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

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

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Informatique Décisionnelle et Optimisation

Par : KRAM Zakaria

SUJET

Insertion d'une opération dans un problème d'ordonnancement a machines parallèles

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

Dr. Mouhoub Nacer Eddine	**Université de M'sila**	**Rapporteur**
……	**Université de M'sila**	**Président**
……	**Université de M'sila**	**Examinateur**

Promotion : 2018 /2019

Remerciements

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

En second lieux je tiens à remercier mon encadreur **Mr. MOUHOUB NACER EDDINE**

Je remercier aussi **Mr. GUERNA Abderrahim** et **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.

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

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

Toutes les lettres ne sauraient trouver les mots qu'il faut… Tous les mots ne sauraient exprimer la gratitude, l'amour, Le respect, la reconnaissance… Aussi, c'est tout simplement que Je dédie ce travail qui n'aura jamais pu voir le jour sans les soutiens

Indéfectibles et sans limite de mes chers parents **(KRAM Merzoug et ATTIA Malika)** qui ne cessent de me donner avec amour le nécessaire pour que je puisse arriver à ce que je suis aujourd'hui.

Que dieux vous protège et que la réussite soit toujours à ma portée pour que je puisse vous combler de bonheur.

Je dédie aussi ce travail à :

- Mes grands-parents.
- Mes frères, ma sœur
- Mes frères qui je connaît dans ma vie.
- Mes oncles, mes tantes et leur famille.
- Tous mes cousins et cousines.
- Tous mes amis, mes collègues et tous ceux qui m'estiment
- A mon encadreur Mr. Mouhoub Nacer Eddin

Table des matières

Chapitre 5 : Conception et réalisation

Liste des Figures

Liste des Tables

List des abbreviations

FIFO: First in First Out

SPT: Shortest Processing Time

LPT: Longest Processing Time

EDD: Earliest Due Date

SRPT: Shortest Remaining Processing Time

AG : Algorithme Génétique

PL : Programmation Linéaire

ST : Slack Time

Introduction Générale

L'organisation et la gestion de la production conditionnent le succès des projets du monde de l'entreprise et de la recherche. Dans ce processus, la fonction ordonnancement vise à organiser l'utilisation des ressources technologiques ou humaines pour répondre à une demande ou satisfaire un plan de production préparé par la fonction planification. Ainsi, des programmes ambitieux privés ou publics ont recours à la fonction ordonnancement pour appréhender la complexité, améliorer les délais ou même s'adapter à des évènements imprévus. Les problèmes d'ordonnancement apparaissent dans de nombreux domaines :l'industrie (atelier, gestion de production), la construction (suivi de projets), mais aussi l'informatique(gestion des processus) et l'administration (emplois du temps).

Un problème d'ordonnancement est composé de façon générale d'un ensemble de tâches soumises à certaines contraintes, et dont l'exécution nécessite des ressources. Résoudre un problème d'ordonnancement consiste à organiser ces tâches, c'est-à-dire à déterminer leurs dates de démarrage et d'achèvement, et à leur attribuer des ressources, de telle sorte que les contraintes soient respectées. Les problèmes d'ordonnancement sont très variés. Ils sont caractérisés par un grand nombre de paramètres relatifs aux tâches (morcelables ou non, indépendantes ou non, durées fixes ou non), aux ressources (renouvelable sou consommables), aux types de contraintes portant sur les tâches (contraintes temporelles, fenêtres de temps . . .), aux critères d'optimalité liés au temps (délai total, délai moyen, retards . . .), aux ressources (quantité utilisée, taux d'occupation . . .) ou à d'autres coûts (production, lancement, stockage . . .). Parmi tous ces problèmes, nous nous intéresserons à l'optimisation de critères réguliers pour des problèmes d'atelier et de fournées classés NP-Difficiles.

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 l'enchaînement. Plus précisément, les tâches sont regroupées en « n » entités appelées travaux ou lots. Chaque lot est constitué de « n » tâches à exécuter sur « m » machines distinctes.

Il existe Une classification très répandue des ateliers, du point de vue ordonnancement, est basée sur les différentes configurations des machines. Les modèles les plus connus sont ceux d'une machine unique, de machines parallèles, d'un atelier à cheminement unique (flow shop) ou d'un atelier à cheminement multiple (job shop). Un critère d'optimalité souvent étudié est la minimisation du délai total de l'ordonnancement (makespan).

Ce mémoire est composé de cinq chapitres dont nous présentons une brève description dans les paragraphes suivants :

Le premier chapitre, qui présente les problèmes optimisation combinatoires et les différentes manières de résoudre ce type de problèmes.

Le deuxième chapitre de cette thèse, nous études les problèmes d'ordonnancement, voyons les différent domaines et l'objectif de l'ordonnancement, puis nous verrons les différents éléments du problème d'ordonnancement. Le troisième chapitre nous allons expliquer les problèmes d'atelier et présent leur schéma de classification, en suite a expliqué les types de problèmes d'ordonnancement d'ateliers et leur complexité. Ensuite en le quatrième chapitre on explique seulement sur les problèmes de Machine parallèle, dans le dernier chapitre je créé une application qui présenté un ordonnancement final avec le digramme de Gantt, et ces ordonnancements traités avec un algorithme génétique, et je proposé une solution pour ajoute une tâche qui arrivante aléatoire dans l'ordonnancement.

Enfin, nous terminons par une conclusion, quelques remarques et perspectives.

Chapitre 1 Problème du L'optimisation combinatoire

1- Introduction

L'optimisation combinatoire, aussi appelée optimisation discrète, est une branche de l'optimisation en mathématiques appliquées et en informatique, également liée à la recherche opérationnelle, l'algorithmique et la théorie de la complexité.

Dans ce chapitre introductif, nous Savoir d'abord la définition d'un problème d'optimisation combinatoire, et comment faire une résolution, et ensuite on a voire les différentes méthodes de résolution un problème d'optimisation combinatoire.

2 - Définition

L'optimisation combinatoire est minimiser ou maximiser une fonction souvent appelée fonction coût, d'une ou plusieurs variables soumises à des contraintes. L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. [MR 08]

Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données. [AL 10]

3- Résolution d'un problème d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points particuliers :

- La définition de l'ensemble des solutions réalisables.
- L'expression de l'objectif à optimiser.
- Le choix de la méthode d'optimisation à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution. Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application.

4- Les méthodes de résolution

La principale difficulté à laquelle est confronté un décideur, en présence d'un problème d'optimisation est celui du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable.

Dans la littérature, on distingue essentiellement deux classes des méthodes, (Figure 1.1)

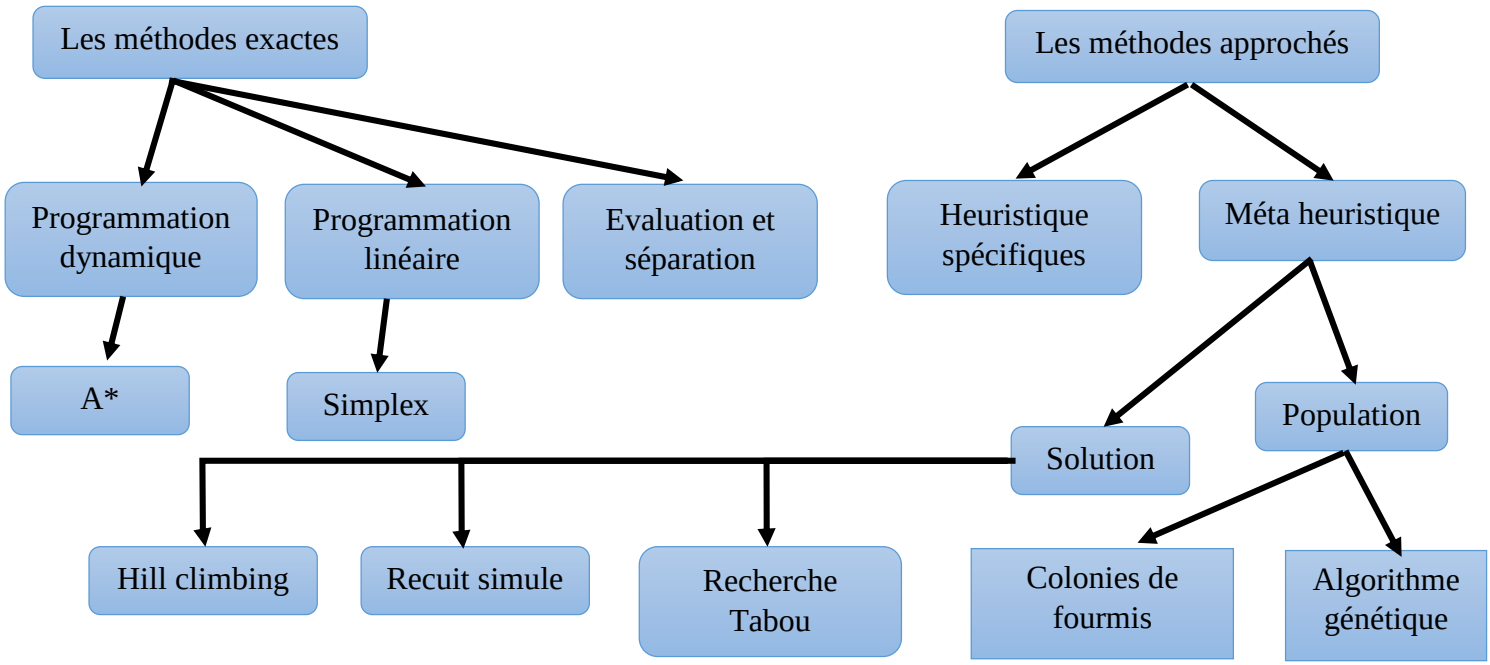

Figure 1.1 : Classification des méthodes d'optimisation combinatoire

- **Les méthodes exactes** Assurant la résolution des problèmes en un temps polynômial
- **Les méthodes approchées(heuristiques)** Permettant de trouver une solution proche de l'optimale en un temps tolérable. [AK 12]

4.1 - Les méthodes exactes

Ces méthodes sont généralement utilisées pour résoudre des problèmes de petite taille.

Dans ce cas, le nombre de combinaisons possibles est suffisamment faible pour pouvoir explorer l'espace de solutions en un temps raisonnable. On distingue trois sous-classes de

méthodes exactes la procédure de séparation et d'évaluation (Branch and Bound) , la programmation dynamique et la programmation linéaire . [KB 17]

4.1.1 - La méthode de séparation et évaluation (Branch and Bound)

L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (BB), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des nœuds, et des feuilles.

L'algorithme de séparation et évaluation est basé sur trois axes principaux

- **L'évaluation** Permet de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale.
- **La séparation** Consiste à diviser le problème en sous-problèmes. Ainsi, en résolvant tous les sous problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Cela revient à construire un arbre permettant d'énumérer toutes les solutions.
- **La stratégie de parcours** Il existe trois parcours possible de l'arbre

  - **La profondeur d'abord** Cette stratégie avantage les sommets les plus éloignés de la racine en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.
  - **Le meilleur d'abord** Cette stratégie consiste à explorer des sous-problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.
  - **La largeur d'abord** Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées. [BB 08]

4.1.2 – La programmation Dynamique

La programmation dynamique a été appelée comme cela depuis 1940 par Richard Bellman et permet d'appréhender un problème de façon différente de celle que l'on pourrait imaginer au premier abord.

Le concept de base est simple, une solution optimale est la somme de sous-problèmes résolus de façon optimale. Il faut donc diviser un problème donné en sous-problèmes et les résoudre un par un. (Figure 1.2)

La méthode comprend deux étapes

- Prolonger le problème dans une famille de sous problèmes de même nature.
- Relier par une relation de récurrence les solutions optimales de ses sous-ensembles.

La Programmation Dynamique est une méthode exacte de résolution de problèmes d'optimisation, due essentiellement à R. Bellman (1957).

Bien que très puissante, son cadre d'application est relativement restreint, dans la mesure où les problèmes qu'elle adresse doivent vérifier un principe dit principe d'optimalité qui stipule qu'une solution optimale d'un problème de taille n peut s'exprimer en fonction de la solution optimale de problèmes de taille inférieure à « n ». [HF 16]

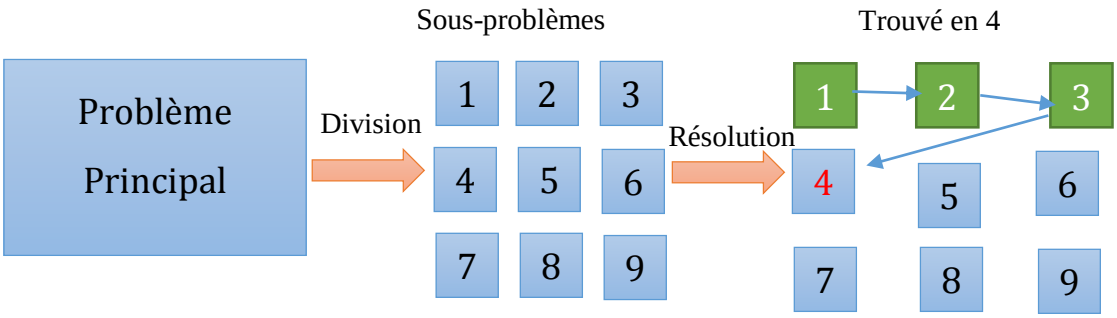

Figure 1.2 : La division en sous-problèmes

4.1.3 - La programmation linéaire

La programmation linéaire (PL) est une branche de l'optimisation permettant de résoudre de nombreux problèmes économiques et industriels. La programmation linéaire désigne la manière de résoudre les problèmes dont la fonction objective et les contraintes sont toutes linéaires.

Si l'ensemble de solutions possibles « S » est formulé comme un ensemble de variables à valeurs dans l'ensemble des réels « R » et si on a des contraintes à satisfaire des inégalités linéaires et si « f » est une fonction linéaire en ces variables, on parle alors d'un problème de programmation linéaire (PL).

