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 : **Marouf Meriem F.Z**

SUJET

Les tâches indépendantes dans les problèmes d'ordonnancement à machines parallèles

Soutenu publiquement le : / /2020 devant le jury composé de :

Dr.	**Université de M'sila Président.**
DR.Nasser Eddine MOUHOUB	**Université de M'sila Rapporteur**
Dr.	**Université de M'sila Examinateur**

Promotion : 2019 /2020

Remerciements

Au début de ce modeste travail, je remercie le bon dieu de nous avoir orientés et guidés par sa merci et sa miséricorde pour réussir à élaborer ce travail.

Nous remercions en particulier Mr. Mouhoub Nasser Eddine, pour l'honneur qu'il m'a fait de bien vouloir m'encadrer, et pour les conseils donnés lors de la réalisation de ce travail.

J'adresse mes remerciements aux membres de jury pour avoir accepté de me prêter de leur attention et évaluer ce travail.

Je tiens remercier tous les enseignants de département d'informatique d'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

Avec les sentiments de gratitude les plus sincères Je dédie ce modeste travail :

À mon père M.KAMEL qu'il est vivent à mon cœur et jamais oublier pour son amour, sa patience, ses sacrifices, ses encouragements permanents pendant mes études.

À ma chère maman H.ZINEB qui m'est la plus chère dans ce monde, qui est toujours éclairé mon chemin depuis le jour où j'ai vu la lumière par ses prières, son amour, ses sacrifices, ses encouragement par l'éducation qu'elle ma inculquée et tous les peines qu'elle s'est donnée pour moi pour ma réussite.

Je dédie aussi ce travail à :

- *Mon frère youcef , mes sœurs(basma, salsabil,insaf).*
- *Mes sœurs qui je connaîs dans ma vie.*
- *Mes oncles, mes tantes et leur famille.*
- *Tous mes cousins et cousines.*
- *Tous mes fidèles amis et mes collègues.*
- *A mon encadreur Mr. Mouhoub Nacer Eddin.*

Table des matières

Introduction Générale 1

Chapitre 1 Problème du L'optimisation combinatoire 3

1- **Introduction 3**

2- **Définition 3**

3- Complexité 3

3.1 La complexité algorithmique 4

3.2 La complexité problématique 4

4- Les méthodes de résolution 5

4.1 Les méthodes exactes 6

4.2Les méthodes approchées 9

5- Classification des méthodes métaheuristiques 10

5.1 Les métaheuristiques à solution unique 11

5.2 Les métaheuristiques à population de solutions 12

Chapitre 02 Les problèmes d'ordonnancements 13

1- Introduction 13

2- Définition 13

3- Les éléments d'un problème d'ordonnancement 13

4- Les objectifs de l'ordonnancement 16

5- La typologie des problèmes d'ordonnancement 17

6- Résolution d'un Problème d'ordonnancement 18

7- Représentation des problèmes d'ordonnancement 18

➢*Le diagramme de Gantt 19*

➢*Graphe potentiel-tâches 20*

➢*Method PERT (Program Evaluation and Research Task) 21*

Chapitre 3 Les problèmes d'ordonnancement d'atelier 22

.1 Introduction 22
2. Définition 22
3. les types des Problémes 20
3.1Modèles à une opération 23
3.2Modèles à plusieurs opérations 25
4. Relations entre les types 28
5. La complexité 30

Chapitre 4 Les tâches indépendantes dans les problèmes d'ordonnancement à machines parallèles 31
1- Introduction 28
2 – Définition (les problèmes d'ordonnancement à machines parallèles) 31
3- Tâches indépendantes (Machines identiques) 31
4- Le problème de Machine identique en parallèle 35
5- Makespan ($Cmax$) 37
6- Difficulté du problème $P|prec|C_{max}$ 37
7- Représentation d un problème d'ordonnancement à machines parallèles 39
8- L'algorithme glouton 39
9- Les algorithmes génétiques 41

Chapitre 5 IMPLEMENTATION DE PROBLEME 48
1- Introduction 45
2- Langage et environnement de développement 48
3- Le problème d'une machine parallèle et les méthodes de résolution de notre problème 49
4 - La conception 50
5- La réalisation 52
6- Conception et résultats de l'AG 54
7- Comparaison 60

Conclusion Générale 61

Liste des Figures

Figure 1.1 : Classification des méthodes d'optimisation combinatoire

Figure 1.2 : Algorithme générale de Branch and Bound

Figure 1.3 : Programmation dynamique

Figure 1.4 : Classification des métaheuristiques.

