreuses activités et interpelle les professionnels de différents domaines de formation.

Les contraintes rencontrées sont de divers ordres :

− Financières (produire a un cout optimal), cout des matières et consommables, cout de stockage des encours et des produits semi ouvres, cout de gestion des magasins, cout des heures de travail supplémentaires, cout des arrêts faisant partie intégrante du cout de revient, maitriser ces derniers est aussi une garantie pour la commercialisation des produits finis,

− Temporelles (produire dans les délais, assurer une livraison juste a temps), éviter les ruptures de stocks, éviter le gonflage des stocks de produits finis. Car cela a une incidence directe sur la satisfaction de la clientèle (pertes de commandes) ou sur le cout de revient du produit finis dû aux couts supplémentaires du stockage,

− Mécaniques (maintenance préventives et gestion des temps d'arrêt), anticiper sur les pannes et prévoir des solutions alternatives en cas d'arrêt d'une machine,

− Qualité (produire avec le moins de défaut possible, le moins de déchets), un produit de bonne qualité participe a la fidélisation de la clientèle, véhicule l'image de marque de l'entreprise.

− Planification : assurer une circulation continue des flux, détecter et supprimer les goulets d'étranglement dans le circuit de production. Il s'agit aussi ace niveau de définir un plan de production, de définir les gammes opératoires, d'ordonnancer les opérations, et enfin de gérer la répartition des taches durant tout le processus de fabrication [1].

2- Décomposition hiérarchique de la gestion de la production

On distingue trois niveaux hiérarchiques de la gestion de production (Figure 1.1) stratégique, tactique et opérationnel [13].

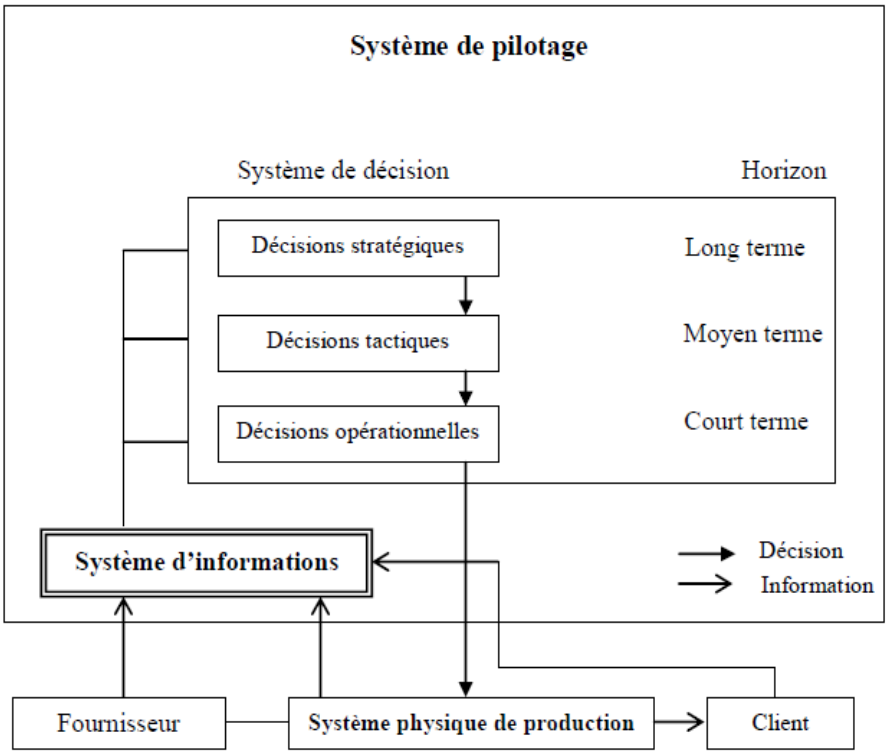


Figure 1.1 décomposition d'un système de production [13].

a- Le niveau stratégique

Ce niveau trace la politique à long terme de l'entreprise (à un horizon de plus de deux ans). Cette politique porte essentiellement sur la gestion des ressources durables. Afin que celle-ci soient en mesure d'assurer la pérennité de l'entreprise.

b- Le niveau tactique

Ce niveau relie les deux niveaux stratégique et opérationnel, il porte sur les décisions à moyen terme. Le but est d'assurer une production satisfaisante à la demande en minimisant les couts, tout en respectant le plan tracé par le niveau stratégique de l'entreprise.

c- Le niveau opérationnel

Ce niveau porte sur les décisions à court terme. Il s'agit d'une gestion quotidienne pour satisfaire les demandes en respectant les décisions tactiques.

Parmi les décisions opérationnelles : la gestion de la main d'œuvre, la gestion des stocks, la gestion des équipements.

3- Rôle de l'ordonnancement en gestion de production

L'ordonnancement d'atelier se fait dans un contexte particulier. Ce contexte est défini par la gestion de la production utilisée par l'entreprise. Il occupe une place particulière et joue un rôle privilégie dans la gestion informatisée des flux production au sein de l'entreprise, s'inscrivant dans des niveaux de décision à la fois tactique et opérationnel. C'est généralement le point de rencontre entre un système hiérarchise et informatise de production et le système de production lui-même.

En pratique, l'occurrence des évènements aléatoires endogènes : pannes de machines, grevés, … et exogènes : commandes imprévues et prioritaires, ... est fréquente et fortement probable. En effet, la mise a 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. La fonction ordonnancement se décompose en trois sous-fonctions ; la première sous fonction s'occupe de l'élaboration des ordres de fabrication (OF) c'est-a-dire transformation des informations du programme directeur de production (suggestion de fabrication) en ordres de fabrication. Dans la seconde, la préparation du programme d'atelier : ceci dit, la détermination en fonction des ordres de fabrication et de la disponibilité des ressources le calendrier prévisionnel de fabrication. Enfin dans la troisième, le lancement et le suivi des opérations de fabrication sont appliquées (voir Figure (1.2)) [1].

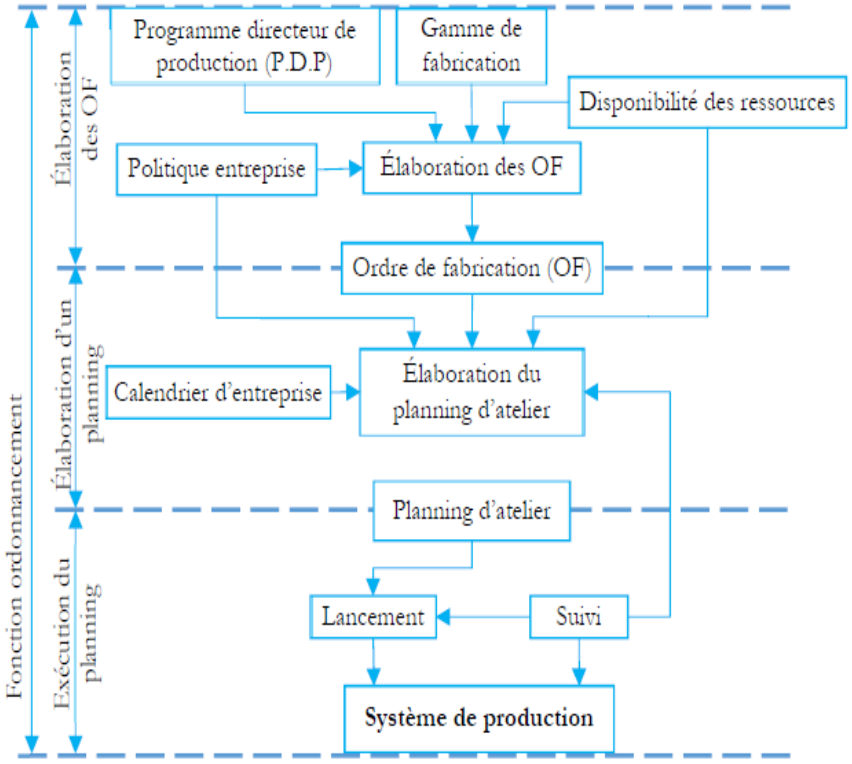


Figure 1.2 les sous-fonctions de l'ordonnancement dans l'atelier [1].

II- Les problèmes d'ordonnancement

1-Définitions et notions fondamentales

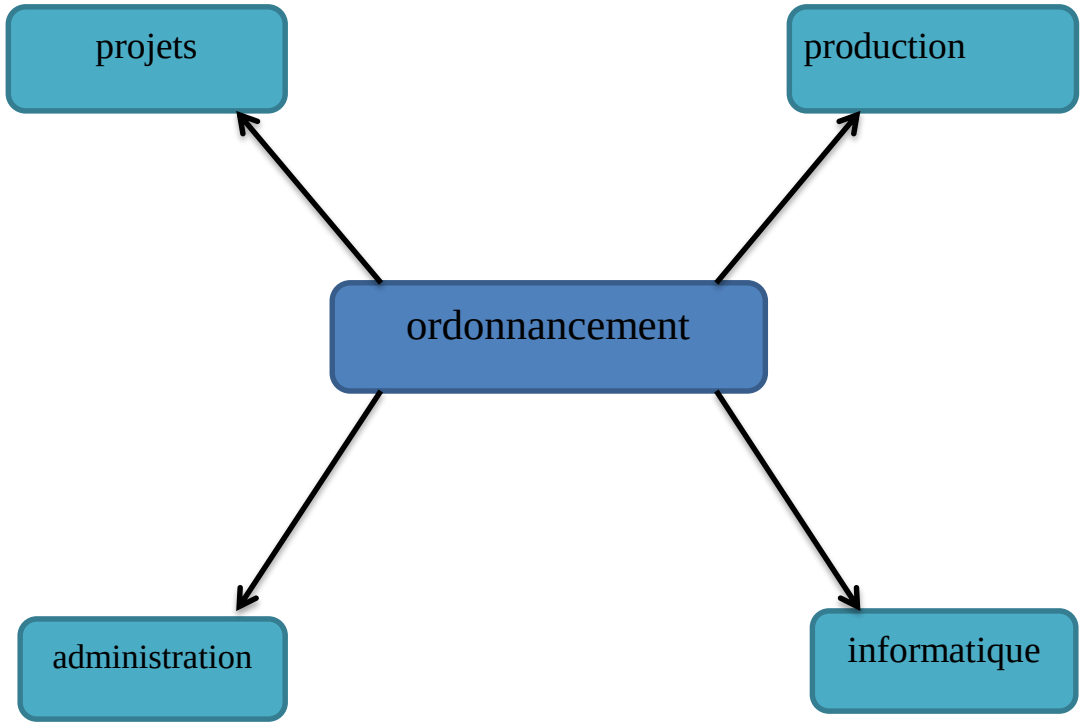
- Un problème d'ordonnancement peut être considéré comme un sous problème de planification dans lequel il s'agit de décider de l'exécution opérationnelle des tâches (jobs) planifiées, et ainsi d'établir leur planning d'exécution et leur allouer des ressources visant à satisfaire un ou plusieurs objectifs sous une ou plusieurs contraintes. En se basant sur les concepts de tâche, ressource, contrainte et objectif, l'ordonnancement peut également être déterminé comme.

- Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie l'informatique (les tâches sont les programmes ; les ressources sont les processeurs, la mémoire, la construction (suivi de projet), l'industrie (activités des ateliers en gestion de production et problèmes de logistique), l'administration (emplois du temps).Dans un problème d'ordonnancement interviennent deux notions fondamentales: les tâches et les ressources. Une ressource est un moyen, technique ou humain, dont la disponibilité limitée ou non est connu a priori. Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource [1].

-Ordonnancer un ensemble de tâches c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début. La théorie de l'ordonnancement traite de modèles mathématiques mais analyse également des situations réelles fortes complexes; aussi le développement de méthodes utiles ne peut-il être que le fruit de contacts entre la théorie et la pratique [10].

Les domaines concernés

Gestion de projets ateliers de produit



Gestion des ressources humaines exécution des programmes

Emplois du temps optimisation de code

Figure 1.3 les domaines de l'ordonnancement.

Dans un problème d'ordonnancement interviennent deux notions fondamentales : les ressources et les tâche.

- Une ressource est un moyen nécessaire humain ou technique utilisé pour exécuter une opération
- Une tache ou une opération qu'on note « i » est une entité élémentaire de travail localisée dans le temps par une date de début $r_i$et une date de fin c_i .dont la réalisation nécessite une durée p_i telle que $p_i = c_i - r_i$. et qui utilise des ressources k avec une intensité a_{ik}

  On distingue deux types de tâches :

- les taches préemptives dont l'exécution peut être divisée en plusieurs intervalles temporels.
- Les taches indivisibles qui sont exécutées en une seule fois et ne peuvent pas être interrompues avant qu'elles ne soient complétement achevées [10].

2- Notion d'objectifs et de critères

Les objectifs des entreprises ont été diversifiés et l'opération d'ordonnancement été de plus en plus multicritère. Après cette opération Les critères ont satisfais un ordonnancement qui ont été variés. D'une façon générale.