Plusieurs problèmes réels de recherche opérationnelle peuvent être exprimés comme un problème de PL. Pour cette raison un grand nombre d'algorithmes pour la résolution d'autres

problèmes d'optimisation sont fondés sur la résolution de problèmes linéaires. Un programme linéaire peut être défini comme suit

$$\begin{cases} min\ c^T x \\ Ax \geq b \\ avec\ c;\ x \in R^n;\ b \in R^m;\ A \in R^{m*n}, \end{cases}$$

« X » est un vecteur n-dimensionnel représentant la solution qui doit être optimisée. C'est un vecteur de la même taille représentant la fonction objectif « $C^T X$» De même, la matrice « A » représente avec le vecteur « b » les contraintes du problème linéaire.

Une solution d'un programme linéaire est une affectation de valeurs aux variables du problème. Une solution est réalisable si elle satisfait toutes les contraintes du problème. L'algorithme d'optimisation le plus couramment utilisé en programmation linaire est l'algorithme du simplex. [HT 07]

Ainsi, étant donné un ensemble d'inégalités linéaires sur « n » variables réelles, l'algorithme peut trouver la solution optimale pour un problème linéaire d'une manière itérative en démarrant avec une solution initiale. [AL 10]

4.2 - Les méthodes approchées

Une méthode heuristique ou approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objective en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principale de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, d'un autre côté les algorithmes d'optimisation tels que les algorithmes de recuit simulé, les algorithmes tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires.

Les méthodes approchées englobent deux classes

- Les méthodes heuristiques
- Les méta-heuristiques

4.2.1 - Les méthodes heuristiques

Les heuristiques sont des méthodes empiriques basées sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de ces méthodes est d'intégrer des

stratégies de décision pour construire une solution proche de l'optimum, tout en essayant de l'obtenir en un temps de calcul raisonnable. [AK 12]

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

4.2.2 - Les méta-heuristiques

Le mot méta-heuristique est dérivé de la composition de deux mots grecs

- Heuristique qui vient du verbe heuriskein et qui signifie 'trouver'.
- Méta qui est un suffixe signifiant au-delà dans un niveau supérieur.

Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, les méta-heuristiques ont fait leur apparition.

Sont des méthodes inspirées de la nature, ce sont des heuristiques modernes dédiées à la résolution des problèmes et plus particulièrement aux problèmes d'optimisation.

Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

Les méta-heuristiques se sont des méthodes inspirées de la nature, ce sont des heuristiques modernes dédiées à la résolution des problèmes et plus particulièrement aux problèmes

d'optimisation, qui visent d'atteindre un optimum global généralement enfoui au milieu de nombreux optimal locaux.

4.2.3 - Classification des méthodes méta-heuristiques

Il existe plusieurs façons de classer les méta-heuristiques, on en donne quelques une, et nous adopterons celle faisant la différence entre les méthodes de trajectoire et les méthodes basées sur une population. [PM 05]

1. **Méthodes de trajectoire** elles manipulent une seule solution à la fois et tentent itérativement d'améliorer cette solution. Elles construisent une trajectoire dans l'espace des solutions en tentant de se diriger vers des solutions optimales. Par exemple le recuit simulé.

Le recuit simulé

Le recuit simulé est une méta heuristique inspirée d'un processus utilisé en métallurgie. Ce Processus alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à Minimiser l'énergie du matériau. Elle est aujourd'hui utilisée en optimisation pour trouver les extrémaux d'une fonction. Elle a été mise au point par trois chercheurs de la société IBM, S.Kirkpatrick, C.D.Gelatt et M.P.Vecchi en 1983 et indépendamment par V.Cerny en 1985. Le recuit simulé s'appuie sur l'algorithme de Métropolies, qui permet de décrire le comportement d'un système en équilibre thermodynamique à une certaine température T, partant d'une configuration donnée (solution initiale). Par analogie avec le processus physique, la fonction objective à minimiser deviendra l'énergie E du système. [WL 05]

2. **Méthodes qui travaillent avec une population de solutions** en tout temps on dispose d'une "base" de plusieurs solutions, appelée population. L'exemple le plus connu est l'algorithme génétique. [SD 14].

Les algorithmes génétiques

Les algorithmes génétiques (AGs) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance

(fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante. [TV 01]

5- 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. [MA 11].

5.1 - 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. [MA 11].

5-2 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 polynomilement 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 [MA 11].

.

Chapitre 02 Les problème d'ordonnancement

1- Introduction

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises.

Dans ce chapitre nous pouvons voir les différents domaine concernés l'ordonnancement et ses objectif, on a vu les différents éléments d'un problème de ordonnancement, et à la fin on voire comment Résoudre un problème d'ordonnancement.

2- Définition

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 défini comme :

« Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leur date de début ». [SO 11]

3- Les domaines concernés [MA 05]

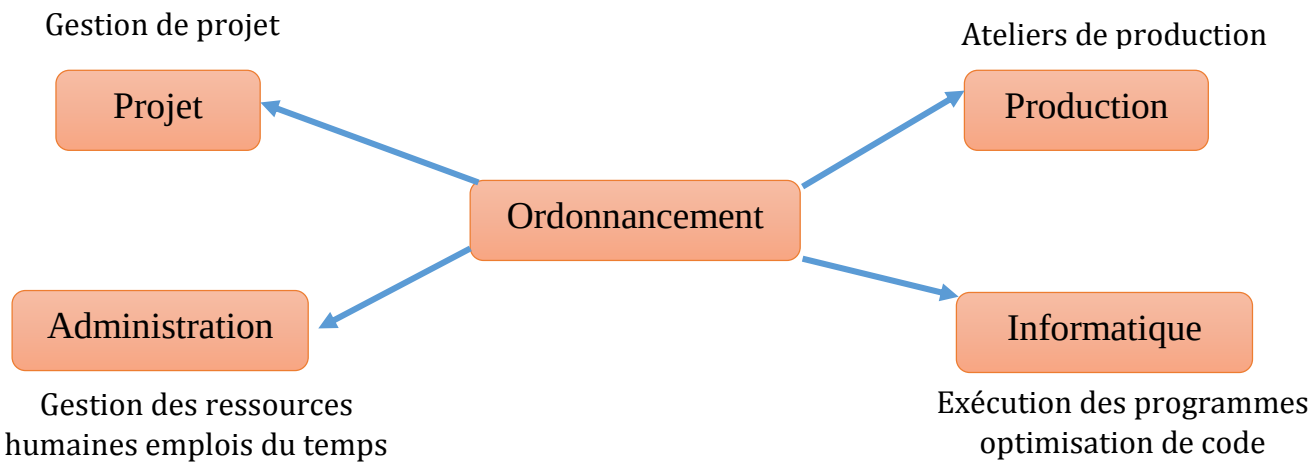

Figure 2.1 : Les domaine concernés des problèmes d'ordonnancement

4- Les objectifs de l'ordonnancement

Le traitement de l'ordonnancement dans la littérature s'est tout d'abord orienté vers une optimisation monocritère. L'environnement manufacturier évoluant rapidement et la concurrence devenant de plus en plus acharnée, les objectifs des entreprises se sont diversifiés et le processus d'ordonnancement est devenu de plus en plus multicritère. Les critères que doit satisfaire un ordonnancement sont variés. D'une manière générale on distingue plusieurs classes d'objectifs concernant un ordonnancement.

- **Les objectifs liés au temps**

  L'ordonnancement joue sur la classification des tâches pour minimiser le temps total d'exécution, du temps moyen d'achèvement, des durées totales de règles ou des retards par rapport aux dates de livraison.

- **Les objectifs liés aux ressources**

  L'ordonnancement pour objectifs d'optimisation l'utilisation des ressources ou le nombre de ressource nécessaires pour réaliser un ensemble de tâches…etc ;

- **Les objectifs liés au coût**

  Ces objectifs sont généralement de minimiser les couts, de lancement, de production, de stockage, de transport, etc.

- **Les objectifs liés à l'énergie ou au débit**

  Les objectifs à satisfaire au niveau de l'ordonnancement son issus des objectifs globaux de l'entreprise par décomposition. Cette décomposition conduit à une structure d'objectifs qui permet de gérer les contradictions et les compromis.

5- Les éléments d'un problème d'ordonnancement

L'ordonnancement est décrit généralement par les quatre éléments sont retenus pour décrire d'une façon explicite un ordonnancement, ces éléments sont : les tâches, les ressources, les contraintes et les critères à prendre en considération lors du processus d'optimisation. Dans ce qui suit, on donnera une définition détaillée de chacun de ces éléments.

- **Les tâches**

Une tâche (task en l'anglais) fait référence à la réalisation d'une opération, cette dernière étant caractérisée par des informations pertinentes nécessaires à la résolution du problème étudié. Ces informations peuvent être différentes selon des critères (l'objectif de l'étude et le point de vue). A titre d'exemple, le planeur s'intéressera au temps requis pour réaliser

l'opération sur une machine donnée, au type de ressources que cette opération consomme et à la quantité, tandis que l'opérateur va lui s'intéresser au procédé de réalisation de l'opération.

Dans le cadre des problèmes d'ordonnancement, parmi les informations utiles, on trouve le temps nécessaire pour la réalisation d'une tâche, la(les) ressource(s) qu'elle requiert (une ressource étant renouvelable ou non), une date de début au plus tôt, une date de fin souhaitée et une date de fin impérative. Des contraintes de précédence conditionnent l'exécution d'une tâche par rapport à la fin d'exécution d'autres tâches. Une tâche est alors considérée comme l'exécution d'une opération sur une certaine machine. Plusieurs tâches peuvent être regroupées pour constituer un « job » pour lequel on donne la gamme de production. Il est également nécessaire de connaitre la nature de la tâche, par exemple savoir si une tâche est interruptible ou non, c'est-à-dire de savoir si le traitement d'une tâche sur une ressource donnée peut s'interrompre pendant une certaine durée et de le poursuivre ensuite, cela se traduit par l'autorisation de la préemption des tâches ou non. Les notations suivantes sont introduites pour décrire une opération. Chaque variable, représentera une information particulière se rapportant à la réalisation de l'opération :

- r_i : Pour représenter la date de disponibilité de la tâche « i » (release date).
- t_i : Pour représenter la date de début de la tâche « i » (start date).
- c_i: Pour représenter la date de fin d'exécution de la tâche « i » (completion time).
- d_i : Pour représenter la date d'échéance de la tâche « i » (due date).
- p_i: Pour représenter la durée opératoire de la tâche « i » (processing time date).
- $F = c_i - r_i$: Pour représenter la durée de séjour de l'opération « i » sur la machine avant qu'elle redevienne disponible (flow time).
- $L_i = c_i - d_i$: Exprime le retard algébrique (lateness) entre la fin d'exécution de la tâche « i » par rapport à sa date d'échéance .
- $T_i=\max(L_i, 0)$: Qui exprime le retard absolu de la tâche « i » (tardiness) .
- $E_i=\max(-L_i , 0)$: Qui exprime l'avancement (earliness) de la tâche « i » .

Ces variables sont notamment retenues dans la littérature pour représenter une opération. La « Figure 2.2 » montre les relations entre les différentes variables représentant une tâche.

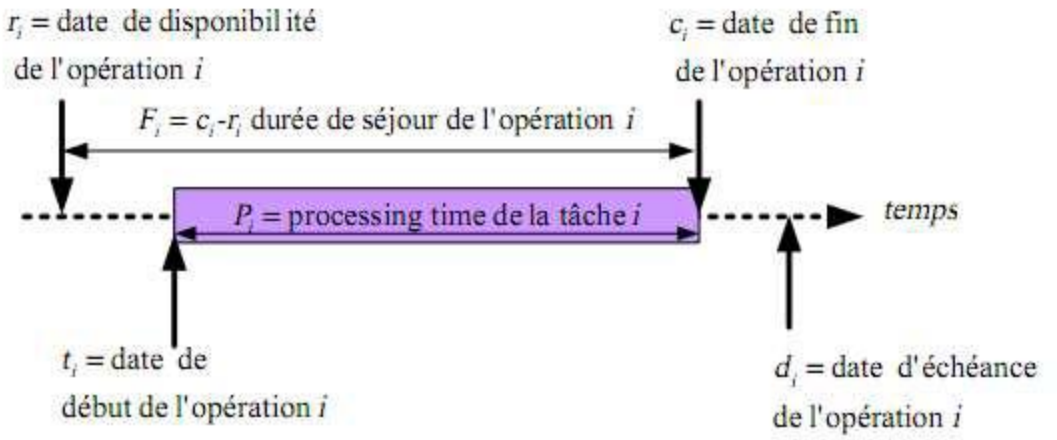

Figure 2.2 : La caractéristique d'une tache

- **Les ressources**

Deux familles de ressources existent

La première famille de ressources est dite renouvelable c'est-à-dire que ces ressources sont réutilisables après la fin d'exécution des opérations. Cette famille est assemblée de machines d'hommes et des outils de productions. Généralement des ressources additionnelles sont considérées comme par exemple l'utilisation des moyens de transport, ou des robots, ou des ressources d'un autre genre comme les buffers d'entrée/sortie. Pendant ces ressources on distingue aussi, des ressources disjonctives qui ne peuvent exécuter qu'une seule opération à la fois et des ressources cumulatives qui savent exécuter plusieurs opérations à la fois par exemple des machines parallèles, notons qu'une machine fait partie de l'ensemble des ressources renouvelable : une machine de capacité 1 est une ressource disponible en un unique exemplaire.

La deuxième famille est dite consommable et regroupe toutes les ressources qui deviennent indisponibles après une utilisation, par exemple des matières premières de l'argent, de l'énergie etc.

- **Les contraintes**

Les contraintes expriment les restrictions que peuvent prendre conjointement les variables de décision. Donc les contraintes nous renseignent sur les limites imposées par l'environnement. On distingue plusieurs types de contraintes, qui sont**,** Les contraintes de positionnement temporelles, Pour la réussite d'un projet ou d'un plan de production, on doit se soumettre aux impératifs de production (réalisation) traduits par le respect d'un planning fixé avant. Au niveau global on parlera d'une date de lancement d'une tâche et d'une date de livraison. A un autre niveau de détail plus fin, pour une tâche ou une opération. Ce dernier

niveau se traduit par la définition des dates suivante : date de disponibilité « ri », date d'échéance « di ».

Les contraintes de précédence, une contrainte qui lie le début d'une activité à la fin d'un autre est appelée contrainte de succession ou de précédence. Les gammes opératoires sont un exemple de ces contraintes, d'autres contraintes sont imposées dans certain cas telles que les contraintes de synchronisation, de simultanéité, ou de recouvrements etc.

Les contraintes de ressources représentent le fait que les activités requièrent tout au long de leur exécution une certaine quantité de ressources**,** elle concernent les deux points suivants :

- L'utilisation de ces ressources (leur nature, la quantité nécessaire, et les caractéristiques d'utilisation).
- La disponibilité et la quantité de ces ressources.

Ces contraintes nous donnent une information précise sur la nature de l'atelier, et influencent le choix des méthodes d'optimisation, Il est important de noter qu'en fonction des objectifs de l'étude, ces contraintes peuvent être strictes ou non. Lorsqu'elles sont strictes, elles constituent des obligations à respecter. Lorsqu'elles ne sont pas strictes, on peut ne pas les satisfaire et on les appelle alors des contraintes de préférence. [LA 01]

6- 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 alors 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.

7 -Représentation des problèmes d'ordonnancement

Il existe des sortes de représentations possibles d'un problème d'ordonnancement, le diagramme de Gantt, le graphe Potentiel-Tâches et la méthode PERT.

7.1 - Le diagramme de Gantt

La représentation par Le diagramme de Gantt est la plus courante pour l'ordonnancement. Celui-ci représente une opération par un segment ou une barre horizontale, dont la longueur est proportionnelle à sa durée opératoire.

Donc, sur ce diagramme sont indiqués, selon une échelle temporelle : l'occupation des machines par les différentes tâches, les temps morts et les éventuelles indisponibilités des machines dues aux changements entre produits. [JH 06]

Deux types de diagramme de Gantt sont utilisés : Gantt ressources et Gantt tâches (voir la Figure 2.3)

Le diagramme de Gantt ressource, est composé d'une ligne horizontale pour chaque ressource (machine). Sur cette ligne, sont visualisées les périodes d'exécution des différentes opérations en séquence et les périodes de l'oisiveté de la ressource.

Le diagramme de Gantt tâches permet de visualiser les séquences des opérations des tâches, en représentant chaque tâche par une ligne sur laquelle sont visibles, les périodes d'exécution des opérations et les périodes ou la tâche est en attente des ressources.

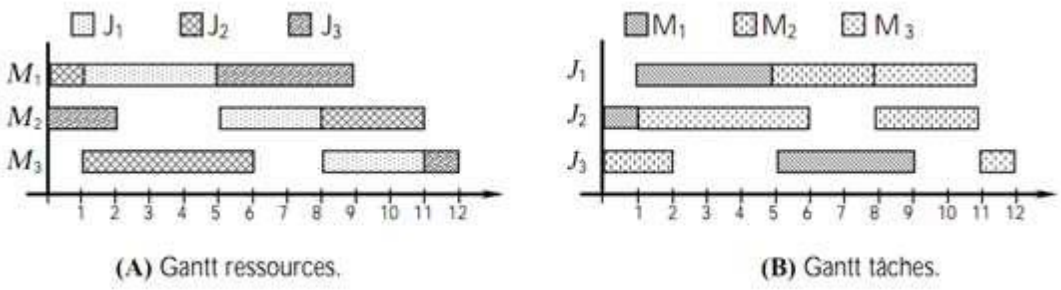

Figure 2.3 Le diagramme de Gantt

7.2- Graphe potentiel-tâches

Cette outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à évènements discrets.

Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les contraintes par des arcs, comme le montre la (Figure 2.4). Ainsi, les arcs peuvent être de deux types

• les arcs conjonctifs illustrant les contraintes de précédence et indiquant les durées des tâches.

• les arcs disjonctifs indiquant les contraintes de ressources

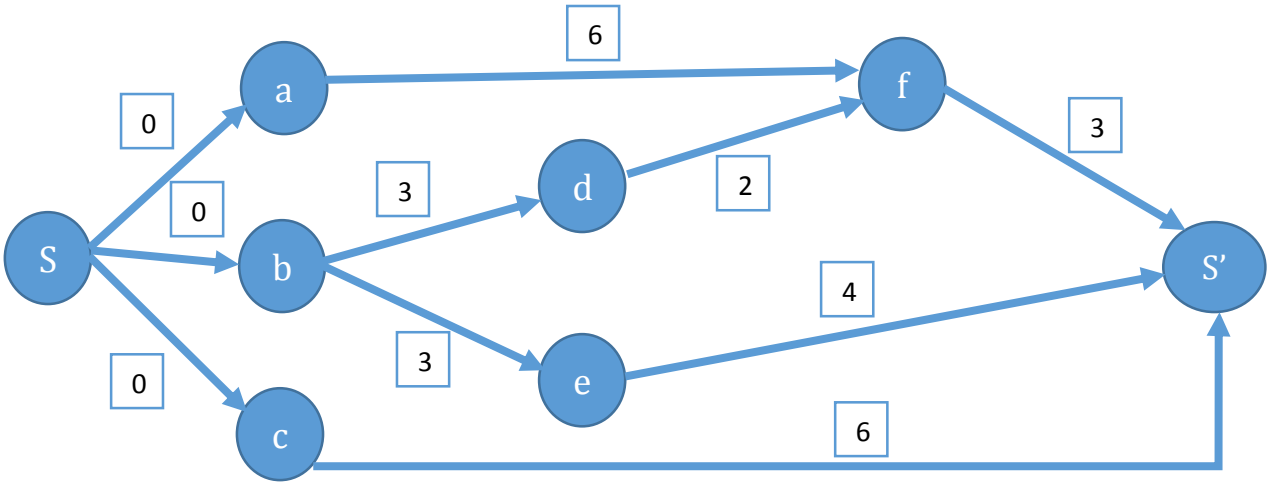

Figure 2.4 Graphe Potentiel-Tâches d'un ordonnancement

Dans l'exemple représenté dans « Figure 4.2 » la tâche « f » ne peut s'exécuter que si a et d ont été réalisées. Donc, pour exécuter « a » il faut 6 mois et pour exécuter « d » il faut 2+3 mois. La tâche « f » ne pourra commencer au plus tôt que 6 mois après le début du projet : c'est donc le plus long chemin entre « a » et « f ».

- la durée du projet, qui correspond au plus long chemin entre S (tâche de début du projet) et S' (tâche de fin du projet).

7.3 - Method PERT (Program Evaluation and Research Task)

Cette représentation, semblable à la précédente, permet de représenter une tâche par un arc, auquel est associé un chiffre qui représente la durée de la tâche. Entre les arcs, figurent des cercles, appelés sommets ou événements, qui marquent l'aboutissement d'une ou de plusieurs tâches. Ces cercles sont numérotés afin de suivre l'ordre de succession des divers évènements. Les méthodes graphiques ont connu une très importante évolution surtout avec l'apparition des Réseaux de Pétri, qui permettent de traduire plusieurs notions fondamentales ayant un lien avec les problèmes d'ordonnancement, telles que :

- Les conflits sur les ressources.
- Les durées opératoires certaines.
- Les durées opératoires aléatoires.
- Les gammes.
- Les disponibilités, les multiplicités et les capacités de ressources

la répétitivité, etc. ….

Chapitre 3 Les problèmes d'ordonnancement d'atelier

1- Introduction

L'ordonnancement d'atelier consiste à organiser dans le temps le fonctionnement d'un atelier pour utiliser au mieux les ressources humaines et matérielles disponibles dans le but de produire les quantités désirées dans le temps imparti.

Dans ce chapitre on peut expliquer c'est quoi un problèmes d'ordonnancement d'atelier, et on peut connu aussi ses schémas de classification, on peut faire une recherche sur les type de problème d'ordonnancement d'atelier, et en fin on peut voir la complexité des problèmes et sont types.

2 –Définition :

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 ni opérations, l'opération j du travail i est notée (i,j) avec (i, 1) < (i, 2) < ... < (i, ni) si le travail i possède une gamme (A < B signifie A précède B). Une opération (i,j) utilise la machine m_{ij} pendant toute sa durée p_{ij} et ne peut être interrompue .

Un atelier se définit 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 (voir la figure 3.1).

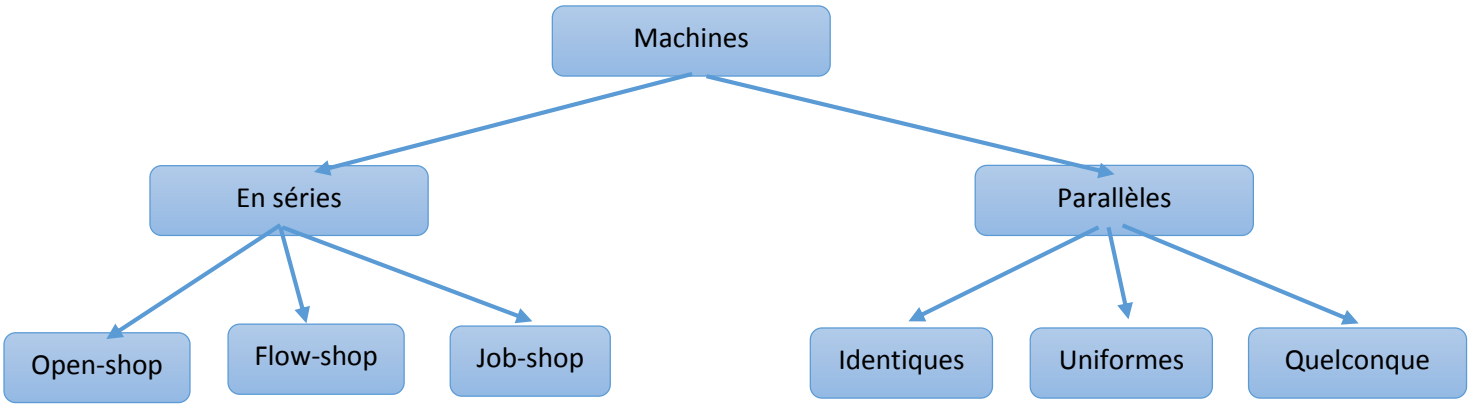

Figure 3.1. Les types des machines [SO 11]

3 - Schémas de classification [SO 11]

Nous suivons les schémas de classification proposée par Graham 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.**
  - $\beta^1 = P_{mtn}$ si la préemption des tâches est autorisée, sinon β1 est absent.
  - S'il y a des contraintes de précédence entre les tâches β2 {prec, chain, tree }, sinon β^2 est vide.
  - $\beta^3 = r_i$ si les dates de début au plus tôt r_i (ou dates de disponibilité) des tâches ne sont pas forcément identiques, sinon (j, $r_i = 0$) β3 est absent.3
  - β^4 = j si la tâche ou le job possède une date de fin obligatoire.
- **γ : le (ou les) critère(s) à optimiser.**

4 - Les types des Problèmes

4.1- Problèmes à une Machine

Toute tâche J_j,j∈ {1,….,n} de durée P_j (processing time) s'exécute sur une machine qui ne peut traiter plus qu'une tache à la fois [IB 14] .Le champ α1 est absent et α2 = 1 (Figure 3.2).

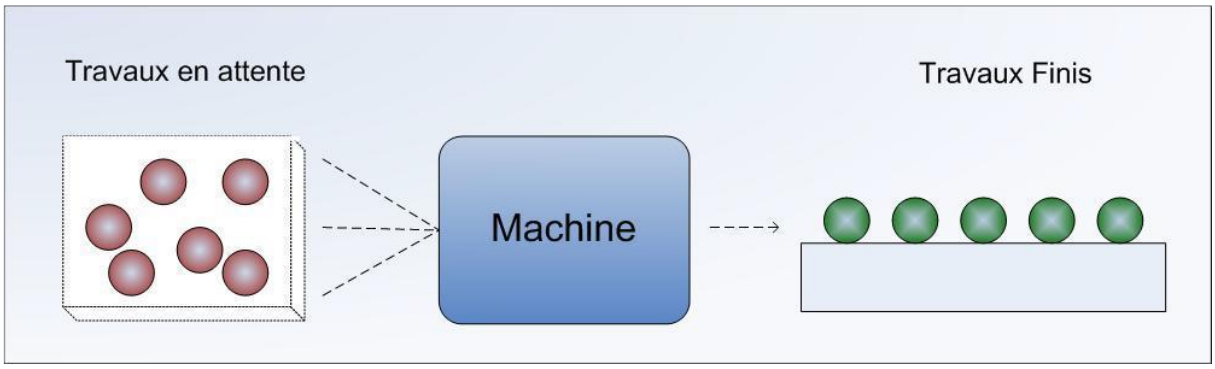

Figure 3.2 : Les problèmes d'ordonnancement a une machine. Tirée de [1]

4.2- Problèmes à machines parallèles

Toute tâche J_j, j∈ {1,….,n} peut être exécutée indifféremment sur une des « m » machines mises en parallèle . [IB 14]

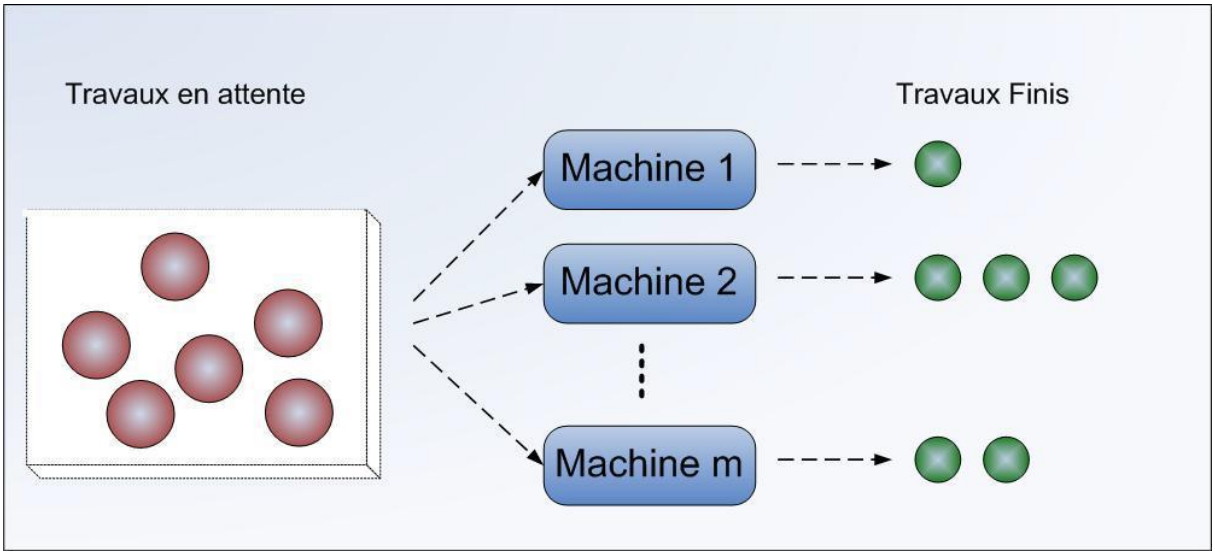

Figure 3.3 Les problèmes d'ordonnancement a machine parallèle. Tirée de [1]

- Si toutes les machines ont la même vitesse d'exécution des tâches, les machines sont appelées « identiques » et le problème noté (P).
- Si les machines sont différentes par leur vitesse d'exécution et la vitesse de chaque machine est constante et ne dépend pas de l'ensemble des tâches, elles sont dites « uniformes ». (Q).
- Si les vitesses d'exécution des machines dépendent des tâches et sont différentes alors elles sont dites « quelconques » ou différentes (R).