Les fonctions économiques ou critères d'optimalité les plus utilisées font intervenir la durée totale de l'ordonnancement, le délai d'exécution, les retards de l'ordonnancement et le coût des stocks d'encours. La durée totale de l'ordonnancement notée C_{max}est égale à la date d'achèvement de la tâche la plus tardive : C_{max}= max c_J. C'est la longueur de l'ordonnancement (schedule length ou makespan).

Le Flow time moyen $\bar{F} = \frac{1}{n}\sum_{j=1}^{n} F_j$ou le flow time moyen pondéré $\overline{F_w} = \sum_{j=1}^{n} W_j F_j / \sum_{j=1}^{n} W_j$

Le critère, flow time,$\sum_{i=1}^{n} W_i C_i$permet d'estimer le coût des stocks d'encours. En effet la tâche " i " est présente dans l'atelier entre les instants r_i et C_i , et donc les stocks dont elle a besoin doivent être disponibles entre ces deux dates; d'où le coût$\sum_{i=1}^{n} W_i(C_i - r_i)$est égale à une constante prés à $\sum_{i=1}^{n} W_i C_i$.

Dans beaucoup de problèmes, il faut respecter les délais, donc les dates au plus tard d_i ; on peut chercher à minimiser le plus grand retard T_{max}= max T_i, ou bien la somme des retards$\sum_{i=1}^{n} T_i$. ou encore la somme pondérée des tâches en retard $\sum_{i=1}^{n} W_i T_i$.

Le décalage maximum L_{max}= max { L_J}.

D'autres critères peuvent être utilisés

- Le retard moyen $T_{moy} = \frac{1}{n}\sum_{j=1}^{n} T_j$. Le retard moyen pondéré$T_w = \sum_{j=1}^{n} W_j T_j / \sum_{j=1}^{n} W_j$.
- Le nombre de tâches en retard $U = \sum_{j=1}^{n} U_j$.où U_j= 1 si C_j> d_j et 0 sinon.
- Le nombre de tâches en retard pondéré $U_w = \sum_{j=1}^{n} W_j U_j$ [6].

Dans ce travail, nous allons nous concentrer sur l'étude des objectifs liés au temps (C_{max} ,$\sum_{i=1}^{n} T_i$).

3-Les contraintes d'ordonnancement

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision. On distingue :

- des contraintes temporelles
  - les contraintes de temps alloué, issues généralement d'impératifs de gestion et relatives aux dates limites des tâches (délais de livraisons, disponibilité des approvisionnements) ou à la durée totale d'un projet ;
  - les contraintes de cohérence technologique, ou contraintes de gammes, qui décrivent des relations d'ordre entre les différentes tâches ;
- des contraintes de ressources
  - les contraintes d'utilisation de ressources qui expriment la nature et la quantité des moyens utilisés par les tâches, ainsi que les caractéristiques d'utilisation de ces moyens ;
  - les contraintes de disponibilité des ressources qui précisent la nature et la quantité des moyens disponibles au cours du temps. Toutes ces contraintes peuvent être formalisées sur la base des distances entre débuts de tâches ou potentiels [19].

4-Résolution d'un problème d'ordonnancement

Résoudre un problème d'ordonnancement, c'est choisir pour chaque tache une date de début, de telle sorte que les contraintes du problème soient respectées (la solution est donc dite admissible ou réalisable) et qu'un ou plusieurs critères donnés soient optimisés. La résolution d'un problème d'ordonnancement doit concilier deux objectifs :

- ✓ L'aspect statique consiste à générer un plan de réalisation des travaux sur la base des données prévisionnelles.
- ✓ L'aspect dynamique consiste à prendre des décisions en temps réel, compte tenu de l'état des ressources et l'avancement dans le temps des différentes taches.

III- Les problèmes d'ateliers

Dans un problème d'atelier, une pièce doit être usinée ou assemblée sur différentes machines. Chaque machine est une ressource disjonctive, c'est-à-dire qu'elle ne peut exécuter qu'une tâche à la fois, et les tâches sont liées exclusivement par des contraintes d'enchaînement. Plus précisément, les tâches sont regroupées en n entités appelées travaux ou lots. Chaque lot est constitué de m tâches à exécuter sur m machines distinctes, et dans le cas des problèmes d'atelier, une tâche est une opération, une ressource est une machine et chaque opération nécessite pour sa réalisation une machine. Dans le modèle de base de l'ordonnancement d'atelier, l'atelier est constitué de m machines, n travaux (jobs), disponibles à la date 0, doivent être réalisés, un travail i est constitué de n_i opérations, l'opération j du travail i est notée (i,j) avec $(i,1) < (i,2) < \ldots < (i,n_i)$ si le travail i possède une gamme ($A < B$ signifie A précède B). Une opération (i,j) utilise la machine $m_{i,j}$pendant toute sa durée $p_{i,j}$etne peut être interrompue.

Un atelier se détermine par le nombre de machines qu'il contient et par son type. Une classification peut exister selon le nombre des machines et l'ordre d'utilisation des machines, pour réaliser un travail (par exemple fabrication d'un produit qui dépend de la nature de l'atelier) .

Solon les caractéristiques des machines on peut distinguer plusieurs types des problèmes :

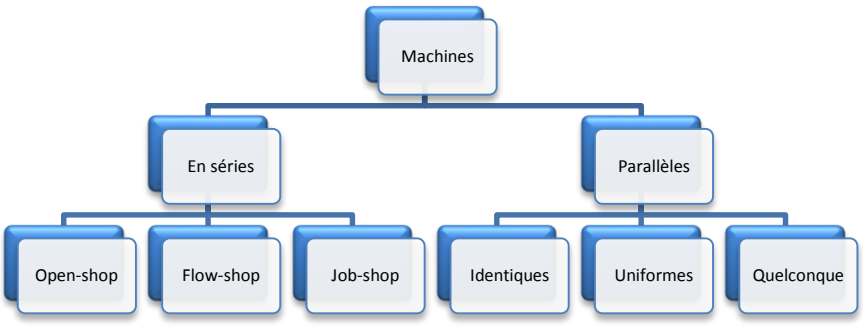


Figure1.4 types de machines.

1. Schémas de classification

Nous suivons les schémas de classification proposée par Graham et al, 1979.

Classification à trois champs **α, β, γ**:

- **α : environnement machine.**
  - α_1 **:** Prend des différents caractères selon le type de problème**.**
  - α_2 **:** Désigne le nombre de machine .Si α2 est entier positive, le nombre de machine supposé constant .si α2 est absent alors ce nombre est supposé arbitraire.
- **β : les caractéristiques des tâches.**
  - β_1 = 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.
  - $\beta_3 = r_j si$ les dates de début au plus tôt r_j (ou dates de disponibilité) des tâches ne sont pas forcément identiques, sinon ($j, r_j = 0$) β3 est absent.
  - β_4 = j si la tâche ou le job possède une date de fin nécessaire.
- **γ :** le (ou les) critère(s) à optimiser.

2- Les types de problème d'ateliers

2-1 machine unique :

Dans ce cas, l'ensemble des tâches à réaliser est fait par une seule machine. Les tâches alors sont composées d'une seule opération qui nécessite la même machine. L'une des situations intéressantes où on peut rencontrer ce genre de configurations est le cas où on est devant un système de production comprenant une machine goulot qui influence l'ensemble du processus. L'étude peut alors être restreinte à l'étude de cette machine [17].

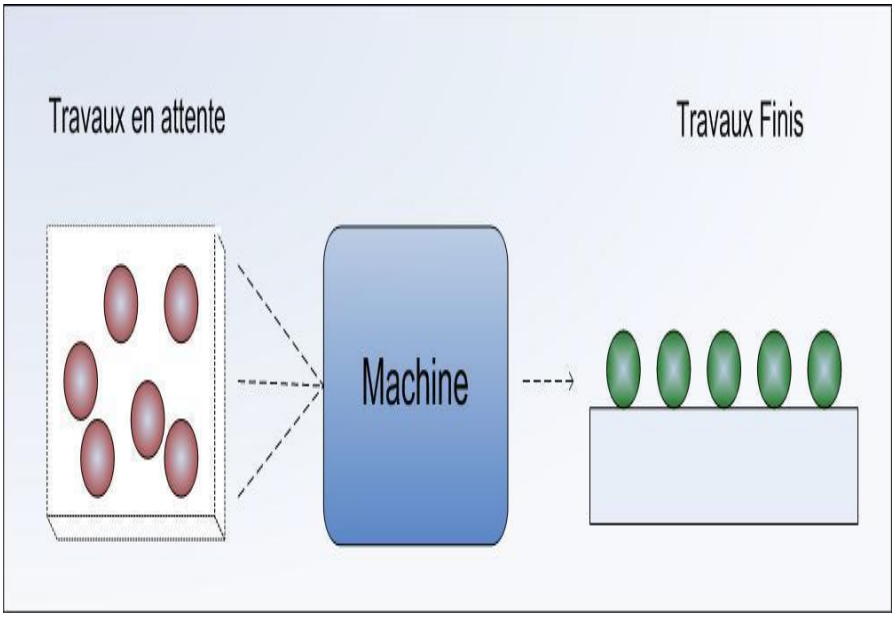


Figure 1.5 la représentation de la machine unique [17]**.**

2-2 Machines parallèles

Dans ce cas, on dispose d'un ensemble de machines identiques pour réaliser les travaux. Les travaux se composent d'une seule opération et un travail exige une seule machine. L'ordonnancement s'effectue en deux phases : la première phase consiste à affecter les travaux aux machines et la deuxième phase consiste à établir la séquence de réalisation sur chaque machine [17].

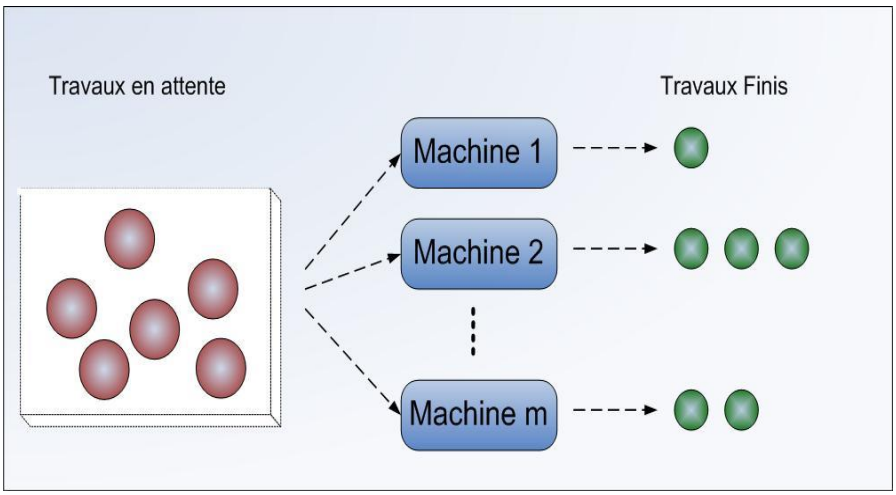


Figure 1.6 représentation des machines parallèles [17].

2-3 Problèmes de Flow shop

Définition **:** On appelle gamme d'un travail ou gamme opératoire, la succession des opérations qui le constituent, ou la succession des machines qui sont associées. Les contraintes et les critères du problème concernent les opérations, leurs localisations temporelles, et les moyens essentiels à leur réalisation.

Les problèmes de Flow shop **:** la gamme de fabrication, qui correspond à l'ordre d'exécution des opérations au sein des travaux, est linéaire, fixée à l'avance et identique pour tous les travaux, ce qui permet de numéroter les machines dans l'ordre d'exécution des opérations à traiter**.** Selon les types de produits élaborés, on distingue la production continue et la production discrète**.** La production continue est caractérisée par la fluidité de son processus et l'élimination du stockage. C'est le cas notamment dans les raffineries, les cimenteries, les papeteries... La production discrète de masse s'applique principalement aux produits de grande achèvement fabriqués à la chaîne (e.g automobile, la majorité du domaine du textile, machines-outils…).Parmi les caractéristiques d'un problème de cette catégorie :

- il existe au minimum n! différentes solutions où n est le nombre de travaux à réaliser. Notons que

n! = n*(n-1)*(n-2)….*1 ;

- une grande productivité mais une faible flexibilité .

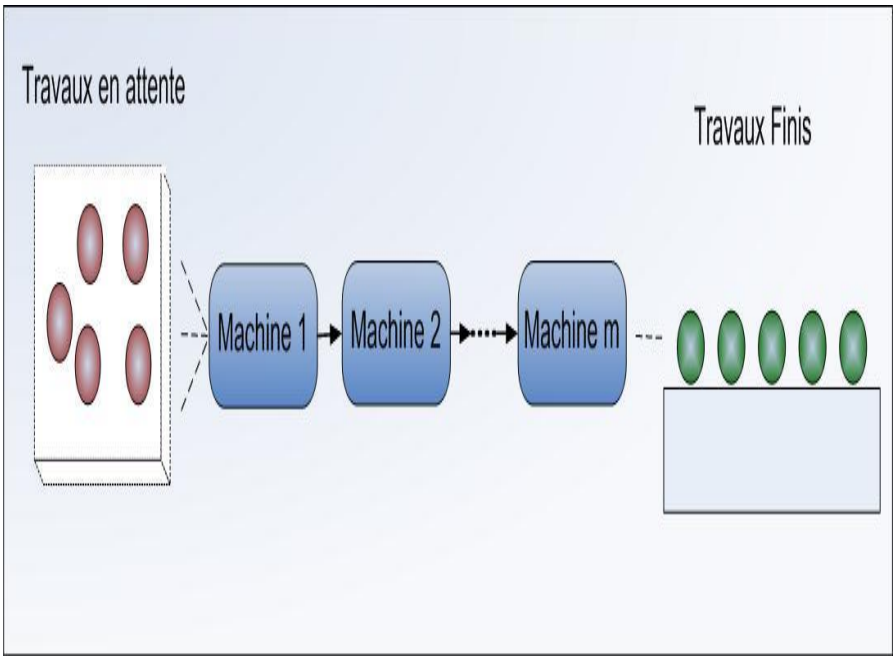


Figure 1.7 représentation ateliers à cheminement unique (flow-shop) [17].

2-4 Les problèmes job-shop

Les ateliers à cheminements multiples (ACM) sont des unités manufacturières traitant une variété de produits individuels dont la production requiert divers types de machines dans des séquences variées. L'une des caractéristiques d'un atelier à cheminement multiple est que la demande pour un produit particulier est généralement d'un volume petit ou moyen. Une autre caractéristique est la variabilité dans les opérations et un mix produit constamment changeant. Ainsi, il est nécessaire que le système soit de nature flexible. Dans un sens général, la flexibilité est la capacité d'un système de répondre aux variations dans l'environnement.

L'objectif le plus considéré dans le cas d'un atelier à cheminements multiples est le même que celui considéré pour un atelier à cheminement unique, à savoir trouver une séquence de tâches sur les machines qui minimise le temps total de production.

La figure suivante montre un exemple d'un atelier à cheminements multiples avec quatre travaux et six machines [17].

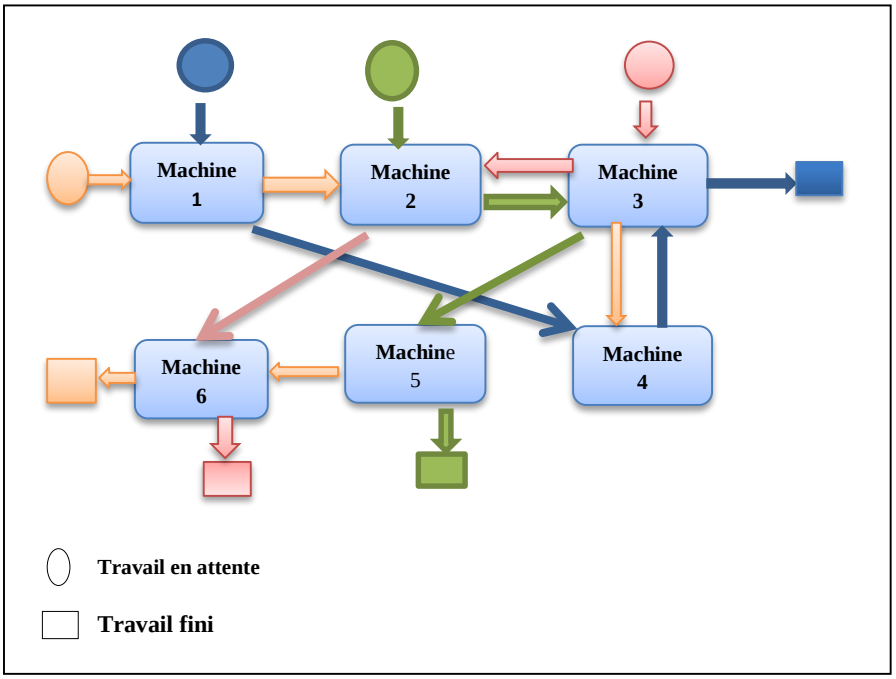


Figure1.8 représentation d'ateliers à cheminement multiple (job-shop) [17].

2-5 les problèmes open-shop :

C'est un modèle d'atelier moins contraint que le Flow shop et le job shop, car l'ordre des opérations n'est pas fixe à priori. Le problème d'ordonnancement comprend

d'une part à déterminer le cheminement de chaque travail et d'autre part à ordonnancer les travaux en tenant compte des gammes trouvées (Un problème de type open shop est un job shop dans lequel les contraintes de précédence sont relâchées. C'est-à dire, les opérations peuvent être effectuées dans n'importe quel ordre).

3- La notion de la flexibilité

- La **flexibilité** est une mode de gestion de la main d'œuvre qui permet aux entreprises d'ajuster rapidement la production et l'emploi (l'offre) aux fluctuations rapides des commandes des clients (demande), et il y a cinq types de flexibilité du travail :
- **la flexibilité externe quantitative** qui permet de faire fluctuer les effectifs de l'entreprise en fonction des besoins en ayant recours aux licenciements et aux contrats de travail de courte durée.
- **la flexibilité externe qualitative (ou externalisation)** qui « consiste à déplacer sur une autre entreprise le lien contractuel avec le travailleur » en aillant recours par exemple aux travailleurs intérimaires ou à l'externalisation d'un certain nombre d'activités annexes à la production (gardiennage, restauration, nettoyage…).
- **la flexibilité salariale** qui permet de faire varier à travers la rémunération des salariés, le coût de la masse salariale de l'entreprise. Elle « est conçue comme un moyen de répercuter sur les salaires les évolutions du chiffre d'affaire et de coûts de revient de l'entreprise en fonction des mouvements conjoncturels ».
- **la flexibilité interne quantitative** qui consiste à faire varier la quantité d'heures travaillées pour un effectif donné. Elle peut être réalisée par des modulations saisonnières à partir d'un contrat portant sur une durée annuelle, des temps partiels, des travails intermittents, des heures supplémentaires...
- **La flexibilité interne qualitative (ou flexibilité fonctionnelle)** qui « consiste, à quantité de travail donnée, à employer les travailleurs à des fonctions variables en fonction des besoins de la chaîne de production ou des fluctuations de la production ».

Afin d'être toujours plus réactives et productives, les entreprises ont cherché à augmenter la flexibilité de leurs systèmes de production. Pour atteindre ce but, il est possible de multiplier le nombre des machines qui peuvent réaliser une même opération. Ces machines, considérées comme identiques dans le cadre de ce

mémoire, sont regroupées en étage ou cellule. Et pour cela on peut distinguer autres types d'atelier :

3-1 Flow shop hybride

Le Flow Shop hybride est une généralisation du Flow Shop classique au cas où nombreuse machines sont disponibles sur un ou plusieurs étages pour exécuter les différentes tâches du Flow Shop. Ces problèmes présentent alors une difficulté supplémentaire par rapport aux problèmes sans flexibilité des ressources. En effet, la machine qui sera utilisée pour exécuter une opération n'est pas connue d'avance, mais doit être sélectionnée parmi un ensemble donné pour construire une solution au problème .

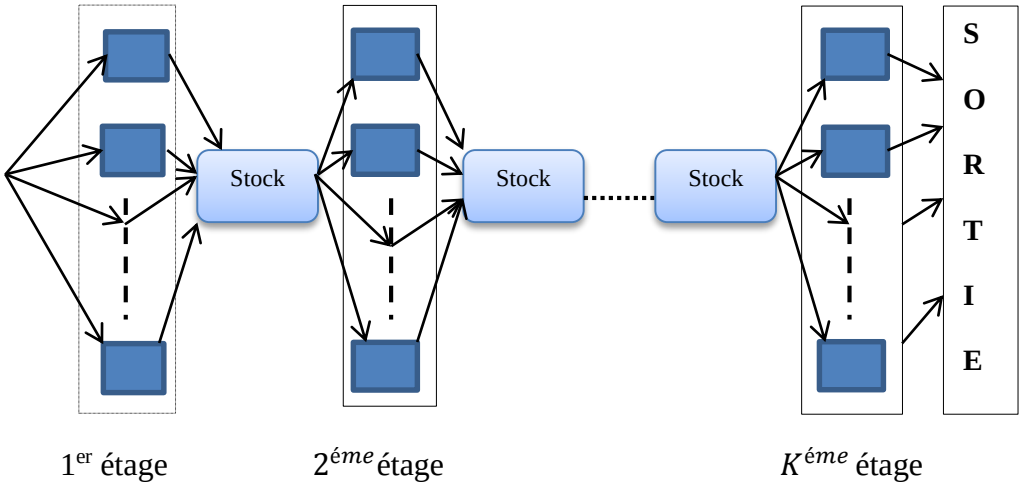


Figure1.9 représentation d'un flow show hybride à « k » étages.

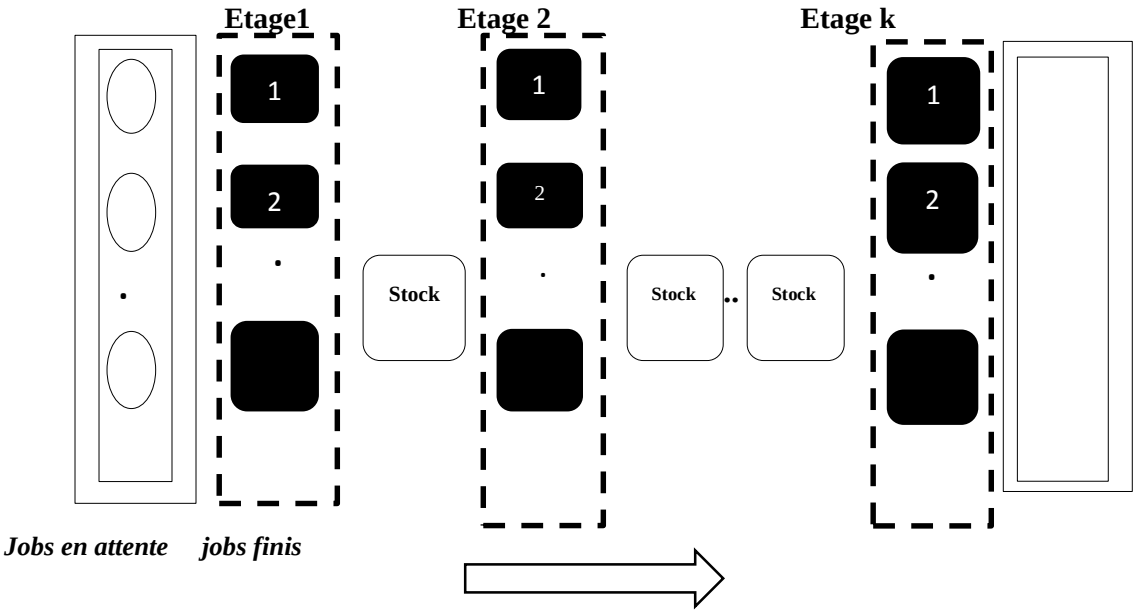
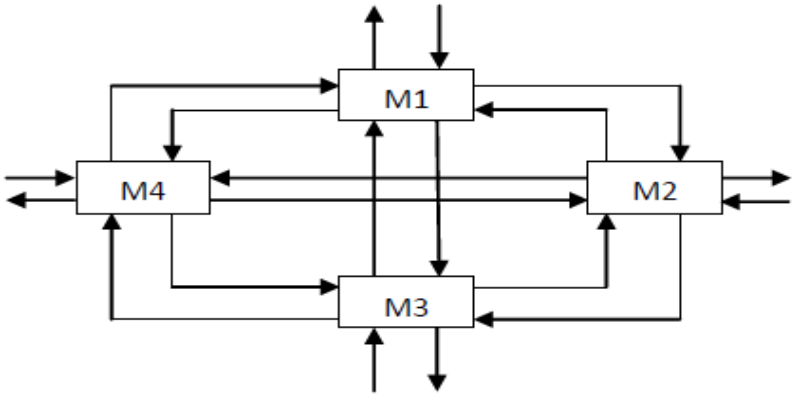


Figure 1.10 représentation d'un flow show hybride à « k » étages.