4.3 - 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 [IB 14]**.** Selon les types de produits élaborés, on distingue la production continue et la production discrète. La production continue est caractérisée par 12 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 consommation fabriqués à la chaîne (automobile, la majorité du domaine du textile, machines-outils…) (Figure 3.4).

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. (Voir la notion de la flexibilité en bas)

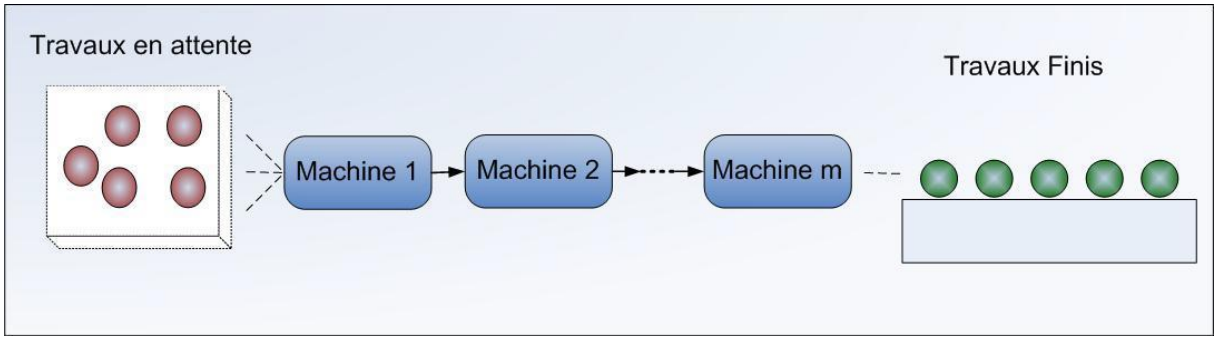

Figure 3.4 La représentation d'un problème d'ordonnancement cheminement unique (flow-shop) Tirée de [1]

4.4 – Les problèmes de job shop

La gamme de fabrication est linéaire et fixée à l'avance, mais peut varier d'un travail à l'autre. Une tâche peut revenir une seconde fois sur la même machine [IB 14]**.** 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 (Figure 3.5).

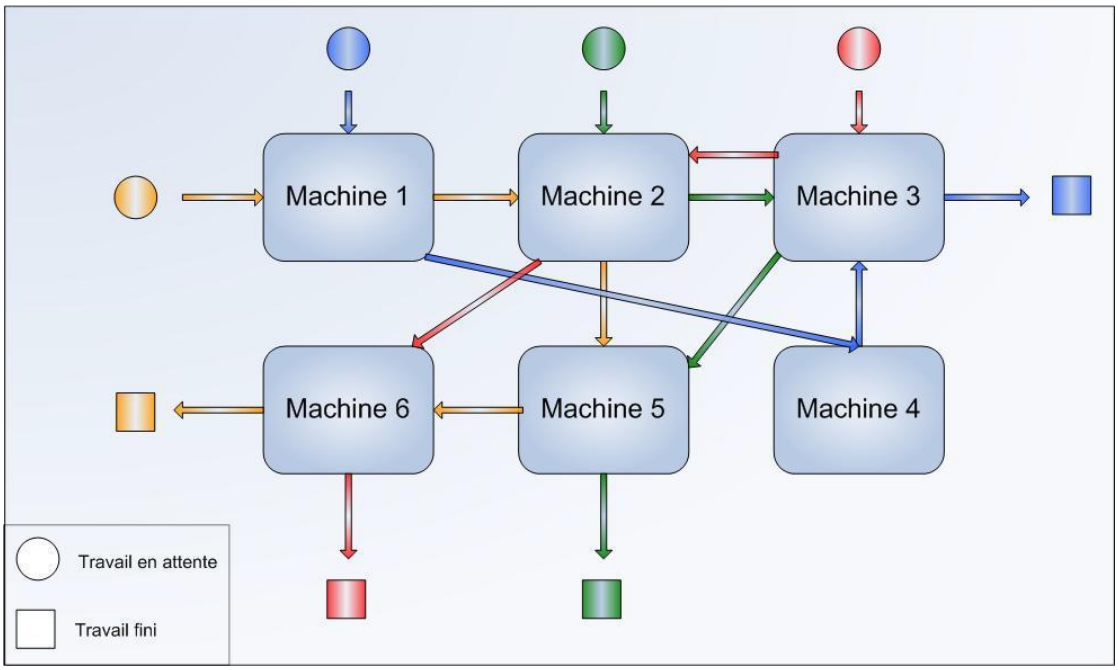

Figure 3.5 La représentation d'un problème d'ordonnancement cheminement multiple (job shop). Tirée de [1]

- Le nombre d'opération n'est pas forcément le même pour tous les jobs.
- Chaque job a son propre ordre de passage sur les machines.

  Parmi les autres caractéristiques d'un problème d'ordonnancement dans un atelier à cheminements multiples.
- Le nombre de solutions possibles est de l'ordre de (n!)m, où n est le nombre de tâches à effectuer et m le nombre de machines. Notons qu'une tâche veut dire la même chose qu'un travail.
- Le problème est considéré parmi les problèmes les plus difficiles à traiter.

4.5 – Les problèmes d'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. [IB 14]

Le problème d'ordonnancement consiste 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).

5 -La notion de la flexibilité

La flexibilité est une mode de gestion de la main d'œuvre qui permet aux entreprises d'adapter 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 cette thèse, sont regroupées en étage ou cellule. Et pour cela on peut distinguer autres types d'atelier

5.1. Flow shop hybride

Le flow shop hybride est une généralisation du flow shop classique au cas où plusieurs machines sont disponibles sur un ou plusieurs étages pour exécuter les différentes tâches du flow shop. 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 [SO 11] (voir la figure 3.6).

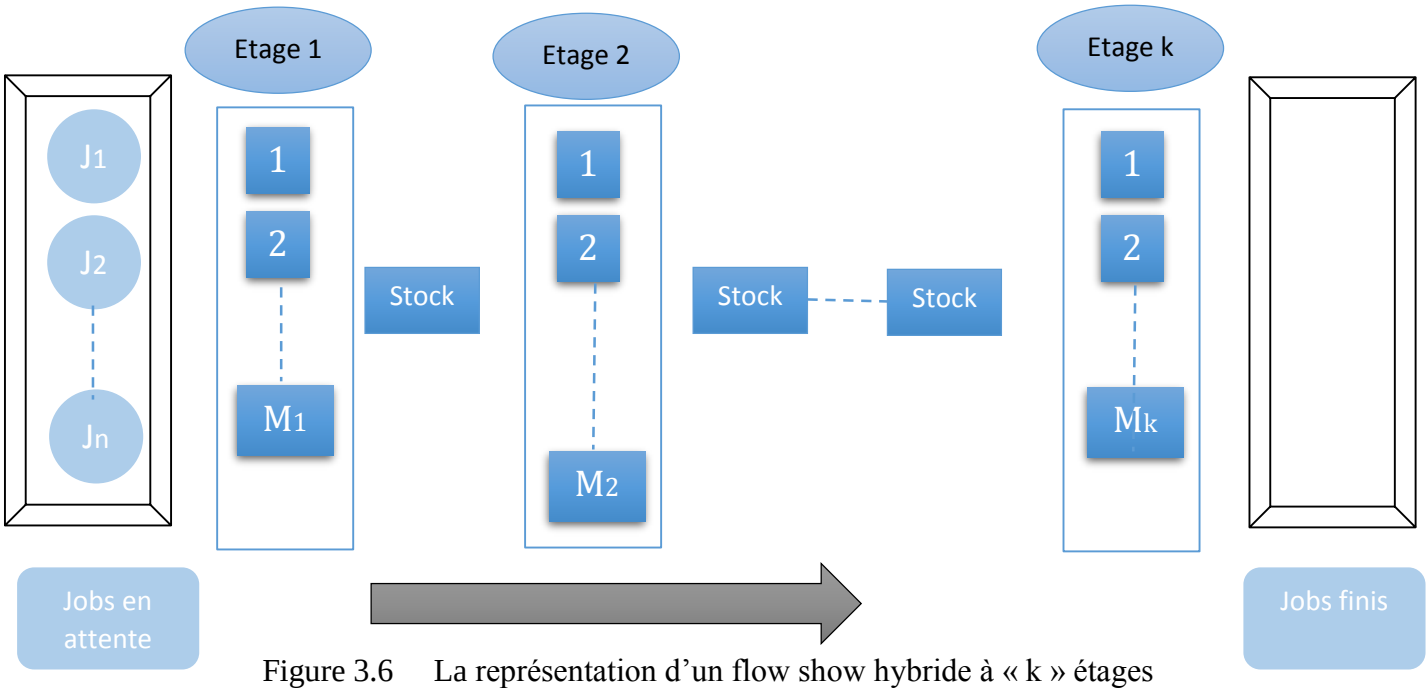

Figure 3.6 La représentation d'un flow show hybride à « k » étages

5.2- Job shop flexible

Le job shop flexible est une extension du modèle job shop classique. Sa particularité essentielle réside dans le fait que plusieurs machines sont potentiellement capables de réaliser

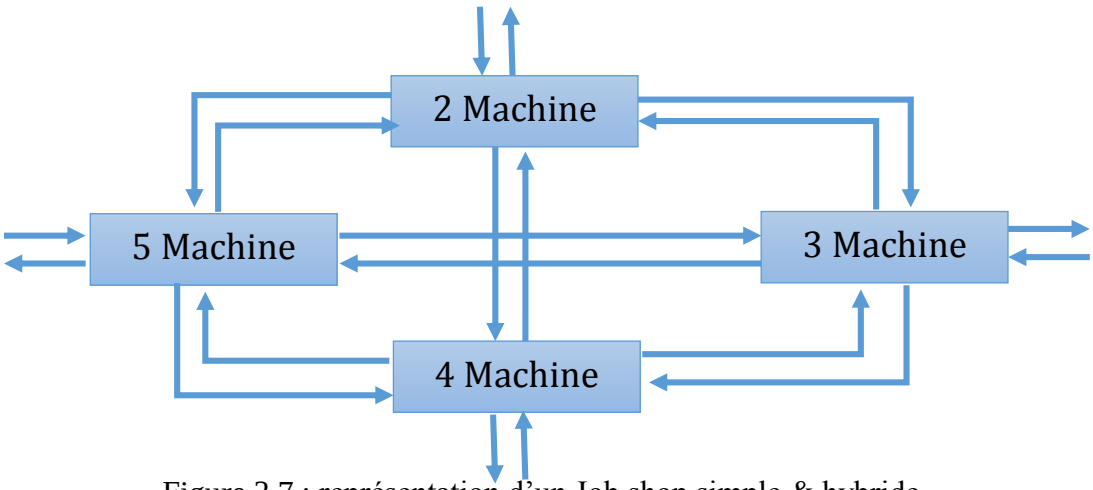

Figure 3.7 : représentation d'un Job shop simple & hybride

un sous ensemble d'opérations. Plus précisément, une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération. [MK 08] (voir la figure 3.7).

6 - La complexité

La complexité de les problèmes d'ordonnancement d'atelier , généralement elle classée dons la class NP-difficile , car la complexité de les algorithme utilisé ne peut pas bornée polynomilement , sauf le problème a une machine elle classé dans la classe P , car est une problème qui peuvent être résolus par des algorithmes polynomiaux

Chapitre 4 Les problèmes d'ordonnancement à machines parallèles

1- Introduction

Dans ce chapitre, nous allons définir c'est quoi les problèmes d'ordonnancement à machines parallèles, et nous savoir les des méthodes de représentation de ces types de problème.

2 – Définition

Le principe des problèmes d'ordonnancement à machines parallèles est Chaque job est constitué d'une seule opération et chaque opération peut être réalisée par n'importe laquelle des machines, disposées en parallèles, mais n'en nécessite qu'une seule. On considère dans toute cette section le critère de minimisation de la durée totale. Dans le cadre de l'informatique on peut ainsi modéliser les « m » processeurs d'une machine parallèle. Ces problèmes sont presque toujours NP-difficiles.