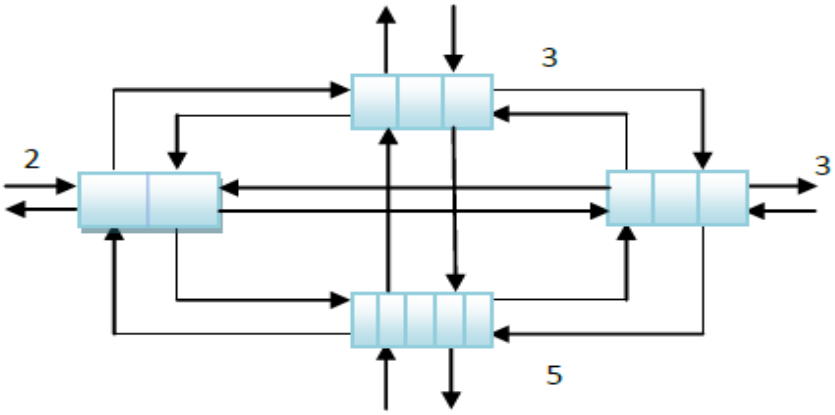
3-2. Job shop flexible

L'atelier Job Shop Flexible est une extension du problème classique décrit précédemment. La flexibilité est due aux ressources, c'est-à-dire, l'attribution d'un sous-ensemble de ressources (machines candidates) pour le traitement de chaque opération de telle sorte que le temps opératoire de chaque opération dépend de la machine candidate sélectionnée.

En effet, il existe plusieurs degrés de flexibilité : la flexibilité faible dans laquelle quelques opérations qui sont traitable dans quelques machines. Ensuite, dans la flexibilité moyenne et forte le nombre d'opérations traitables dans plusieurs machines devient de plus en plus important. En arrivant à la flexibilité externe (totale), n'importe quelle opération est traitable sur n'importe quelle machine [13].



(A) Job Shop simple avec 4 machines



(B) Job Shop hybride à 4 postes de travail

Figure 1.11 Job shop simple & hybride [13]

4- La complexité

A première vue, comment déterminer un algorithme efficace ? Pour un problème donné, chercher un algorithme efficace, veut dire trouver un algorithme où le temps nécessaire à son exécution ne soit pas trop important. Un problème est dit facile si on peut le résoudre facilement, c'est-à-dire s'il ne fait pas trop de temps pour arriver à la solution. Donc, s'il existe un algorithme efficace pour un problème donné, alors ce dernier est dit facile.

Un problème pour lequel on ne connaît pas d'algorithme efficace, est ce qu'il est facile ou difficile ?

De nombreux chercheurs se sont penchés sur ce genre de problèmes et ils ont développé une théorie appelée de la complexité. Nous n'allons pas détailler cette théorie, mais nous allons, quand même donner une idée globale du sujet en question.

4-1 complexités 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.

4-2 La complexité problématique

La complexité problématique est relative au problème à résoudre ainsi que la méthode de résolution adoptée pour élaborer la solution optimale par rapport au critère retenu.

- Un problème de décision comprend deux parties : une partie donnée du problème et un processus binaire ayant « oui »ou « non » comme réponse possible.

- Un problème de recherche est un problème constitué d'un ensemble de données dont chacun représente un ensemble de solutions. Donc, la résolution d'un problème de recherche consiste à trouver pour chaque ensemble de données D des solutions S associées. Un problème d'optimisation est un problème de recherche en associant à chaque solution une valeur qualitative. A chaque problème

d'optimisation, on peut associer un problème de décision (par exemple l'exclusion ou l'inclusion d'une solution dans les futures générations pour les AG), donc l'étude de la complexité du problème de décision peut donner des indications au problème d'optimisation associé.

La théorie de la complexité permet de classer les problèmes en deux classes *P* et *NP*. La classe *P* regroupe les problèmes qui peuvent être résolus par des algorithmes polynomiaux. Un algorithme est dit polynomial, lorsque son temps d'exécution est borné par *O(P(x))* ou *p* est un polynôme et *x* est la longueur d'entrée d'une instance du problème. Les algorithmes dont la complexité ne peut pas être bornée polynomialement sont qualifiés d'exponentiels et correspondent à la classe *NP*.

Un problème de décision est dit *NP-complet* s'il appartient à la classe *NP* et il résolu, au mieux, en un temps exponentiel.

Un problème d'optimisation est dit *NP-Difficile,* si le problème de décision associé est *NP-complet* [13].

Conclusion

Dans ce chapitre, nous avons définit l'ordonnancement, quelques notions de base d'ordonnancement tel que la notion de tâche, de ressource, les problèmes d'ordonnancement, ses contraintes, ses objectifs et ses schéma de classification, ainsi que les différents types de problèmes d'ordonnancement d'atelier. Dans le chapitre suivant nous allons apprendre au modelé mathématique de chaque type**.**

CHAPITRE 2

LES MODELES MATHEMATIQUES DES PROBLEMES D'ORDONNANCEMENT

Introduction

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. L'avantage de ce type de modélisation est d'obtenir des expressions mathématiques simples et robustes, qui peuvent être exploitées facilement en programmation.

Dans ce chapitre nous discuterons la minimisation de makespan C_{max} et la somme des retards $\sum T$ de problème M Machine et le problème Flow-shop.

1- 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. La minimisation du *makespan* s'accompagne généralement de contraintes qui peuvent être temporelles ou liées aux ressources. Les contraintes temporelles se divisent en deux catégories : des contraintes de temps alloué (impératif de gestion : délai de livraison, disponibilité, achèvement) et des contraintes d'antériorité (cohérence technologique : gammes de fabrication, inégalité de potentiels : précédence). Les contraintes liées aux ressources peuvent être des contraintes disjonctives (une tâche i doit s'exécuter avant ou après une tâche}) ou des contraintes cumulatives (respect des capacités des ressources).

On peut aussi considérer d'autres critères de performance, tels que le temps moyen d'achèvement des jobs, le temps total de traitement, le temps de retard total, le temps d'attente des jobs, le taux d'occupation de machines, le nombre de jobs en retard, le temps de séjour d'un job dans le système avant sa réalisation, etc. [7].

2- Le problème de M-Machine identique en parallèle

Après les études des modèles mathématiques, les données du problème et le modèle qui s'expriment de la manière suivante :

- n : le nombre de tâches.
- m : le nombre de machines.
- j : l'indice de la tâche, où $j = 1, \ldots . n$.
- k : l'indice de la machine, où $k = 1, \ldots . m$.
- r : la position de la tâche dans une machine, où $r = 1, \ldots . n_k$.
- r_{jk}: date de début.
- d_{jk} : date de fin la tâche j.
- s_{ij}: temps de préparation de la tâche j effectuée 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.
- n_k: nombre de tâches affectées à la machine k.

Minimiser $(C_{max}, \sum T)$ (1)

Sous contraintes :

$\sum_{j=1}^{n} X_{jkr}$ $k = 1,2, \ldots . , m\ r = 1,2, \ldots , n_k$ (2)

$\sum_{k=1}^{m} \sum_{r=1}^{n_k} X_{jkr}$ $j = 1,2, \ldots . , n$ (3)

$P_{[kr]} = \sum_{j=1}^{n} X_{jkr} P_{jk}$ $k = 1,2, \ldots . , m\ r = 1.2, \ldots . n_k$ (4)

$S_{[kr]} = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{jkr} Y_{ij} s_{ij}$ $k = 1,2, \ldots . , m\ r = 1,2, \ldots . n_k$ (5)

$r_{[kr]} = \sum_{j=1}^{n} X_{jkr} r_{jk}$ $k = 1,2, \ldots . , m\ r = 1,2, \ldots . n_k$ (6)

$d_{[kr]} = \sum_{j=1}^{n} X_{jkr} d_{jk}$ $k = 1,2, \ldots . , m\ r = 1,2, \ldots . n_k$ (7)

$$C_{[kr]} = \max\left(C_{[k,r-1]} + s_{[kr]}, r_{[kr]}\right) + P_{[kr]} \qquad k = 1,2,..,m\ r = 1,2,\ldots.n_k \qquad (8)$$

$$T_{[kr]} = \max(C_{[kr]} - d_{[kr]}, 0) \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n_k \qquad (9)$$

$$C_{max} = max_{k=1}^{m} max_{r=1}^{n_k} C_{[kr]} \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n_k \qquad (10)$$

$$\sum T_{jk} = \sum_{k=1}^{m} \sum_{r=1}^{n_k} T_{[kr]} \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n_k \qquad (11)$$

$$X_{jkr} = 0\ or\ 1 \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n_k\ j = 1,2,\ldots.,n \qquad (12)$$

$$Y_{ij} = 0\ or\ 1 \qquad i = 1,2,\ldots.,n\ j = 1,2,\ldots.,n \qquad (13)$$

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 de singularité des tâches sur la machine *k* et à la position *r*. Elles garantissent qu'il y a seulement une tâche sur la machine *k* et à la position *r*, et que chaque tâche est déplacée seulement une fois sur ces machines.

La contrainte (4) calcule la durée d'opération de la tâche qui est en position *r* sur la machine *k*. La contrainte (5) définit le temps de préparation de la tâche qui est en position *r* sur la machine *k*.

Les contraintes (6) - (9) 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 (10) et (11) représentent le calcul du makespan et de la somme des retards. La variable binaire X_{jkr} est égale à 1, si la tâche *j* est ordonnée en position *r* sur la machine *k* et 0 sinon. 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 Y_{ij} est égale à 1, car aucune tâche ne précède *j* [5].

3- Le problème Flow-shop

- n : le nombre de tâches.
- m : le nombre de machines.
- j : l'indice de la tâche, où $j = 1, \ldots . n$.
- k : l'indice de la machine, où $k = 1, \ldots . m$.
- r : la position de la tâche dans une machine, où $r = 1, \ldots . n_k$.
- r_{jk}: date de début.
- d_{jk} : date de fin la tâche j.
- s_{ij}: temps de préparation de la tâche j effectuée 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 ($C_{max}, \sum T$) (1)

Sous contraintes :

$\sum_{j=1}^{n} X_{jkr}$ $\_ = 1,2, \ldots . , m \; r = 1,2, \ldots , n$ (2)

$\sum_{k=1}^{m} \sum_{r=1}^{n} X_{jkr}$ $j = 1,2, \ldots . . , n$ (3)

$R_{ki} = R_{kj}$ $k = 1,2, \ldots , m \; i = 1,2, \ldots , m-1 \; j = 2, \ldots , m$ (4)

$P_{[kr]} = \sum_{j=1}^{n} X_{jkr} P_{jk}$ $k = 1,2, \ldots . , m \; r = 1.2, \ldots . n$ (5)

$S_{[kr]} = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{jkr} Y_{ij} s_{ij}$ $k = 1,2, \ldots . . , m \; r = 1,2, \ldots .$ (6)

$r_{[kr]} = \sum_{j=1}^{n} X_{jkr} r_{jk}$ $k = 1,2, \ldots . . , m \; r = 1,2, \ldots . n$ (7)

$d_{[kr]} = \sum_{j=1}^{n} X_{jkr} d_{jk}$ $k = 1,2, \ldots . . , m \; r = 1,2, \ldots . n$ (8)

$$C_{[kr]} = \max\left(C_{[k,r-1]} + s_{[kr]}, r_{[kr]}\right) + P_{[kr]} \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n \qquad (9)$$

$$T_{[kr]} = \max(C_{[kr]} - d_{[kr]}, 0) \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n \qquad (10)$$

$$C_{max} = max_{k=1}^{m} max_{r=1}^{n}\, C_{[kr]} \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n \qquad (11)$$

$$\sum T_{jk} = \sum_{k=1}^{m} \sum_{r=1}^{n} T_{[kr]} \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n \qquad (12)$$

$$X_{jkr} = 0\ or\ 1 \qquad k = 1,2,\ldots..,m\ r = 1,2,\ldots.n\ j = 1,2,\ldots.,n \qquad (13)$$

$$Y_{ij} = 0\ or\ 1 \qquad i = 1,2,\ldots.,n\ j = 1,2,\ldots.,n \qquad (14)$$

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 X_{jkr} est égale à 1, si la tâche *j* est ordonnée en position *r* sur la machine *k* et 0 sinon. 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 Y_{ij} est égale à 1, car aucune tâche ne précède *j* [5].

Conclusion

Dans ce chapitre nous avons donné une définition de makespan et les modèles mathématiques de problème M-machine identique en parallèle et de problème Flow-shop. Nous discuterons dans chapitre 3 à les méthodes de résolution.

CHAPITRE 3

LES METHODES DE RESOLUTION DES PROBLEMES D'ORDONNANCEMENT

Introduction

La résolution de problème est le processus d'identification puis la mise en œuvre d'une solution de problème.

Dans ce chapitre nous représentons les méthodes de résolution d'un problème d'ordonnancement qui distingue deux types : le premier type est exacte et la deuxième type est approché, chaque type contient des méthodes, qui sont détaillé comme suit :

1- Les algorithmes exacts

Une méthode exacte permet de trouver une solution optimale à un problème donné. Toutefois, ces méthodes peuvent devenir rapidement couteuses en temps d'exécution, notamment pour les problèmes NP-difficiles. En effet, le temps de traitement et la complexité du problème sont généralement liés (plus c'est complexe, plus le temps d'exécution sera important). Ci-dessous, quelques méthodes exactes parmi les plus connues [2] :

– Procédure par Séparation et Évaluation.
– Programmation dynamique.
– Programmation par contraintes.
– Programmation linéaire.