3- Makespan ($\boldsymbol{Cmax}$)

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. [BS 08]

4- Les différents modèles

Pour permettre la classification des problèmes d'ordonnancement, ceux-ci ont été séparés en deux grands modèles suivant les tâches à exécuter

- Les modèles dits statiques : le nombre de tâches à exécuter et leurs dates de disponibilité sont connus à l'avance et ne peuvent être modifiés par la suite,
- Les modèles dynamiques : les tâches arrivent de façon aléatoire. Leurs nombres, ainsi que leurs dates de disponibilité ne sont pas connus à l'avance.

5- Les modèles mathématique des problèmes à machine 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,…..n.
- k : l'indice de la machine, où k=1,…..m .
- r : la position de la tâche dans une machine, où r=1,…..nk.
- r_{jk}: date de début.
- d_{jk}: date de fin la tâche j.
- s_{jk}: 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.
- $Cmax$: makespan.
- n_k: nombre des tâches affectées à la machine k.

Minimiser ($Cmax$, ΣT). (1)

Sous contraintes :

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

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

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

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

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

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

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

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

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

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

X_{jkr} = 0 or 1 k = 1,2,….. mr = 1,2, … n_k j=1,2,…..n (12)

Y_{ij} = 0 or 1 i=1,2,……..n j=1,2,……..n (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 [BE 15].

6 - Représentation des problèmes d'ordonnancement à machines parallèles

Dans les problèmes d'ordonnancement à machines parallèles, on peut afficher les ordonnancements pour nous problèmes avec le diagramme de Gantt, car il est permis de visualiser les séquences des opérations des tâches, en représentant chaque tâche par une ligne sur

laquelle sont visibles, les périodes d'exécution des opération et les périodes ou la tâche est en attente des ressources.

7- Insertion d'une tâche venant aléatoirement

Une approche basée sur le contexte aléatoire, c'est la méthode d'insertion des tâches dans un ordonnancement prévisionnel. Cette méthode consiste à sélectionner et choisir le bon endroit (emplacement) où ces nouvelles tâches doivent être inclues. Dans la littérature on trouve que l'utilisation de la méthode d'insertion de tâches n'été pas limitée au problème d'ordonnancement d'une seule ressource, mais aussi pour plusieurs ressources (comme les ressources humaines dans les problèmes de d'ordonnancement de la maintenance).

8- Deux méthodes de résolution

Pour résoudre ce problème nous avons opté pour deux méthodes célèbres à caractère approché :

8.1- Les algorithmes génétiques

8.1.1- Définition

Les algorithmes génétiques (GA) ont été proposés par JOHN HOLLAND dans les années 70, sont des techniques de recherche et d'optimisation stochastique dérivées de la génétique et des mécanismes de la sélection naturelle et de l'évolution. Leurs champs d'application sont très vastes optimisation de fonctions (cout ou les pertes), planification, et bien d'autres domaines. La raison de ce grand nombre d'application est claire, la simplicité et l'efficacité. [BB 12]

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature sélections, croisements, mutations, etc. Ils appartiennent á la classe des algorithmes évolutionnaires.

[CE 01] On peut dire que l'algorithme génétique est une méthode de programmation qui repose sur le principe de l'évolution pour effectuer la recherche d'une solution adéquate à un problème.

8.1.2- Présentation

Les techniques de recherche et d'optimisation sont en général classées en trois catégories. Énumératives, déterministes et stochastiques. Les AGs font partie de la troisième catégorie et quatre caractéristiques les distinguent des autres techniques d'optimisation.

- Ils utilisent un codage des paramètres et non les paramètres eux-mêmes.
- Ils travaillent sur une population d'individus (ou de solutions).

• Ils n'utilisent que les valeurs de la fonction à optimiser, pas sa dérivée, ou une autre connaissance auxiliaire.

• Ils utilisent des règles de transition probabilistes et non déterministes

8.1.3 – Les étapes de l'algorithme génétique

• Génération de la population initiale

Plusieurs mécanismes de génération de la population initiale sont utilisés dans la littérature. Le problème principal dans cette étape est le choix de la taille de la population. Si la taille de la population est trop grande, le temps de calcul augmente et demande un espace mémoire important. Par contre, une population de taille très petite, la solution obtenue n'est pas satisfaisante. Il faut donc trouver le bon compromis.

• Evaluation (Fitness)

Une fonction d'évaluation est utilisée pour mesurer les performances de chaque individu, qui correspond à une solution donnée du problème à résoudre. Cette fonction permet d'évaluer la capacité d'un individu à survivre en lui affectant un poids appelé fitness. La force de chaque chromosome de la population est calculée an que les plus forts soient retenus dans la phase de sélection, puis modifiées dans la phase de croisement et mutation.

La fonction d'évaluation dépend d'un problème à un autre. Par exemple, dans le cas des problèmes d'ordonnancement, cette fonction peut prendre différents critères (makespan, retard, etc.)

• Codage des individus

Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très utilisés à l'origine.

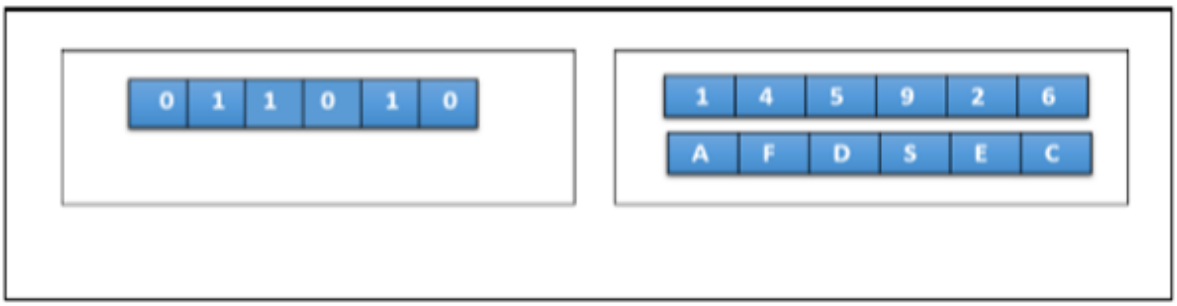

Figure 4.1 La représentions des modules de codage

Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles. (Figure 4.1)

• L'opérateur de sélection

La sélection consiste à choisir des individus qui permettront de générer de nouveaux individus. Plusieurs méthodes existent pour sélectionner des individus destinés à la reproduction. On citera les deux méthodes classiques les plus utilisées.

La sélection par la roulette

La sélection des individus par le système de la roulette s'inspire des roues de loterie.

À chacun des individus de la population est associé un secteur d'une roue. L'angle du secteur étant proportionnel à la qualité de l'individu qu'il représente. Vous tournez la roue et vous obtenez un individu. Les tirages des individus sont ainsi pondérés par leur qualité. Et presque logiquement, les meilleurs individus ont plus de chance d'être croisés et de participer à l'amélioration de notre population. (Figure 4.2) Illustre une population de 5 individus dont les performances sont représentées en roulette.

individu	Fonction objectif	pourcentage
1	30	39%
2	5	7%
3	16	21%
4	10	13%
5	15	15%

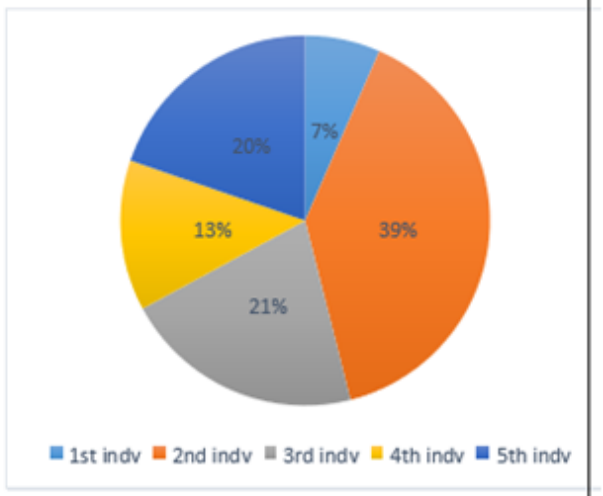

Figure 4.2 La représentions de la sélection par roulette

La sélection par tournoi

Le principe de la sélection par tournoi augmente les chances pour les individus de piètre qualité de participer à l'amélioration de la population. Le principe est très rapide à implémenter.

Un tournoi consiste en une rencontre entre plusieurs individus pris au hasard dans la population. Le vainqueur du tournoi est l'individu de meilleure qualité. Cette méthode est en général satisfaisante. (Voir la Figure 4.3).

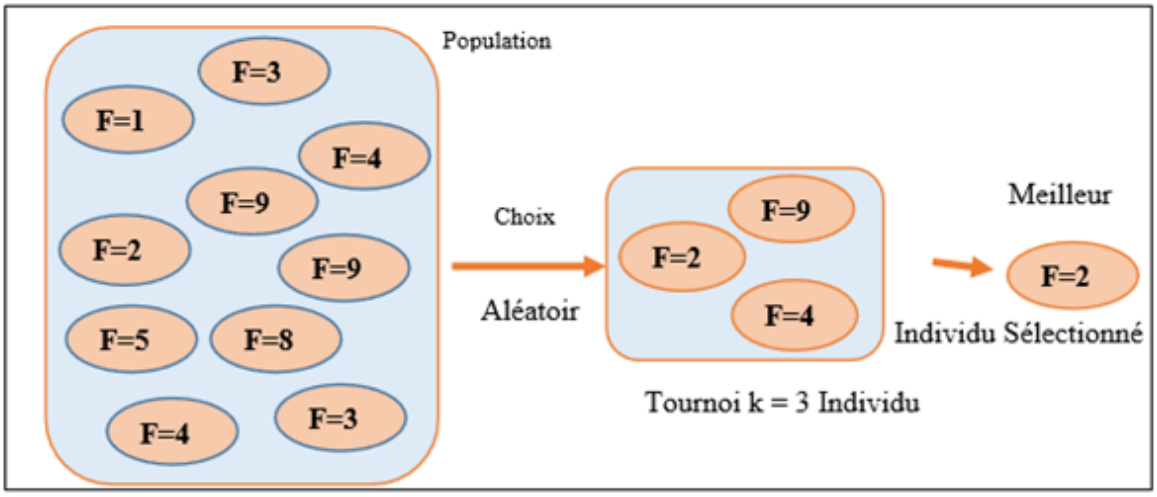

Figure 4.3 La représentions de la sélection par tournois

- **L'opérateur de croisement**

Le croisement est une étape de recombinaison essentielle de l'algorithme évolutionnaire car il permet l'exploration de l'espace de recherche. Une fois la population intermédiaire déterminée, les individus sont aléatoirement répartis en couples. Les chromosomes sont alors copiés et recombinés de façon à former, en général, deux descendants possédant des caractéristiques issues des deux parents. On forme ainsi la génération suivante.

L'opérateur de croisement opère avec une probabilité « pc », fixée selon le problème concerné. Plus ce taux est élevé, plus il y a de nouvelles structures qui apparaissent dans la population. Mais, s'il est trop élevé, les bonnes solutions risquent d'être modifier trop vite par rapport à l'amélioration que peut apporter la sélection. D'autre faible, la recherche risque de stagner, à cause part, si le taux de croisement est très du faible taux d'exploration.

Parmi les méthodes de croisement les plus utilisées on peut souligner trois opérateurs le croisement à un point, le croisement multipoints et le croisement uniforme.

Croisement à un point

Il s'agit de choisir, au hasard, un point de croisement pour chaque couple de chromosomes et d'effectuer un échange des ensembles d'allèles se trouvant de part et d'autre de ce point entre les deux parents. (Figure 4.4).

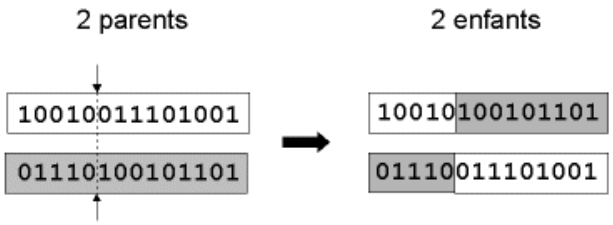

Figure 4.4 La représentation de croissement a un point

On peut étendre ce principe en découpant le chromosome non pas en 2 sous chaînes, mais en 3, 4, etc.

Croisement multipoints

Dans ce cas, plusieurs points de croisement sont sélectionnés et il y a un échange des différentes parties d'allèles cernées par ces points, entre les parents. (Figure 4.5)

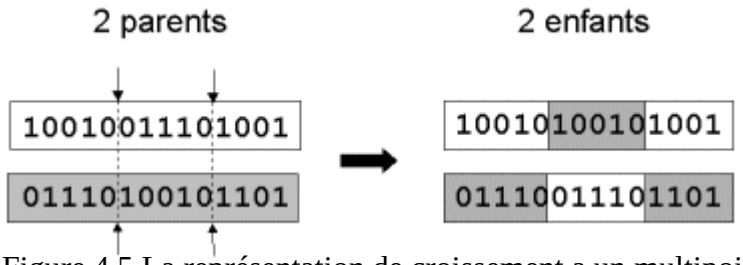

Figure 4.5 La représentation de croissement a un multipoints

Croisement uniforme

Il opère à l'aide d'un masque qui représente les tirages aléatoires, pour décider de la transmission de la valeur de l'allèle à l'un ou l'autre des descendants. Si, à la même position que l'allèle, la valeur du masque est égale à 1, l'allèle du parent 1 passe à celui de l'enfant 1 et l'allèle du parent 2 passe à l'enfant 2. Sinon, c'est l'inverse qui se produit. D'autres types de croisement, plus spécifiques au problème traité, peuvent bien entendu être utilisés dans le cadre d'un algorithme génétique. L'efficacité du croisement dépend souvent de son adaptation au problème (voir la figure 4.6).

Parent 1	1	0	0	0	1	1	1	1	0
Parent 2	1	1	1	1	0	0	0	0	1
Masque	1	0	0	1	1	1	0	0	1	0
Enfant 1	1	1	1	0	1	1	0	0	1	1
Enfant 2	1	0	0	1	0	0	1	1	0	0

Figure 4.6 La représentation de croissement uniforme

• L'opérateur de Mutation

La mutation est définie comme étant la modification aléatoire de la valeur d'un allèle dans un chromosome. Elle joue le rôle de bruit, empêche l'évolution de se figer et garantit que l'optimum global peut être atteint.

Cet opérateur évite donc une convergence prématurée vers les optimums locaux. Il est appliqué avec une probabilité fixée « pm ». Le taux de mutation rend la recherche trop aléatoire s'il est trop élevé. Par ailleurs, s'il est trop faible, la recherche risque de stagner (Figure 4.7).

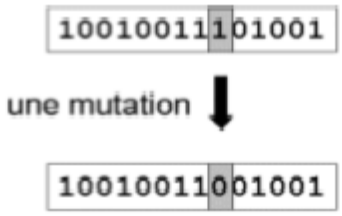

Figure 4.7 La représentation de mutation

Existence d'autres façons d'effectuer des mutations:

- Transposition de deux allèles consécutifs.
- Transposition d'allèles dans un chromosome.
- Inversion de l'ordre des allèles présents entre deux coupes.

8.1.3 - Critères d'arrêt

Le critère d'arrêt indique que la solution est suffisamment approchée de l'optimum.

Plusieurs critères d'arrêt de l'algorithme génétique sont possibles. On peut arrêter l'algorithme après un nombre de générations suffisant pour que l'espace de recherche soit convenablement exploré, ce critère peut s'avérer coûteux en temps de calcul si le nombre d'individus à traiter dans chaque population est important. L'algorithme peut aussi être arrêté lorsque l'évolution de la population est ralentie. On peut aussi envisager d'arrêter l'algorithme lorsque la fonction d'adaptation d'un individu dépasse un seuil fixé au départ. Nous pouvons également faire des combinaisons des critères d'arrêt précédents.

8.2- L'algorithme glouton

Lors de la résolution d'un problème d'optimisation, la construction d'une solution se fait souvent de maniéré séquentielle, l'algorithme faisant à chaque étape un certain nombre de choix. Le principe glouton consiste à faire le choix qui semble le meilleur sur le moment (choix local), sans se préoccuper des conséquences dans l'avenir, et sans revenir en arrière. Un

algorithme glouton est donc un algorithme qui ne se remet jamais en question et qui se dirige le plus rapidement possible vers une solution. Aveuglé par son appétit démesuré, le glouton n'est pas sûr d'arriver à une solution optimale, mais il fournit un résultat rapidement. Même si la solution n'est pas optimale, il n'est pas rare que l'on s'en contente (par exemple pour un problème NP-difficile). On rentre alors dans le monde des algorithmes d'approximation.

Pour que la méthode gloutonne ait une chance de fonctionner, il faut que le choix local aboutisse à un problème similaire plus petit. La méthode gloutonne ressemble ainsi à la programmation dynamique. Mais la différence essentielle est que l'on fait d'abord un choix local et on résout ensuite un problème plus petit (progression descendante). En programmation dynamique, au contraire, on commence par résoudre des sous-problèmes, dont on combine ensuite les résultats (progression ascendante)

Chapitre 5 Conception et réalisation

1-Introduction

Dans ce chapitre, nous allons passer à la conception et l'implémentation d'un problème M-machine en parallèle et on a voir aussi comment ajoute un tache ce problème.

2 –Le langage Java

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995au SunWorld.

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications, mais qui ont l'inconvénient d'être plus lourd à l'exécution (en mémoire et en temps processeur) à cause de sa machine virtuelle. Pour cela, divers plateformes et Framework associés visent à guider, sinon garantir, cette portabilité des applications développées en Java

- **Java FX**

Avec l'apparition de Java 8 en mars 2014, JavaFX devient l'outil de création d'interface graphique officiel de Java, pour toutes les sortes d'application (applications mobiles, applications sur poste de travail, applications Web…), le développement de son prédécesseur Swing étant abandonné (sauf pour les corrections de bogues). Java FX est une pure API Java (le langage de script spécifique qui lui a été un temps associé est maintenant abandonné). Java FX contient des outils très divers, notamment pour les médias audio et vidéo, le graphisme 2D et 3D, la programmation Web, la programmation parallèle, etc

3- Le problème d'une machine parallèle

Le principe de problème d'ordonnancement à machine parallèle est Chaque job est constitué d'une seule opération et chaque opération peut être réalisée par n'importe laquelle des machines, disposées en parallèles, mais n'en nécessite qu'une seule. On considère dans toute

cette section le critère de minimisation de la durée totale, Ces problèmes sont presque toujours NP-difficiles.

4- Les méthodes de résolution de notre problème

J'ai utilisé deux algorithme pour résoudre ce problème, AG et l'algorithme glouton car

4.1- L'algorithme génétique

Nous avons choisi la méthode algorithme génétique pour résoudre notre problème de machine parallèle presque cette méthode il possédé plusieurs avantages comme

- Ils sont adaptables à plusieurs types des problèmes
- Robustes
- Faciles à implémenter
- Faciles à hybrider
- Faciles à paralléliser

4.2- L'algorithme glouton

Est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local, dans l'espoir d'obtenir un résultat optimum global. Par exemple, dans le problème du rendu de monnaie (donner une somme avec le moins possible de pièces), l'algorithme consistant à répéter le choix de la pièce de plus grande valeur qui ne dépasse pas la somme restante est un algorithme glouton. Dans les cas où l'algorithme ne fournit pas systématiquement la solution optimale, il est appelé une heuristique gloutonne. L'illustration ci-contre montre un cas où ce principe est mis en échec.

Exemple Rendu de monnaie

Suivant le système de pièces, l'algorithme glouton est optimal ou pas. Dans le système de pièces européen (en centimes : 1, 2, 5, 10, 20, 50, 100, 200), où l'algorithme glouton donne la somme suivante pour 37 : 20+10+5+2, on peut montrer que l'algorithme glouton donne toujours une solution optimale. Dans le système de pièces (1, 3, 4), l'algorithme glouton n'est pas optimal, comme le montre l'exemple simple suivant. Il donne pour 6 : 4+1+1, alors que 3+3 est optimal. Tirée de [2]

5- Le schéma de réalisation de problème

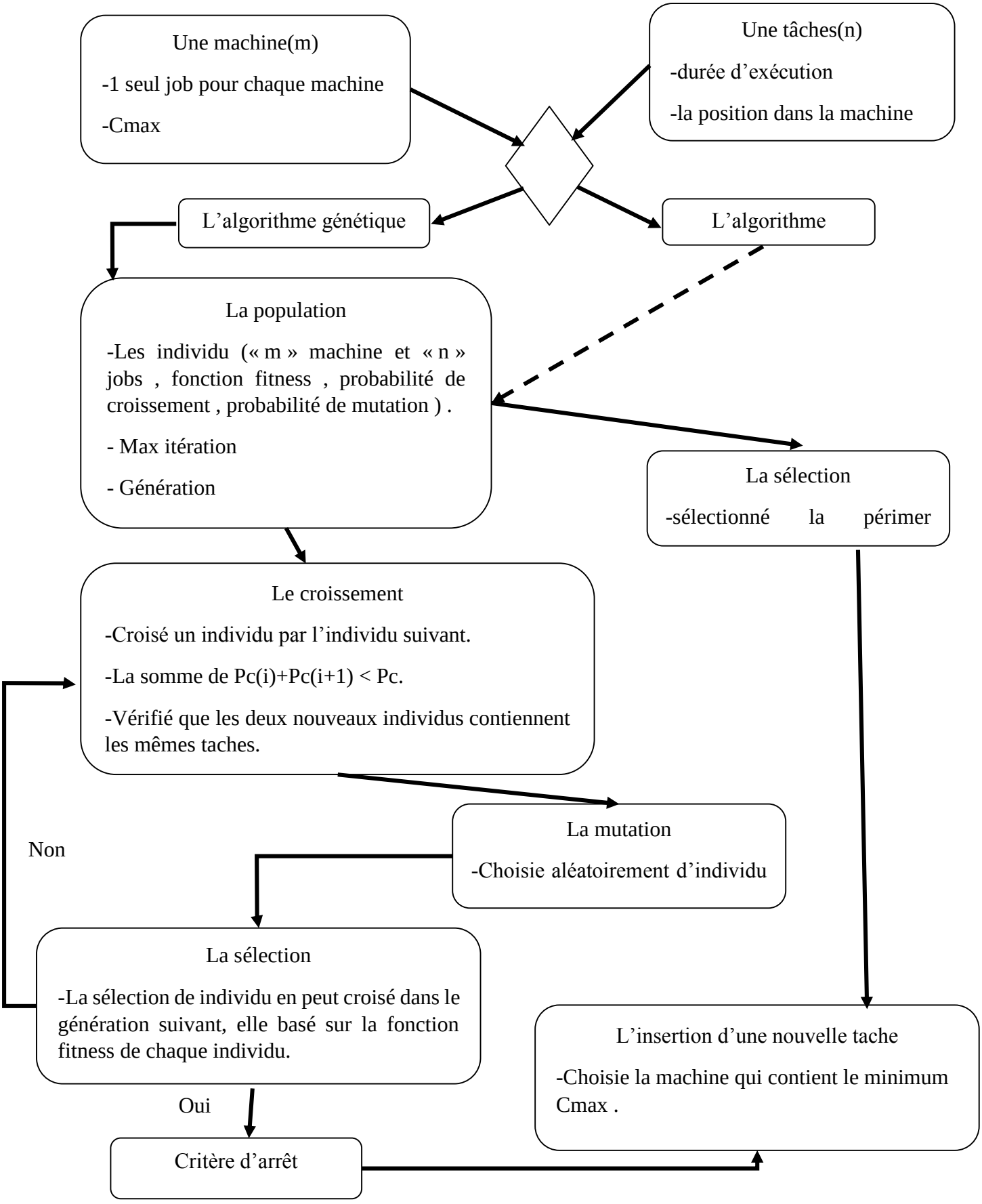

Figure 5.1 Le schéma de réalisation de problème

6- La conception du code

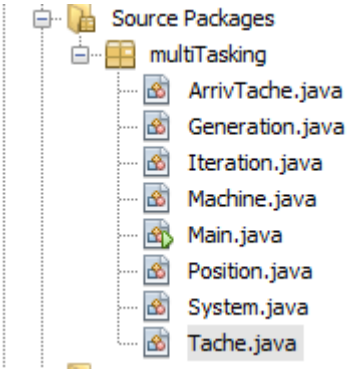

Figure 5.2 Les procédures de code

La figure 5.2 représenté tous les procédures que je créé pour réaliser mon problème dans la figure 5.3 j'ai expliqué comment les procédures travaillant.

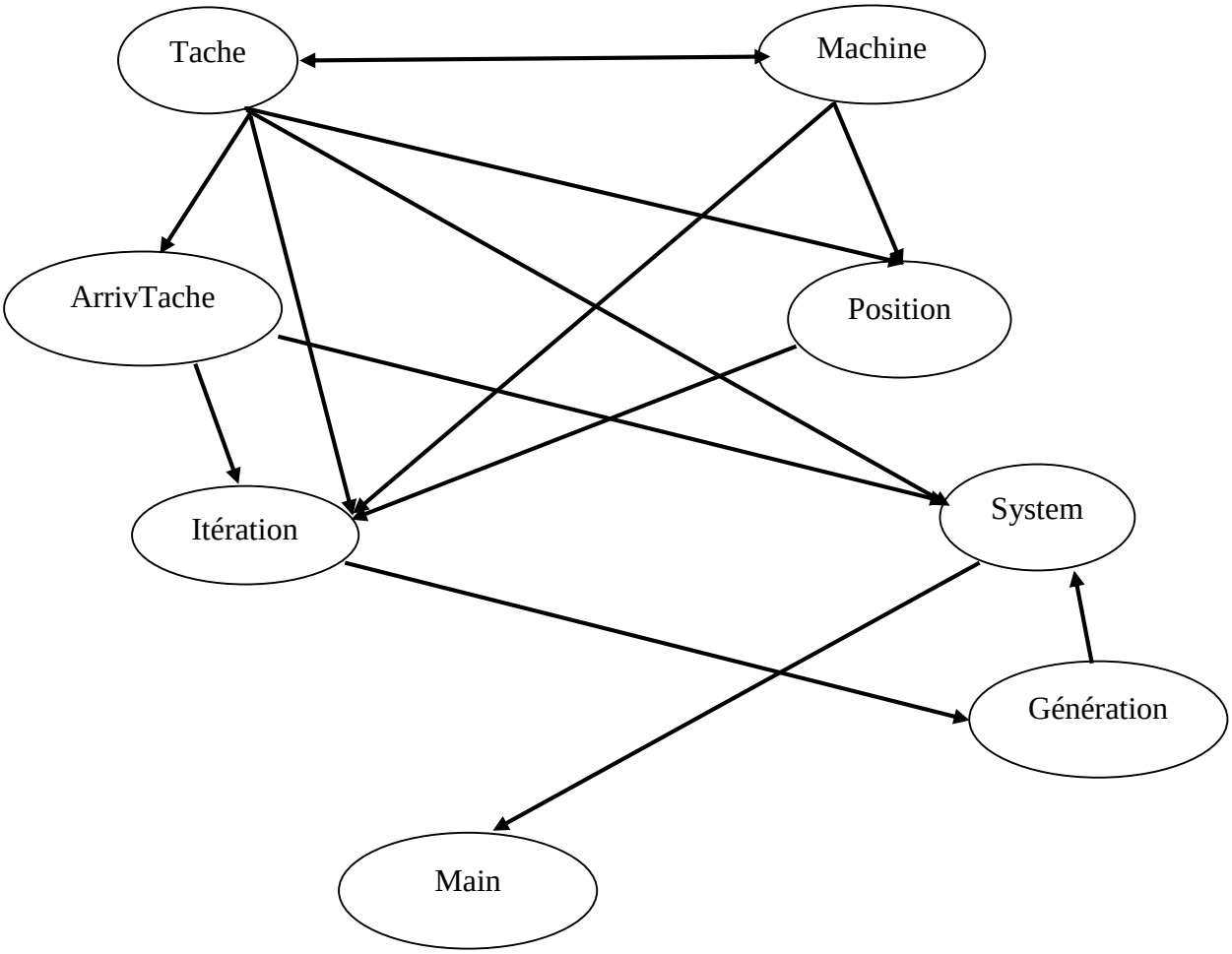

Figure 5.3 La représentante des procédures du code

« Tache » c'est la fonction que je créé les taches.