1-1 La méthode de "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.

L'algorithme branch and bound accomplit une recherche complète de l'espace des solutions d'un problème donné, pour découvrir la meilleure solution .La démarche de l'algorithme Branch and Bound consiste à :

- Diviser l'espace de recherche en sous espaces,
- Chercher une borne minimale en terme de fonction objectif

associée à chaque sous espace de recherche,

- Eliminer les mauvais sous-espaces,
- Reproduire les étapes précédentes jusqu'à l'obtention de l'optimum global.

1-2 La programmation dynamique

Introduite par Bellman dans les années 50, La **«** programmation dynamique **»** est un paradigme de programmation, c'est-à-dire une façon particulière d'appréhender un problème algorithmique donné.

C'est une méthode utile pour obtenir une solution exacte à un problème algorithmique, là où une solution « classique » se trouve être trop complexe, c'est-à-dire trop peu efficace. On parle alors d'optimisation combinatoire [14].

Cette méthode décompose un problème de dimension n en n problèmes de dimension. Le système est alors constitué de n étapes que l'on résout séquentiellement, le passage d'une étape à une autre se faisant à partir des lois d'évolution du système et d'une décision. Le principe d'optimalité de Bellman est basé sur l'existence d'une équation récursive permettant de décrire la valeur optimale du critère à une étape en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille k -1 et porte le nombre de sous-ensembles considérés à *2n* (où n est le nombre d'éléments considérés dans le problème) [8].

1-3 La programmation par contraintes

(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) [18].

1-4 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- Les algorithmes approchés

L'objectif d'une méthode approchée est de obtenir une solution exécutable, prenant en considération la fonction objectif mais sans garantie d'optimalité. Son utilisation offre de multiples avantages par rapport à une méthode exacte, citons

- Elles sont plus simples et plus rapides à mettre en œuvre quand la qualité de la solution n'est pas trop importante.
- Elles sont plus souples dans la résolution des problèmes réels. En pratique, il arrive souvent que l'on doive prendre en considération de nouvelles contraintes qu'on ne pouvait pas formuler dès le départ. Ceci peut être fatal pour une méthode exacte si les nouvelles contraintes changent les caractéristiques sur lesquelles s'autorisait la méthode.
- Elles fournissent des solutions et des bornes qui peuvent être utiles dans la conception de méthodes exactes. Pour les problèmes d'ordonnancement, si on cherche une permutation optimale des tâches, l'espace des solutions admissibles comporte n! Permutations. Un exemple de voisinage d'une solution est défini par toutes les permutations obtenues en échangeant deux tâches consécutives de la permutation.

- Les méthodes approchées diffèrent aussi, les unes des autres, par le type de compromis qualité / complexité qu'elles offrent. Nous en distinguons deux classes : les heuristiques et les métaheuristiques.[3]

2-1 Les heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable. Parmi ces stratégies, nous distinguons :

-FIFO (First In First Out) où la première tâche arrivée est la première à être ordonnancée,

-SPT (Shortest Processing Time) où la tâche ayant le temps opératoire le plus court est traitée en premier,

-LPT (Longest Processing Time) où la tâche ayant le temps opératoire le plus important est traitée en premier,

-EDD (Earliest Due Date) où la tâche ayant la date due la plus petite est la plus prioritaire [13].

2-2 Les métaheuristiques

- Un algorithme de résolution métaheuristique est un algorithme heuristique "générique" qu'il faut ajuster à chaque problème.

- Une méta-heuristique est une heuristique généraliste, ils peuvent être appliqués à de nombreux problèmes d'optimisation.

- Classification habituelle des métaheuristiques : en fonction du nombre de solution qu'ils traitent

  - métaheuristiques à solution unique.
  - métaheuristiques à population de solutions.

2-2-1 métaheuristiques à solution unique

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [11].

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.

2-2-1-1 les méthodes de descentes

La méthode de descente décrite de manière générique dans l'algorithme 1 est un exemple de méthode de recherche locale. Une telle méthode progresse au travers de X en choisissant à chaque étape la meilleure solution voisine de la solution courante. Ce procédé est répété aussi longtemps que la valeur de la fonction objective diminue. La recherche s'interrompt dès lors qu'un minimum local de f est atteint.

Historiquement, les méthodes de descente ont toujours compté parmi les méthodes heuristiques les plus populaires pour traiter les problèmes d'optimisation combinatoire.

Toutefois elles comportent deux obstacles majeurs qui limitent considérablement leur efficacité:

- suivant la taille et la structure du voisinage $N(s)$ considéré, la recherche de la meilleure solution voisine est un problème qui peut être aussi difficile que le problème (P) initial;

- une méthode de descente est incapable de progresser au-delà du premier minimum local rencontré. Or les problèmes d'optimisation combinatoire comportent typiquement de nombreux optima locaux pour lesquels la valeur de la fonction objectif peut être fort éloignée de la valeur optimale [12].

Initialisation
choisir une solution admissible initiale s∈X ;
poser s*:=s ;
Processus itératif
tant que le critère d'arrêt n'est pas satisfait faire
générer N(s) ;
déterminer s'∈N(s) telle que
f(s')= min f(s") ;
s" N(s)
s := s' ;
si f(s) < f(s*) alors s*:=s ;
sinon le critère d'arrêt est satisfait

Algorithme 3.1 La méthode de descente

2-2-1-2 le recuit simulé

Est une méthode empirique (métaheuristique) destinée à résoudre au mieux les problèmes dits d'optimisation. Cette méthode est une technique de recherche locale inspirée du processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lents et de réchauffage ou de recuit qui tendent à minimiser l'énergie phisique. Le recuit simulé compter sur l'algorithme de Metropolis-Hastings, qui permettre la description du développement du système thermodynamique. Comme mesuré par le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système.

L'algorithme du recuit simulé commence à partir d'une solution donnée. Et la modifier à plusieurs reprises jusqu'au refroidissement du système. Les solutions trouvées peuvent améliorer le critère que l'on cherche à optimiser, on dit que nous avons

réduit l'énergie du système, car il peut être décomposé. Si on accepte une solution qui améliore le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. Contrairement autres méthodes de recherche locale, le recuit simulé peut accepter des solutions dont la qualité est moins bonne en fonction de la dégradation de la solution considérée.

```
Commencer avec une solution initiale : solution courante
Température := T0(*Température initiale*)
Tant que la condition d'arrêt n'est pas remplis faire
    Pour i de 1 à N faire
        Calculer nouvelle solution (perturber la solution)
        Δ = cout( nouvelle solution) – cout (perturber solution)
        Si Δ <0 alors
            Conserver nouvelle solution
        Sinon
            Si random[0,1]<e^[−Δ/Température] alors
                Conserver nouvelle solution
            Fin si
        Fin si
    Fin pour
        Température =á*Température (*nouveau palier de Température *)
Fin Tan que
```

Algorithme 3.2 le recuit de simulé

2-2-1-3 La méthode tabou

La méthode Tabou est une méthode de recherche proposée par Fred Glover dans les années 1980 et est devenue très conventionnelle dans l'optimisation globale. Il diffère des méthodes de recherche locales simples en recourant à un historique des solutions qui ont été visitées, afin de rendre la recherche "aveugle" un peu moins. Et pour éviter

de tomber périodiquement au minimum local, certains les solutions sont bloquées, sont « tabou ».

Contrairement à recuit simulé qui génère aléatoirement une solution proche l'une de l'autre $s' \in N(s)$ à chaque itération, Tabou examine un échantillonnage de solutions de $N(s)$ et retient la meilleure s' même si $f(s') > f(s)$. La recherche Tabou ne s'arrête donc pas au premier optimum trouvé.

Alors le risque est de revenir à s immédiatement. Puisque s est meilleure que s'.Pour éviter de tourner ainsi en rond, on crée une liste T qui mémorise les dernières solutions visitées et qui interdit tout déplacement vers une solution de cette liste. Cette liste T est appelée liste Tabou.

2-2-2 Les métaheuristiques à population de solutions

Les méthodes d'optimisation à population de solutions améliorent, Avec répétition continue, un ensemble de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

2-2-2-1 Les algorithmes génétiques

Un AG est une méthode d'optimisation dans laquelle un groupe appelé population de solution potentielles, appelées individus, est progressivement mis à jour à travers le mécanisme de sélection et le processus génétiques : Le croisement et La mutation.

Déroulement :

Dans les algorithmes génétiques, on essaye de simuler le processus d'évolution d'une population. Au début on génère une population de solutions d'une manière aléatoire, cette population forme la population initiale. Le degré d'adaptation de chaque individu à l'environnement (exprimé par la valeur de la fonction coût dite fonction fitness) est calculé.

Après, des couples des individus $P1$ et $P2$ (appelés parents) sont sélectionnés en fonction de leurs adaptations ; ensuite un opérateur de croisement est appliqué sur $P1$ et $P2$ avec une probabilité Pc et génère des couples $C1$ et $C2$ (dites enfants). D'autres individus P sont sélectionnés en fonction de leur adaptation ; un opérateur de mutation est appliqué sur P avec une probabilité Pm (Pm est généralement très inférieur à Pc) et génère des individus mutés $P0$.

Le niveau d'adaptation des enfants (C1, C2) et des individus mutés P0 sont ensuite évalués avant de les insérer dans la nouvelle population ; l'insertion est faite par le choix

de N individus parmi la population des parents et la population des enfants. En itérant ce processus jusqu'à un critère d'arrêt est atteint. Un critère d'arrêt peut être un nombre fixe d'itérations ou lorsque la population n'évolue d'une manière plus ou moins suffisamment rapide. La figure (1) montre ce déroulement [4].

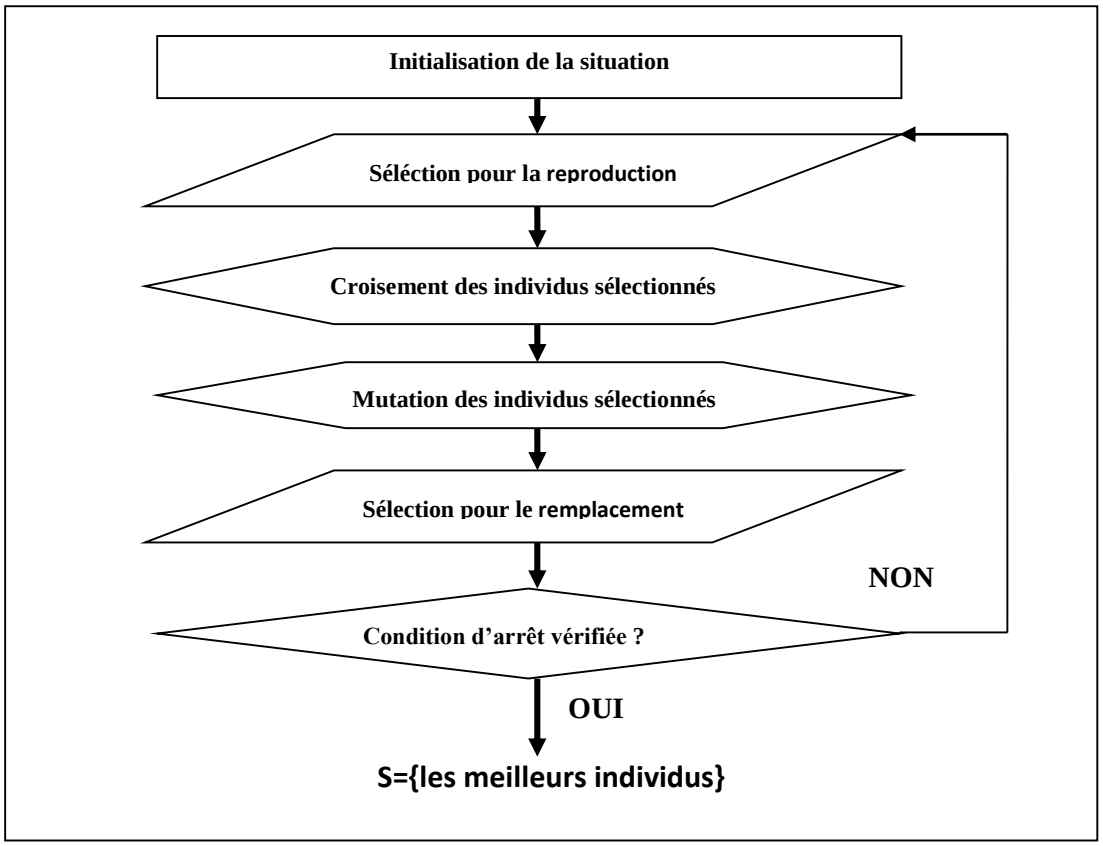


Figure 3.1 principe de fonctionnement d'un algorithme.

Le principe de l'algorithme génétique :

L'algorithme génétique repose sur une boucle qui relie les étapes de sélection et les étapes de croisement. Tout d'abord, à partir d'un groupe d'individus α, nous identifions ceux autorisés à se reproduire.

Ensuite, nous les croisons, afin d'obtenir un groupe d'enfants, qui peut aléatoirement transformer certains gènes.

La performance des enfants est évaluée, grâce à la fonction de fitness en forme, basée sur, le total des enfants + enfants, et les individus autorisés à survivre, de sorte que nous pouvons commencer à partir d'un nouveau groupe de α individus.

La boucle est terminée, et nous commençons l'étape de sélection pour la reproduction, la mutation de stade, et ainsi de suite.

Comme pour les métaheuristiques vues précédemment, Le critère d'arrêt permet à la boucle de partir, par exemple un certain nombre de répétitions sans amélioration significative de la performance des individus

Codage d'un algorithme génétique

Pour les algorithmes génétiques, un des facteurs les plus importants, si ce n'est le plus important, est la façon dont sont codées les solutions (ce que l'on a nommé ici les chromosomes), c'est-à-dire les structures de données qui coderont les gènes [16].

Parmi les techniques les plus couramment utilisées pour coder le personnel, on distingue :

Le codage en binaire

Dans ce type de codage, pour un individu on code ses variables et on les concatène par exemple, la chaîne binaire 1000| 0110| 1101, correspond à un individu défini par 3 variables (8, 6, 13) en codage binaire naturel sur 4 bits chacune.

C'est le codage le plus utilisé pour plusieurs raisons : pour des raisons historiques, ce codage a été utilisé par J. Holland et ses étudiants ; plusieurs résultats théoriques sont basés sur ce codage, et il est facile de mettre en place les opérateurs génétiques avec ce codage. Cependant si la longueur de la chaîne augmente la performance de l'algorithme diminuera [4].

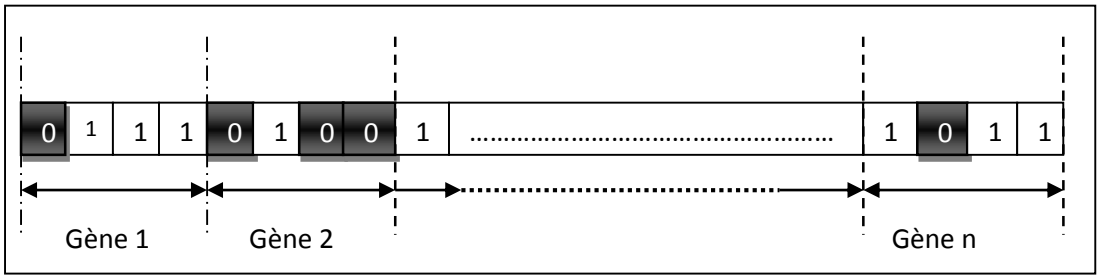


Figure 3.2 Structure d'un chromosome en codage binaire.

Le codage réel

Cela peut être particulièrement utile dans le cas où l'on cherche au maximum une fonction réelle.

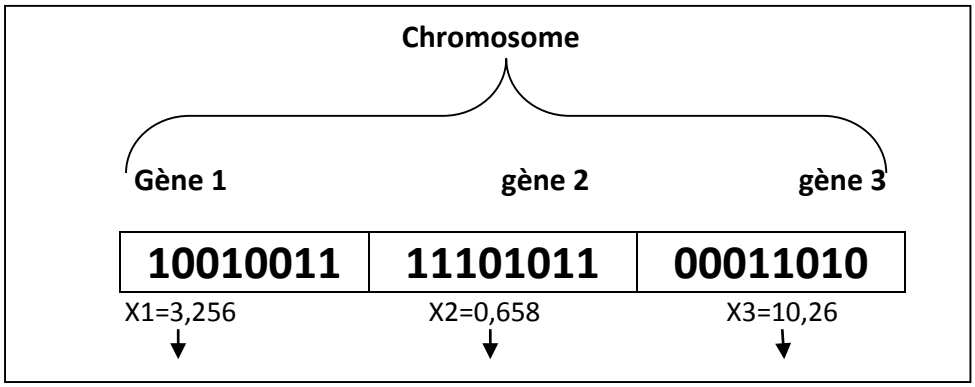


Figure 3.3 illustration schématique du codage des variables réelles

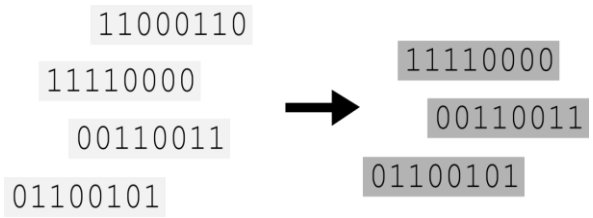
- Les opérateurs génétiques : Cette opération symbolise dans l'algorithme génétique la reproduction.

La 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.

Voici un exemple avec des individus en représentation binaire une fois la sélection faite:

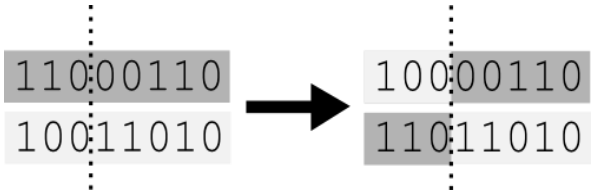


La croissement : Le croisement, ou enjambement, est le résultat obtenu lorsque les chromosomes partagent leurs propriétés.

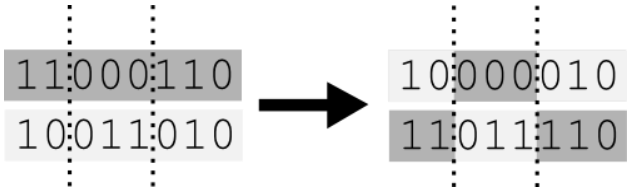
Cela permet une confusion génétique parmi la population et l'application du principe de l'héritage à la théorie de Darwin.

Il existe deux méthodes de croissement : simple ou double enjambement.

- Le simple enjambement : consiste à fusionner les particularités de deux individus à partir d'un pivot, afin d'obtenir un ou deux enfants



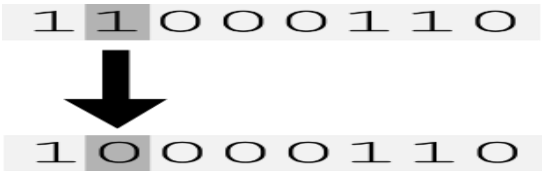
- Le double enjambement est basé sur le même principe sauf qu'il y a deux pivots.



On réalise ensuite une mutation sur les enfants obtenues lors du croisement..

La Mutation : est le changement dans le gène est fait dans le chromosome en fonction du facteur de mutation. Ce facteur est la probabilité d'une mutation sur l'individu. Cet opérateur applique le principe de variation de la théorie de Darwin et permet, dans le même contexte, d'éviter une convergence prématurée de l'algorithme vers un maximum local.

Voici un exemple de mutation sur un individu ayant un seul chromosome :



Avec ces trois opérateurs d'évolution, nous pouvons appliquer les algorithmes génétiques.

Les avantages des algorithmes génétiques :

Les algorithmes génétiques ont plusieurs avantages citons:

- Ils sont adaptables à plusieurs types de problèmes.
- Puissants.
- Facile à mettre en œuvre.
- Hybridation facile.
- C'est facile à mettre en parallèle.

Les inconvénients des algorithmes génétiques

Parmi les inconvénients des algorithmes génétiques, nous trouvons

- Il n'y a aucune garantie de rapprochement.
- Le temps de calcul est important (si la taille de la population est grande).

2-2-2-2 L'optimisation par colonie de fourmis

L'optimisation par colonies de fourmis ou Ant Colony Optimization (ACO) est une méthode d'optimisation basée sur la manipulation d'un ensemble de solutions, cette méthode représente toute une classe des métaheuristiques qui reposent sur la notion de l'intelligence collective. La solution finale est plus complexe que celle d'un composant simple. Plusieurs mots apparaissent dans ce domaine : l'auto-organisation, émergence et autres [4].

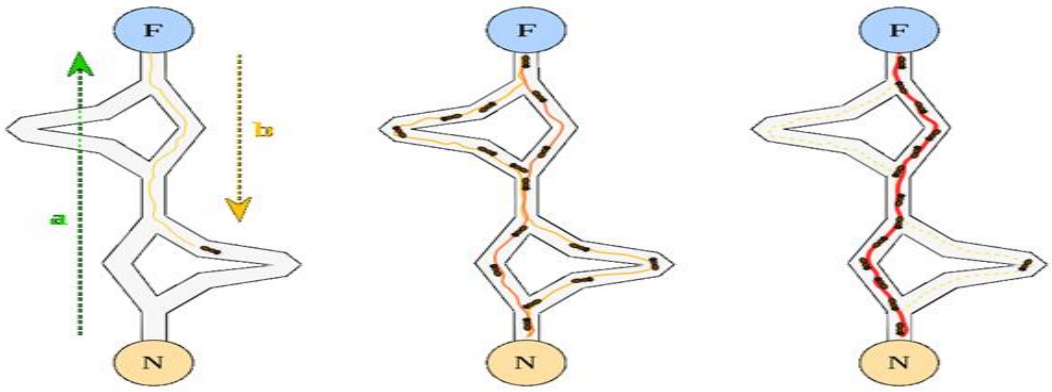


Figure 3.4 l'influence de l'expérience sur le choix des fourmis [15].

1) 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).

2) les fourmis empruntent indifféremment les quatre chemins possibles, mais le renforcement de la piste rend plus attractif le chemin le plus court.

3) les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromones.

Description et algorithme

Le premier algorithme de colonies de fourmis proposé est appelé le *Ant system* (système fourmi). Il vise notamment à résoudre le problème du voyageur de commerce, où le but est de trouver le plus court chemin permettant de relier un ensemble de villes.

L'algorithme général est relativement simple, et repose sur un ensemble de fourmis, chacune parcourant un trajet parmi ceux possibles. À chaque étape, la fourmi choisit de passer d'une ville à une autre en fonction de quelques règles :

1. elle ne peut visiter qu'une fois chaque ville.
2. plus une ville est loin, moins elle a de chance d'être choisie (c'est la « visibilité »).
3. plus l'intensité de la piste de phéromone disposée sur l'arête entre deux villes est grande, plus le trajet aura de chance d'être choisi.
4. une fois son trajet terminé, la fourmi dépose, sur l'ensemble des arêtes parcourues, plus de phéromones si le trajet est court.
5. les pistes de phéromones s'évaporent à chaque itération [15].

L'algorithme de la colonie de fourmis a d'abord été proposé pour résoudre le problème des navetteurs commerciaux .Il est basé sur trois étapes de base:

- Construis le chemin de chaque fourmi.
- Distribuez des phéromones sur chaque chemin de fourmis.
- Évaporation des voies de phéromone

Début
Initialiser une population de m fourmis ;
Evaluer les m fourmis ;
Tant que la condition d'arrêt n'est pas satisfaite **faire**
Pour i=1 à m **faire**
Construire le trajet de la fourmi i;
Déposer des phéromones sur le trajet de la fourmi i;
Fin pour
Evaluer les m fourmis;
Evaporer les pistes de phéromones;
Fin Tant que
Retourner la ou les meilleures solutions ;
Fin

Algorithme 3.3 L'algorithme de colonies de fourmis pour le TSP

L'algorithme 3 représente le schéma général de l'algorithme de colonies de fourmis pour le problème du voyageur de commerce (TSP: Traveling Salesman Problem). Il est à noter qu'à part le problème du voyageur de commerce, l'algorithme de colonies de fourmis a été appliqué avec succès sur d'autres problèmes d'optimisation comme: les problèmes des tournées de véhicules, le problème d'affectation quadratique…etc. [9].

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 ($\varepsilon -$ appro.)**
Aspect	**Déterministe**	**Stochastique (guidée)**
Complexité	**O(2^n) (souvent)**	**O(n^k)**
Handicap	**Explosion combinatoire**	**Convergence précoce**
Adaptée aux POC	**De petites tailles**	**NP-difficiles de grandes tailles**

Tab 3.1 La déférence entre méthode exacte et méthode approchée.

Conclusion

Dans ce chapitre on a étudié les différentes méthodes de résolution des problèmes d'ordonnancement. On distingue deux algorithmes : méthode exactes et méthode approchés, chaque méthode contient différentes méthodes comme PL, programmation dynamique ... (algorithmes exacte) et heuristique, Métaheurestique (algorithme approchés) .L'existence de plusieurs méthodes a pour but la résolution des problèmes d'ordonnancement d'une manière pertinente.

CHAPITRE 4

IMPLEMENTATION DU PROBLEME M-MACHINES IDENTIQUES EN PARALLELES

Introduction

Dans ce chapitre , nous allons passer à la conception et l'implémentation d'un problème M-machine en parallèle par l'algorithme génétique et particulièrement NSGA-II (Non-dominated Sorting Genetic Algorithm II).