« Machine » c'est la fonction que je créé les nombres des machines, et je utilisé la fonction des « Tache ».

«ArrivTache » c'est le partie que je créé un nouvelle tâche, et je donné « Tache » comme un donnée.

« Position » c'est la fonction qui je créé pour savoir la position de la tache dans la machine.

« Itération » dans cette fonction j'ai inséré la nouvelle tâche, et calcule les Cmax de chaque machine.

« System » j'ai codé les étapes de les deux algorithme, en plus l'interface graphique (buttons, scrollpane, text field …).

« Génération » c'est la fonction pour créer la critère d'arrête de AG, utilisant la fonction de « Itération ».

« Main » c'est la fonction qui peut exécuter du programme.

7-La réalisation

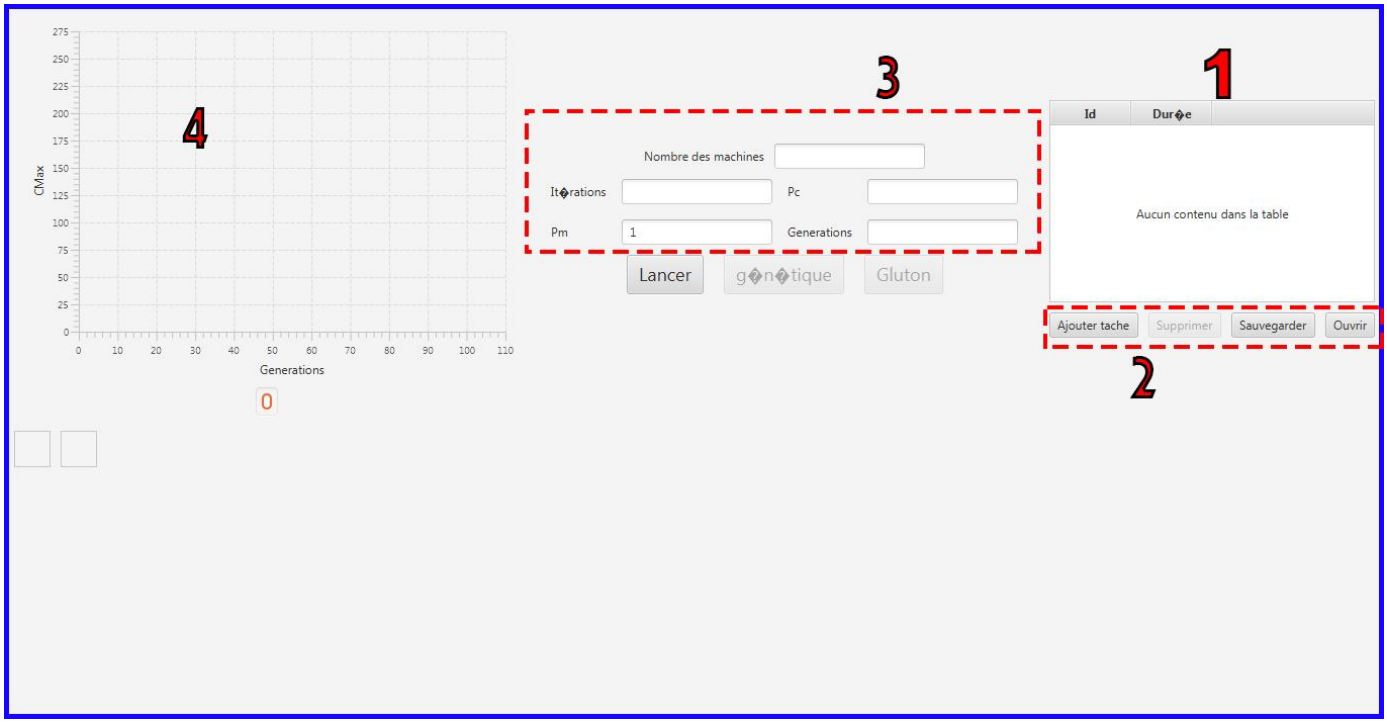

Figure 5.4 L'interface graphique de l'application

De la figure 5.4, j'ai expliqué comment créé mon interface graphique. La premier sélection représente le tableau des taches et sont durée des exécutions , la deuxième sélection représentent les buttons que je utilisée dans mon problème , il y a le bouton « ajouté » , qui peut inséré un tache dans le table dès les taches , il y a aussi un bouton qui « supprimé » un taches de le tableau , le bouton « sauvegarder » sont travail est de enregistre notre tableau des taches dans le ordinateur de forma fichier texte avec un extension (.txt) , si le fichier est déjà sur l'ordinateur , on peut utiliser les bouton ouvrir et on peut sélection le ficher . Pour la troisième partie de l'interface, c'est la partie que je entre les donnée de l'algorithme que je utilise pour affiche la bonne solution de mon problème avec les tableau des taches que je contiens, la quatrième partie c'est une diagramme qui affiché la change de Cmax dans les Générations .

8- L'implémentation de l'algorithme génétique

Dans ce problème j'utilise l'algorithme génétique pour résoudre ce problème et donnée le meilleur ordonnancement pour les taches que je contiens (figure 5.6)

Après ajouté les nombres des taches ou bien sélectionné le fichier texte À partir de l'ordinateur

8.1-Codage

Le chromosome représentant l'individu ou la solution. Une population sera donc un ensemble de permutations de « n » tâches.

8.2- initialisation de population

Après la sélection des taches en peut utiliser, on peut sélectionner les nombres des machines on peut utiliser, dans notre problème on peut utiliser plus de 2 machine (m>2), et l'algorithme peut générer aléatoirement les tache dans les machines. Pour crée une population des individué, et chaque individué contient a un ordonnancement pour les taches (figure 5.5)

Exemple 1

Id	Dur◆e
0	28
1	24
2	45
3	45
4	55
5	21
6	34

Figure 5.5 le tableau des tâches

Nous ajoutons 15 taches dans le tableau des taches avec leur durée d'exécution.

Tache	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Durée	28	24	45	45	55	21	34	22	41	49	17	17	10	48	19

Tab 5.1 Le tableau des tâches

Après on peut enter les données on peut utiliser pour générer la population.

De la (figure 5.5) en peut entrer le nombre de machines pour données et le nombre des itérations qui généré les nombres des chromosomes ou bien individué, et chaque individué contient un ordonnancement pour le problème.

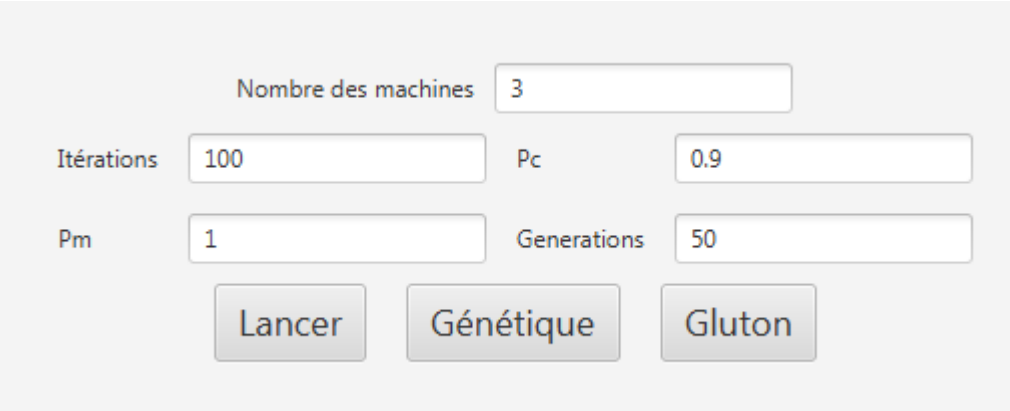

Figure 5.6 L'insertion des données

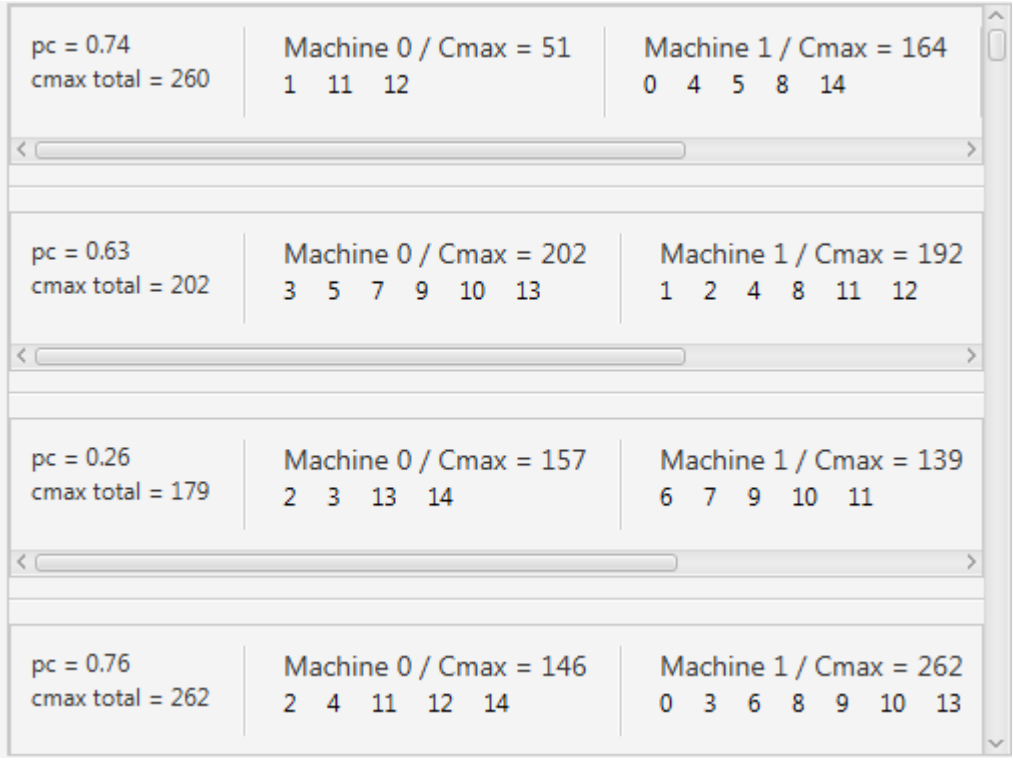

Figure 5.7 La population que on a sélectionnée (100 individu)

8.3-Fonction d'évaluation (fitness)

Le fitness croit inversement avec la fonction objective quand le problème est de minimisation les durées des tâches. La fonction fitness d'un individu « xi », f(xi) est ∑C(xi) où C(xi) la longueur de la durée de l'ordonnancement obtenu pour chaque machine. La probabilité de sélection d'un individu prob(xi)=Cmax(xi)/ ∑C(xi).

8.4- Sélection

A chaque génération ou itération, on cherche à choisir les individus à reproduire. La probabilité de sélectionner un individu est proportionnelle à son fitness. La méthode de sélection utilisée est basée sur le principe de la roulette biaisée individu fort peut être sélectionné plusieurs fois, et un individu faible peut ne pas l'être.

8.5- Croisement

Pour chaque chromosome il contient un probabilité de croisement qui on a créé aléatoire , pour faire le croisement on peut test la probabilité de croisement pour chaque deux chromosome , si « $\sum P_{Ci}+P_{Ci+1}$ < Pc » on faire le croisement , et je utilise le croisement multipoints . Ils préservent la faisabilité des parents. Si les deux parents sont réalisables alors leurs fils le seront aussi.

8.6- Mutation

On a utilisé la mutation qui pour le chromosome d'un individu, transpose un certain nombre de fois, deux tâches consécutives choisies aléatoirement.