1- MATLAB (« *matrix laboratory* »)

Est un langage de programmation de quatrième génération émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. Les utilisateurs de MATLAB (environ un million en 2004[2]) sont de milieux très différents comme l'ingénierie, les sciences et l'économie dans un contexte aussi bien industriel que pour la recherche. Matlab peut s'utiliser seul ou bien avec des *toolbox* (« boîte à outils »).

2- Non-dominated Sorting Genetic Algorithm II (NSGA II)

Est une deuxième version de NSGA avec de nouvelles techniques, tente de résoudre toutes les critiques faites sur NSGA : complexité, utilisation du sharing et non élitisme. En remplaçant la fonction de 'sharing' afin d'éviter le réglage des paramètres exigés, une fonction de distance 'crowding' a été introduite. Cette fonction est liée à deux caractéristiques propres à chaque individu, respectivement le rang de 'non domination' et la distance de 'crowding'. Comparativement à NSGA, NSGA-II obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de ce qui fait de cet algorithme un des plus utilisés aujourd'hui.

Dans cet algorithme, à chaque génération t, une population de parents (Pt) de taille (N) et une population d'enfants (Qt) de taille de même taille (N) sont mélangées pour

produire une population ($Rt = Pt \cup Qt$), comme le montre la figure (4.1). Cet assemblage permet d'assurer l'élitisme. La population Rt 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 ($Pt + 1$) 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, (Fi), non inclus dans ($Pt + 1$).

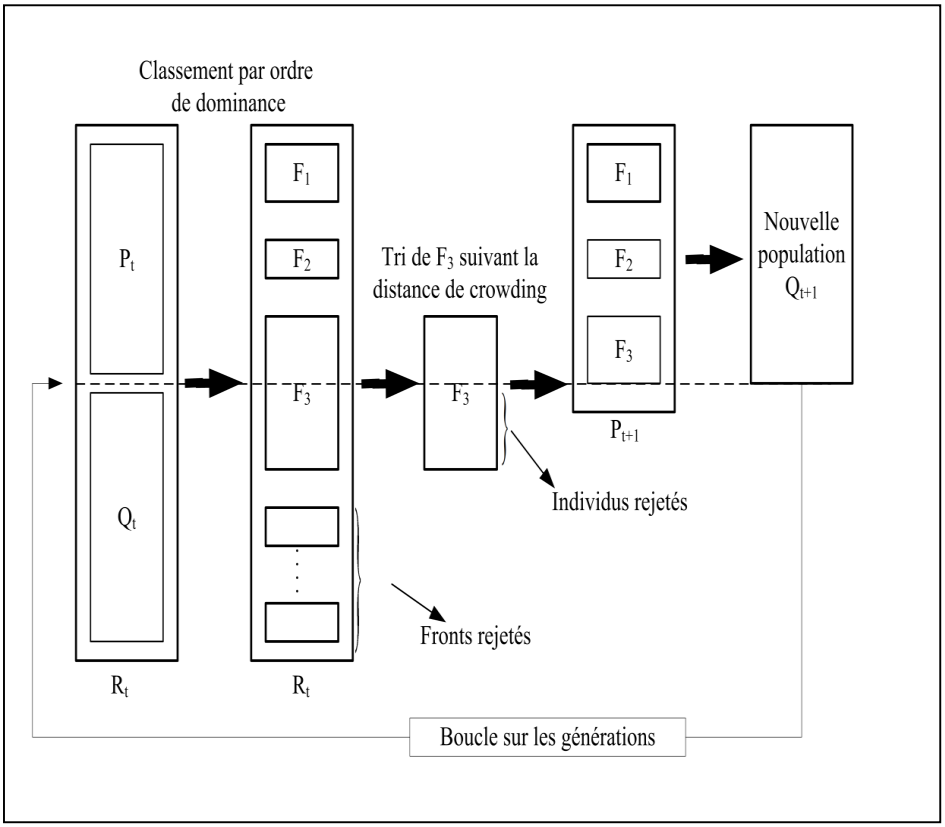


Figure 4.1 Schéma de l'évolution de l'algorithme NSGA-II [5].

Le but de cet opérateur est d'insérer les ($N - |P_t + 1|$) meilleurs individus de Fi qui manquent dans la population($Pt + 1$). 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 ($Pt + 1$) sont identifiés, une nouvelle population enfant ($Qt + 1$) 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 (n) 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 :

2-1 Une itération de NSGA-II

Après initialisation aléatoire de la population initiale $P0$, une itération de NSGA-II se déroule comme suit :

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 $R_t = Q_t \cup P_t$ Trier l'ensemble résultant R_t en sous-ensemble F_I.
3. Soit une nouvelle population $P_{t+1} = \emptyset$. Soit le compteur des sous-ensembles non dominés $i = 1$.
4. Tant que $|Pt + 1| + |Fi| < N, Pt + 1 \leftarrow Pt + 1 \cup Fi\ et\ i \leftarrow (i + 1)$.
5. Ordonner l'ensemble Fi 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 $P_t + 1$.

Il est important de noter que le tri de R_t en sous-ensembles non-dominés fait en stade 1 et le remplissage de la population $P_t + 1$ peuvent être effectués simultanément. Chaque fois qu'un nouveau front est trouvé on vérifie s'il peut rentrer dans $P_t + 1$ entièrement si ce n'est pas le cas, le processus du ranking s'arrête [5].

3- Option de NGPM

NGPM est l'abréviation de « A NsGa-II program in Matlab », qui est l'implémentation de NSGA-II dans Matlab.

Ce programme est écrit pour un problème d'optimisation par objets finis, le "croisement intermédiaire " et la " mutation gaussienne" sont adéquats pour nous utilisation. Ainsi, nous n'implantons pas d'autres opérateurs génétiques dans NGPM. Le codage réel/entier, la sélection de tournois binaire. L'opérateur de mutation Gaussien et l'opérateur de croisement intermédiaire fonctionnent biens dans ces outils.

4- Diagramme de Gantt

▪ Le diagramme de Gantt est un outil utilisé (généralement en complément d'un réseau PERT) en ordonnancement et gestion de projet et permettant de visualiser dans

le temps les diverses tâches liées composant un projet (il s'agit d'une représentation d'un graphe connexe, value et orienté).

▪ Il permet de représenter graphiquement l'avancement du projet sous la forme d'un graphe connexe (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) et orienté (bon ça c'est simplement à cause de la flèche du temps).

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

En résumé, un diagramme de Gantt répertorie toutes les tâches à accomplir pour mener le projet à bien, et expose la date à laquelle ces tâches doivent être effectuées (le planning).

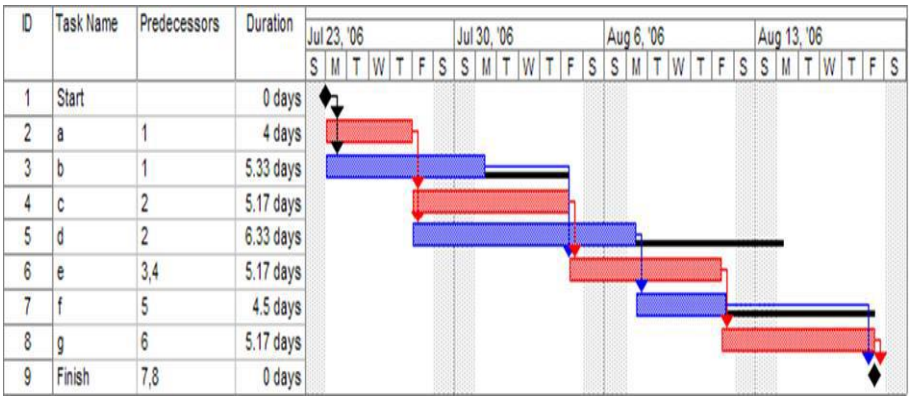


Figure 4.2 un exemple de Diagramme de Gantt simple.

5- Quelque exemple sur l'implémentation

Exemple 01 : Dans un problème de 4 machines et 7 jobs tels que les données sont :

Taches	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P (temps de traitement)	**4**	**3**	**4**	**5**	**2**	**6**	**3**
d (temps de livraison)	**8**	**8**	**12**	**11**	**14**	**18**	**20**
r (temps de disponibilité)	**0**	**2**	**2**	**4**	**6**	**8**	**10**

Tab 4.1 La table des taches Exemple 1.

Avec notre implémentation nous avons trouvé plusieurs solutions optimales d'ordre de l'ordonnancement on va choisie deux solutions optimale comme suite :

- La 1ére solution donne l'ordre de l'ordonnancement suivant :

Taches	1	2	3	4	5	6	7
n(indice de machine)	2	2	2	3	2	1	4
k (indice de position sur la machine)	1	2	3	1	4	1	1

Tab 4.2 La table de la 1ére solution Exemple 1.

au début temps on a n=1,k=1 les matrices Pr, R, D,C,T comme suite :

$$Pr = \begin{pmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 3 & 4 & 2 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 6 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 8 & 12 & 14 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 7 & 11 & 13 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Alors que Pr c'est la matrice de temps opératoire de la tâche j, R c'est la matrice de date de début, D c'est la matrice date de fin la tâche j, C c'est la matrice de makespan, T c'est la matrice retard réel de la tâche j.

- Diagramme de Gantt :

Figure 4.3 Diagramme de Gantt de 1ére solution d'exemple 1,

- La 2éme solution donne l'ordre de l'ordonnancement suivant :

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

Tab 4.3 La table de la 2éme solution Exemple 1.

Les matrices Pr, R, D, C, T sont :

$$Pr = \begin{pmatrix} 4 & 2 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ 2 & 10 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 12 & 14 & 0 & 0 & 0 & 0 & 0 \\ 8 & 18 & 0 & 0 & 0 & 0 & 0 \\ 8 & 20 & 0 & 0 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 6 & 8 & 0 & 0 & 0 & 0 & 0 \\ 4 & 14 & 0 & 0 & 0 & 0 & 0 \\ 5 & 13 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Alors on a $Cmax$=14, $\sum T$=0, le temps d'exécution est 49,0192 s

- Diagramme de Gantt :

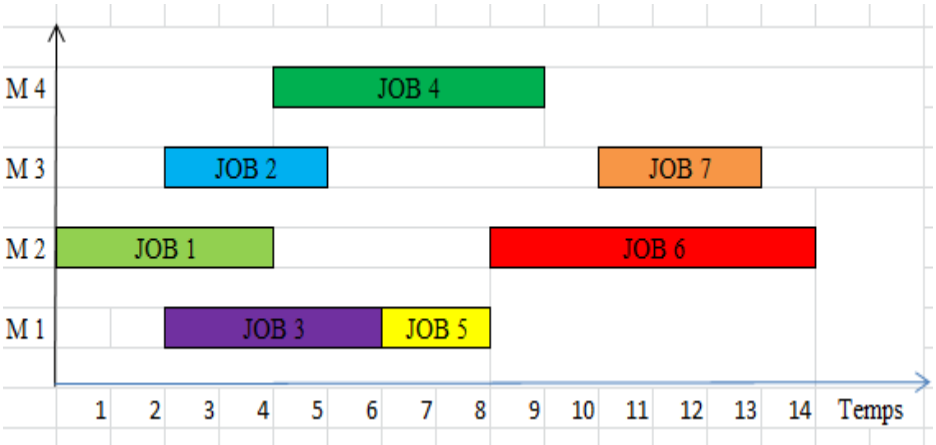


Figure 4.4 Diagramme de Gantt de 2émé solution d'exemple 1.

Exemple 02 : Dans un problème de 3 machines et 7 jobs tels que les données sont :

Taches	1	2	3	4	5	6	7
p(temps de traitement)	5	3	4	6	1	2	6
r(temps de livraison)	1	0	4	7	5	9	10
d (temps de livraison)	9	8	11	14	12	7	16

Tab 4.4 La table des taches Exemple 2.

avec notre implémentation nous avons trouvé plusieurs solutions optimales d'ordre de l'ordonnancement on va choisie deux solutions optimale comme suite :

- La 1ére solution donne l'ordre de l'ordonnancement suivant :

Taches	1	2	3	4	5	6	7
n(indice de machine)	3	2	1	2	1	1	3
k (indice de position sur la machine)	1	1	1	2	2	3	2

Tab 4.5 La table de la 1ére solution Exemple 2.

Les matrices Pr, R, D, C, T comme suite :

$$Pr = \begin{pmatrix} 4 & 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 4 & 5 & 9 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 1 & 10 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 11 & 12 & 7 & 0 & 0 & 0 & 0 \\ 8 & 14 & 0 & 0 & 0 & 0 & 0 \\ 9 & 16 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 8 & 9 & 11 & 0 & 0 & 0 & 0 \\ 3 & 13 & 0 & 0 & 0 & 0 & 0 \\ 6 & 16 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Diagramme de Gantt :

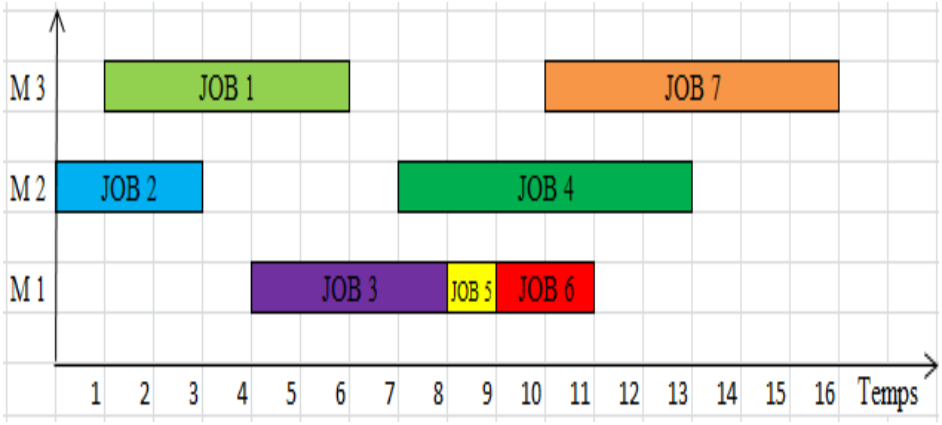


Figure 4.5 Diagramme de Gantt de 1ére solution d'exemple 2.

- La 2éme solution donne l'ordre de l'ordonnancement suivant :

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

Tab 4.6 La table de 2éme solution Exemple 2.

Les matrices Pr, R, D, C, T comme suite :

$$Pr = \begin{pmatrix} 4 & 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 4 & 9 & 5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 10 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 11 & 7 & 12 & 0 & 0 & 0 & 0 \\ 8 & 9 & 16 & 0 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 8 & 11 & 12 & 0 & 0 & 0 & 0 \\ 3 & 8 & 16 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Alors on a $Cmax$=16,$\sum T = 4$, le temps d'exécution est :18.4231s

- Diagramme de Gantt :

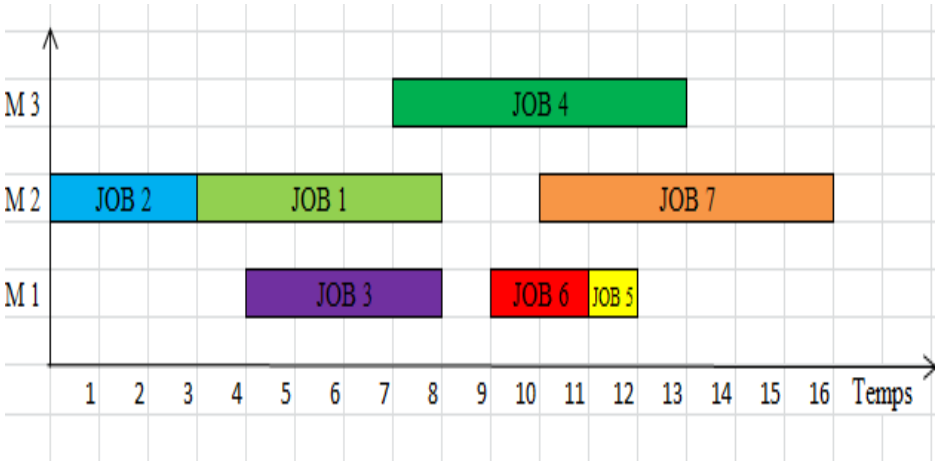


Figure 4.6 Diagramme de Gantt de 2éme solution d'exemple 2.

Exemple 03 : Dans un problème de 5 machines et 6 jobs tels que les données sont :

Taches	**1**	**2**	**3**	**4**	**5**	**6**
p(temps de traitement)	**5**	**2**	**3**	**8**	**1**	**7**
r(temps de livraison)	**1**	**3**	**2**	**5**	**9**	**7**
d (temps de livraison)	**5**	**6**	**9**	**0**	**14**	**15**

Tab 4.7 La table des taches Exemple 3.

avec notre implémentation nous avons trouvé plusieurs solutions optimales d'ordre de l'ordonnancement on va choisie deux solutions optimale comme suite :

- La 1ére solution donne l'ordre de l'ordonnancement suivant :

Taches	**1**	**2**	**3**	**4**	**5**	**6**
n(indice de machine)	**3**	**1**	**1**	**4**	**5**	**2**
k (indice de position sur la machine)	**1**	**1**	**2**	**1**	**1**	**1**

Tab 4.8 La table de 1ére solution Exemple 3.

Les matrices Pr, R, D, C, T comme suite :

$$Pr = \begin{pmatrix} 2 & 3 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 3 & 2 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 6 & 9 & 0 & 0 & 0 \\ 15 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 5 & 8 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Diagramme de Gantt :

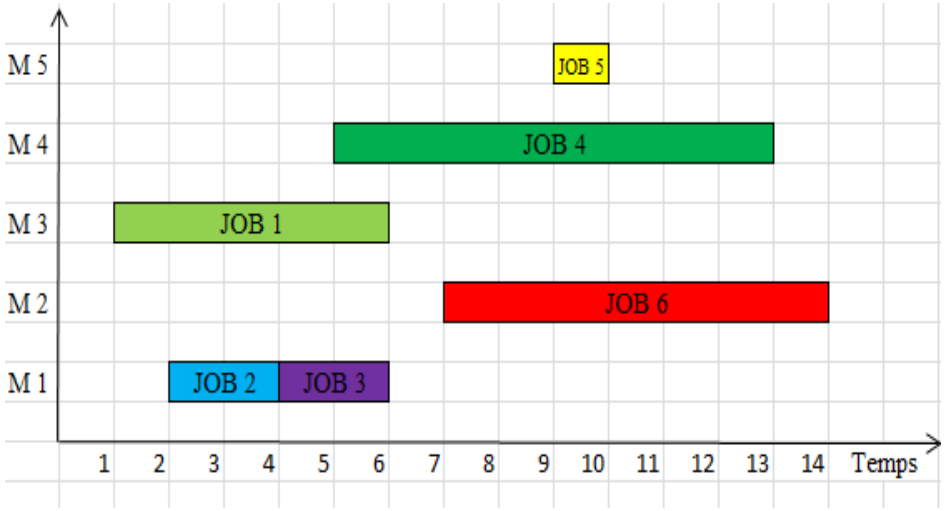


Figure 4.7 Diagramme de Gantt de 1ére solution d'exemple 3.

- La 2éme solution donne l'ordre de l'ordonnancement suivant :

Taches	**1**	**2**	**3**	**4**	**5**	**6**
n(indice de machine)	**3**	**1**	**2**	**4**	**5**	**3**
k (indice de position sur la machine)	**1**	**1**	**1**	**1**	**1**	**2**

Tab 4.9 La table de 2éme solution Exemple 3.

Les matrices Pr, R, D, C, T comme suite :

$$Pr = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 5 & 7 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 7 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 6 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 \\ 5 & 15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \\ 6 & 14 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Diagramme de Gantt :

Figure 4.8 Diagramme de Gantt de 2éme solution d'exemple 3.

Alors on a $Cmax$=14,$\sum T = 14$, le temps d'exécution est :31.1100s

Conclusion

Dans ce chapitre, nous calculons le makespan (Cmax) et la somme de retard ($\sum T$) en fonction de la mise en œuvre que nous avons fait avec mention le temps d'exécution.

CONCLUSION GENERALE

L'ordonnancement repose essentiellement sur la résolution de problèmes combinatoires. Ces problèmes consistent à rechercher , dans un ensemble de solutions possibles, une solution particulièrement intéressante au regard d'un ou de plusieurs critères. Ces problèmes sont pour plupart NP-complets.

Le problème traité dans ce travail est le problème d'ordonnancement de tâches dans un atelier M-Machine identique en parallèle en vue de minimiser le makespan et la somme de retard.

Dans ce mémoire, nous nous sommes, tout d'abord, intéressés à l'étude de l'ordonnancement dans sa globalité, avec ses différentes composantes (tâches, ressources, contraintes, critères), en rapport avec les différents types d'ateliers étudiés et avec les méthodes d'optimisation pour trouver de bonnes solutions aux problèmes correspondants en utilisant pour leur résolution, des méthodes exactes (programmation linéaire, branch & bound) et des méthodes approchées (recuit simulé, recherche tabou, algorithmes génétiques et algorithmes de colonies de fourmis).

Une méthode a été suggérée basée sur les algorithmes génétiques, ont été, ensuite, proposée pour la résolution de problèmes d'ordonnancement M-machine en parallèle en gestion de production : Non-dominated Sorting Genetic Algorithm II (NSGA II). Celle-ci ont été appliquée avec succès à la résolution de problèmes d'ordonnancement en gestion de production pour différents critères.

Comme perspectives, nous envisageons examiner d'autres algorithmes génétiques multicritères et les comparer avec le MOGA actuellement utilisé.

BIBLIOGRAPHIES

[1] Bahmani Y., 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.

[2] Baptiste M., Méthodes approchées pour la résolution d'un problème d'ordonnancement avec travaux interférant, Rapport final de PFE, Université de Tours, France 2014.

[3] Bellache N. E. I., Développement et implémentation d'un solveur bio inspiré pour la résolution d'un problème d'ordonnancement d'atelier, Mémoire Master en informatique, Université de M'sila , 2016 /2017.

[4] Benkaddour H., Aribi R., Méta-heuristiques parallèles pour la résolution des problèmes difficiles, Mémoire Master en Informatique, université Kasdi Merbah Ouargla, 2013.

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

[6] Dr-Derbala A.**,** Ordonnancement dans les ateliers , chapitre 2 formulation des problèmes d'ordonnancement ,page 18,université Saad Dahlab de Blida.

[7] El Bahloul S.**,** Flow-shop à deux machines avec des temps de latence : approche exacte et heuristique, mémoire présenté comme exigence partielle, Université du Québec partielle de la maitrise en informatique,2008.

[8] Flipo-Dhaenens C., Optimisation d'un réseau de production et de distribution, Thèse de doctorat, INPG de Grenoble. 1998.

[9] Gherboudj A.**,** Méthodes de résolution de problèmes difficiles académiques, Thèse Doctorat en informatique, Université de Constantine 2 ,2013.

[10] Group Gotha**,** Les problèmes d'ordonnancement, RAIRO. Recherche opérationnelle, tom 27,n°1(1993).

[11] Laetitia J., Métaheuristiques pour l'extraction connaissances application à la génomique, Thèse de Doctorat en informatique, Université des sciences technologique de LILLE U.F.R. D'I.E.E.A ,Année 2003.

[12] Marino W.**,** Les métaheuristiques : des outils performants pour les problème industriels**,** Université de Fribourg France Département d'informatique,2001.

[13] Meziane M. E. A., Optimisation par phases pour les problèmes d'ordonnancement des ateliers de type job-shop totalement flexibles, Mémoire de magister en des sciences, Université d'Oran, Algérie 2011.

Webographie

[14] Openeclassroms SAS, www.openclassrooms.courses.introduction-à-la-programmation-dynamique.com,consulté le 31 octobre 2017.

[15] Wikipédia, www.wikipédia.algorithme_de_colonies_de_fourmis.com, consulté le 8 mai 2018.

[16] Wikipédia, www.wikipédia.algorithme_génétique.com, consulté le 5 juin 2018.

[17] Wikipédia, www.wikipédia.ordonnancement_d'atelier.com, consulté le 16 février 2018.

[18] Wikipédia, www.wikipédia.programmation_par_contrainte.com, consulté le 7 mai 2018.

[19] Wikipédia, www.wikipédia.théorie_de_l'ordonnancement.com, consulté le 4 mai 2018.

الملخص :

المشكلة التي تتناولها هذه المذكرة هي جدولة العمليات على مجموعة أجهزة متوازية بنفس الصفات من أجل تقليل من وقت الاتمام الاكبر, يمكن إجراء العمليات بالتوازي على عدة أجهزة .قمنا بدراسة طريقة تستند إلى الخوارزميات الجينية ، وبرمجناها لحل المشكلة.

الكلمات الرئيسية: *الجدولة, الات متوازية متطابقة, وقت الاتمام الاكبر,* الخوارزمية الجينية.

Résumé :

Le problème traité dans cet mémoire concerne l'ordonnancement d'opérations sur des machines parallèles identiques en vue de minimiser le makespan et la somme des retard. Les opérations peuvent être exécutées en parallèle sur plusieurs machines. Une méthode a été suggérée basée sur les algorithmes génétiques, ont été, ensuite, proposée pour la résolution de problèmes.

Mots clés: *Ordonnancement, machines parallèles identiques, le makespan, algorithme génétique.*

Abstract :

The problem addressed in this thesis concerns the scheduling of operations on identical parallel machines in order to minimize the makespan and the sum of the delays. The operations can be performed in parallel on multiple machines. A method was suggested based on genetic algorithms, then it was used for solving the problem.

Key words: *Scheduling, identical parallel machines, the makespan, genetic algorithm*.