8.7-Critère d'arrêt

On a choisi d'arrêter le déroulement de l'algorithme si on atteint un certain nombre de générations. (De notre exemple on a sélectionné 20 génération)

9- L'implémentation de l'algorithme glouton

Un algorithme glouton est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local, dans l'espoir d'obtenir un résultat optimum global (figure 5.7)

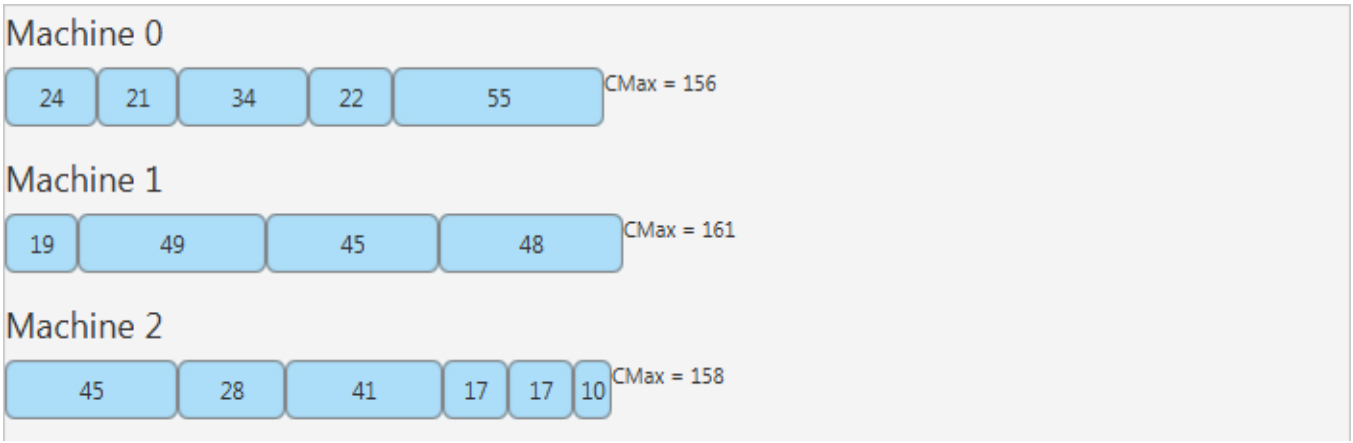

Figure 5.8 Le diagramme de Gantt de l'AG

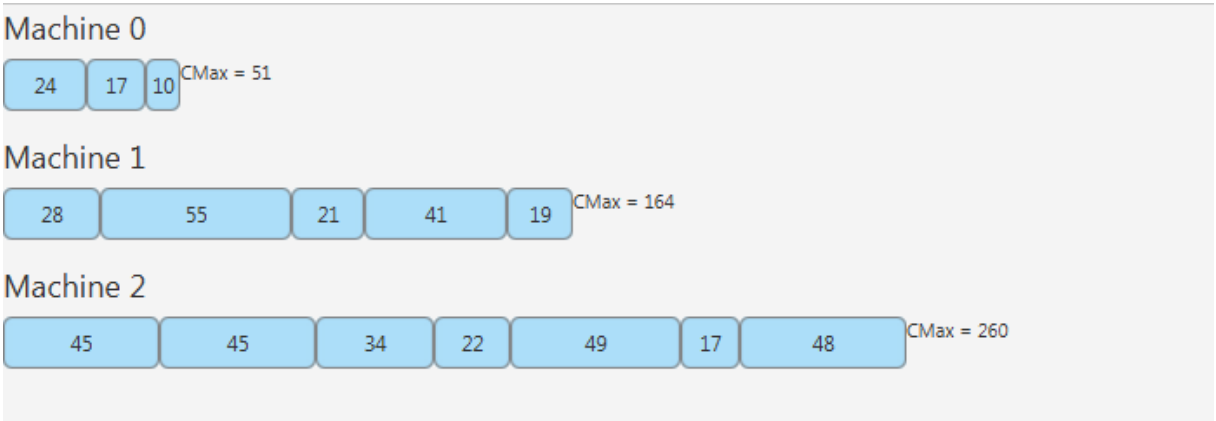

Figure 5.9 Le diagramme l'algorithme glouton

10– L'insertion d'une nouvelle tâche dans l'ordonnancement

L'insertion d'une tâche fait partie du modèle dynamique, car les taches arrivent de façon aléatoire et leurs dates de disponibilité ne sont pas connus à l'avance, donc on gére ce problème d'insertion où le minimum Cmax de l'ordonnancement est dans la machine.

On utilise les données de « exemple 1 » et après on achoisi le meilleur ordonnancement avec l'algorithme génétique, et l'algorithme glouton, et en peut ajouter une tâche et leur durée durée de tâche= 5).

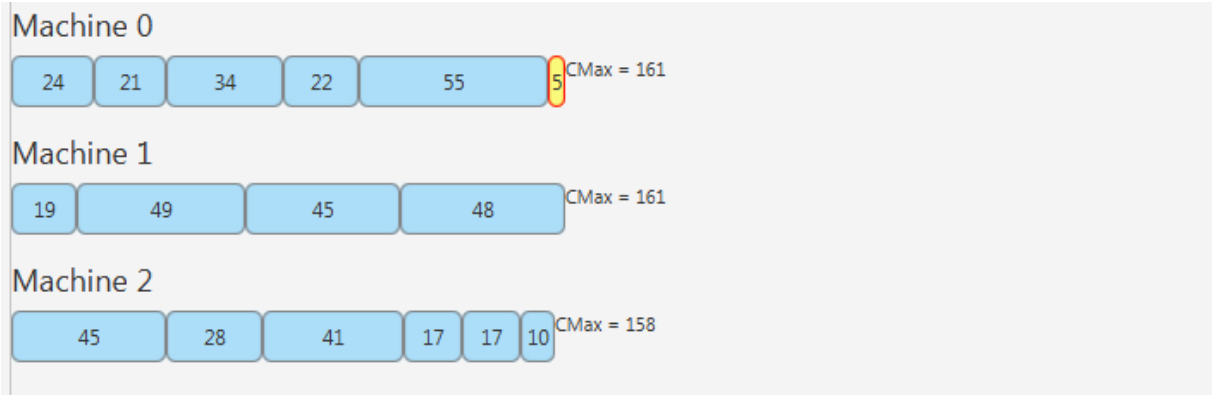

Figure 5.10 Le diagramme de Gantt de AG après l'ajoutons une tâche

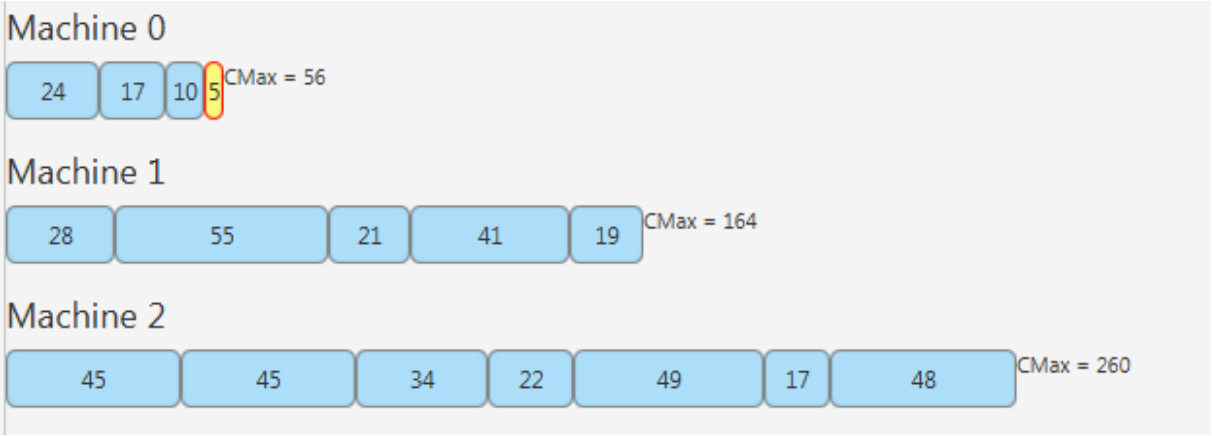

Figure 5.11 Le diagramme de Gantt de Algorithme Glouton après

Tache	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Durée	16	11	46	21	37	49	47	47	21	16	17	41	42	28	49	5

Tab 5.2 : le tableau des tâches après l'ajoutons une tâche

11 - Comparaison

Considérons le cas du problème avec deux algorithmes, l'algorithme local qui donne une solution local est l'algorithme glouton (voir la figure 5.4), est l'algorithme global qui donne une solution globale est l'algorithme génétique (voir la figure 5.3). On a vu que l'ordonnancement de l'algorithme génétique meilleur que l'algorithme glouton car, il donne Cmax minimum part à port l'algorithme glouton. Après l'ajout d'un nouveaux tâche on peut choisir la machine qui contient le minimum Cmax (min Cmax) est on peut ajouter cette tâche à la fin de

l'ordonnancement de cette machine, (voir la figure 5.6 et la figure 5.7), on peut voir que le Cmax ne change pas dans les deux algorithmes.

Conclusion Générale

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 M-Machines identiques en parallèle et comment insérer une opération dans le problème et en vue de minimiser le Makespan (Cmax).

Dans ce mémoire, nous nous sommes, tout d'abord, intéressés à l'étude des problèmes de l'optimisation combinatoire avec leurs méthodes de résolution pour trouver de bonnes solutions aux problèmes correspondants, à savoir des méthodes exactes (programmation linéaire, séparation et évaluation) et des méthodes approchées (recuit simulé, recherche tabou, algorithmes génétiques). Nous sommes penchés sur les problèmes d'ordonnancement dans sa globalité, avec ses différentes composantes (tâches, ressources, contraintes, critères), et aussi avec les différents types d'ateliers étudiés.

Deux méthodes de résolution ont été suggérées basées sur les algorithmes génétiques et l'algorithme glouton, et qui ont été proposée pour la résolution de problème d'insertion d'une opération dans un problème d'ordonnancement a machines parallèles.

La comparaison des deux méthodes a montré la supériorité de la solution globale sur la solution locale, c-à-d la supériorité des algorithmes génétiques sur l'algorithme glouton. Malheureusement, le facteur temps ne nous a pas permis de faire une étude statistique pour prouver cela. C'est l'une des projets qui restent à faire.

Bibliographie

[AL 10] Abdesslam Layeb. « Utilisation des Approches d'Optimisation Combinatoire pour la Vérification des Applications Temps Réel,Thèse de Doctorat, Université Mentouri de Constantine ».(2010)

[AK 12] Asma Karray. « Contribution à l'ordonnancement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides,thése de doctorat ,Ecole centrale de Lille Universite de Tunis El Mmanar Ecole Nationale D'ingenierus de TUNIS ». (2012)

[BB 08] Boris Bontoux. « Techniques hybrides de recherche exacte et approchée : application à des problèmes de transport,thése de doctorat , l'Université d'Avignon et des Pays de Vaucluse ». (2008)

[BB 12] Bouabdallah Benamara. « Application des algorithmes génétiques Au dispatching économique et environnemental, thése de master,Université de biskra ». (2012)

[BE 15] Bounif M. E., Optimisation à base de simulation pour le développement des syst-mes décisionnels, Thèse Doctorat 3éme cycle en informatique, université de M'sila, 2014/2015

[BS 08] 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.

[CE 01] C. Eyquem et A. Montaut. « Les Algorithmes Génétiques ». (2001)

[HF 16] Hamdaoui Fawzia. « Contribution à la resolution du probléme du sac à dos multidimensionnel n à choix multiple,thése de magister,Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf ».(2016)

[HT 07] Hamdy A. T. « Operations Research: An Introduction, 8th ed., Prentice Hall ». (2007)

[IMB 14] Imen Medhbi Brinis, « Ordonnancement d'ateliers de traitements de surfaces pour une production mono-robot/multi-produits : Résolution et étude de la robustesse » Thèse en vue de l'obtention du garde Docteur en Spécialité « Automatique, génie informatique, traitement du signal et images » le 16 mai 2014

[JH 06] Jeffrey W. Herrman, & al. Handbaook of Production Scheduling, Springer NY, 2006.

[KB17] Kacem B. « Ordonnancement multicritère des job-Shop flexibles : formulation, bornes inférieures et approche évolutionniste coopérative. Thèse de Doctorat, Université des Sciences et Techniques de Lille1 ». (2017)

[LA 01] Letouzey A. ordonnancement interactif basé sur des indicateurs. 2001.

[MA 05] : Mohamed Ali Aloulou, « Introduction aux problèmes d'ordonnancement », le 28 novembre 2005, LAMSADE, Université Paris Dauphine.

[MA 11] 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.

[MK 08] Mebarek Kebabla, « Utilisation des stratégies méta-heuristiques pour l'ordonnancement des ateliers de type job shop » Mémoire en vue de l'obtention du Diplôme de Magister, présenté au laboratoire d'Automatique et Productique LAP. le 02 juillet 2008.

[MR 08] Mostepha, R : Résolution de problèmes d'optimisation combinatoire par systèmes artificiles auto-organisés. Thèse de magister, Université Mentouri de Constantine ,2008.

[PM 05] Palpant M. « Recherche exacte et approchée en optimisation combinatoire : schémas d'intégration et applications. Thèse de Doctorat, Université d'Avignon ». (2005)

[SD 14] S. Le Digabel. « Introduction aux métaheuristiques , Ecole Polytechnique de Montréal ». (2014)

[SO 11] Samia Ourari, « L'ordonnancement déterministe à l'ordonnancement distribué sous incertitude », le 28 janvier 2011, Thèse en vue de l'obtention du garde de docteur de l'Université de TOULOUSE.

[TV 01] Thomas Vallée et Murat Yıldızoglu« Présentation des algorithmes génétiques et de leurs applications en Economie » Université Montesquieu Bordeaux IV Avenue Léon Duguit 2001.

[WL 05] Wafaa LABBI et Mourad BOUDHAR « Méta heuristiques pour un problème d'ordonnancement sous contraintes de préparation » Laboratoire RECITS, Faculté de Mathématiques, USTHB BP 32 El-Alia, BEZ, Alger, Algérie 2015,

Les sites web

[1] https://fr.wikipedia.org/wiki/Ordonnancement_d%27atelier

[2] http://sdz.tdct.org/sdz/les-algorithmes-gloutons.html

الملخص

المشكلة التي تتناولها هذه المذكرة اضافة عملية بعد 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 ،و الخوارزمية الجشعة وبرمجناها لحل المشكلة.
الكلمات الرئيسية: الجدولة, الات متوازية متطابقة, وقت الاتمام الاكبر, الخوارزمية الجينية خوارزمية الجشعة

Résumé

Le problème traité dans ce mémoire concerne l'insertion d'un novelle opération dans l'ordonnancement à machines parallèles en vue de minimiser le makespan. Les opérations peuvent être exécutées en parallèle sur plusieurs machines. Une méthode a été suggérée basée sur les algorithmes génétiques, et l'algorithme glouton ont été, ensuite, proposée pour la résolution de problèmes.
Mots clés : Ordonnancement, machines parallèles identiques, le makespan, algorithm génétique, algorithm glouton.

Abstract

The problem addressed in this thesis concerns is to add a new job to the scheduling of operations on identical parallel machines in order to minimize the Makespan. The operations can be performed in parallel on multiple machines. A method was suggested based on genetic algorithms and glouton algorithm, then it was used for solving the problem.
Key words: Scheduling, identical parallel machines, the makespan, genetic algorithme , Greedy algorithme