Figure 2.1 : La caractéristique d'une tache.

Figure 2.2 :la représentation des types d'ordonnancement.

Figure 2.3 Le diagramme de Gantt

Figure 2.4 Graphe Potentiel-Tâches d'un ordonnancement

Figure 3.1 : Les types des machines

Figure 3.2 : Modèle à machine unique.

Figure 3.3 : Modèle à machines parallèle.

Figure 3.4 : Modèle flow-shop.

Figure 3.5 : Modèle job-shop.

Figure 3.6 : Modèle open-shop.

Figure 3.7 : La représentation d'un flow show hybride à « k » étages.

Figure 3.8 : La représentation d'un job shop hybride.

Figure 3.9: Relations entre les différentes organisations.

Figure 4.1 : – Earliest Due Date — Lmax = 0.1.

Figure 4.2 : – Un meilleur ordonnancement — Lmax = −0.4.

Figure 4.3 : Illustration de la réduction Partition α P2| |Cmax.

Figure 4.4 : Diagramme de Gantt.

Figure 4.5 : Exemples de codage par valeurs.

Figure 4.6 : Schéma d'un algorithme génétique .

Figure 4.7 : La représentation de croissement a un point.

Figure 4.8 : La représentation de croissement a un multipoints

Figure 4.9 : La représentation de croissement uniforme.

Figure 4.10 : La mutation en codage binaire.

Figure 5.1 Le schéma de réalisation de problème

Figure 5.2 :Fenêtre de départ.

Figure 5.3 : L'interface graphique de l'application.

Figure 5.4: les données de l'algorithme.

Figure 5.5 : tableau des taches.

Figure 5.6 : L'insertion des données de l'algorithme et le tableau des tâches

Figure 5.7 : La population.

Figure 5.8 : Le diagramme de Gantt de l'AG.

Figure 5.9 : Le diagramme de Gantt de l'AG.

.

Introduction Générale

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 présente les notions de base relatives aux problèmes d'optimisation combinatoire. Il rappelle aussi quelques concepts sur la théorie de la complexité, et donne un aperçu des éléments de la résolution des problèmes d'ordonnancement, leurs notations et leurs classifications .

Le deuxième chapitre de cette thèse, nous études les problèmes d'ordonnancement, voyons les différent domaines, puis nous verrons les différents éléments , la classification et la représentation du problème d'ordonnancement.

Le troisième chapitre nous allons expliquer les problèmes d'atelier et présent leur schéma de classification, expliqué les types de problèmes d'ordonnancement d'ateliers et la relation entre eux ,et leurs complexité.

Ensuite en le quatrième chapitre on explique seulement sur les problèmes de Machine parallèle et nous savoir les des méthodes de représentation de ces types de problème , suivi d'une description du problème à étudier.

Le cinquième et dernier chapitre portera sur l'implémentation de notre application ainsi que l'élaboration des tests expérimentaux.

Nous finirons notre travail par une conclusion générale qui présente un résumé de ce qu'a été étudié dans ce mémoire, les résultats obtenus et le travail qui reste à faire pour l'accomplissement de cette étude.

Chapitre 1 Problème du L'optimisation combinatoire

3- Introduction

L'optimisation combinatoire définit un cadre formel pour de nombreux problèmes de de l'industrie, de la finance ou de la vie quotidienne. Les problèmes d'optimisation combinatoire sont habituellement définis comme une problématique de choix d'une meilleure alternative dans un ensemble très grand mais fini d'alternatives.

Dans ce chapitre , nous Savoir la définition d'un problème d'optimisation combinatoire, quelques concepts sur la théorie de la complexité et aussi nous présentons les méthodes de résolution d'un problème d'ordonnancement.

2 - Définition

Un problème d'optimisation combinatoire consiste à trouver la *meilleure* solution dans un ensemble discret dit ensemble des solutions réalisables. En général, cet ensemble est fini mais compte un très grand nombre d'éléments, et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

Pour définir la notion de *meilleure solution*, une fonction, dite *fonction objectif*, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution (ou *solution optimale*) est celle qui minimise ou maximise la fonction objectif. Clairement, un problème d'optimisation combinatoire peut avoir plusieurs solutions optimales.

1. 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. [15].

La complexité d'ordonnancement peut être divisée en deux grandes catégories : algorithmique et problématique.

3.1 La complexité algorithmique

L'objectif de la théorie de complexité est d'analyser les cout de résolution surtout en termes de temps de calcul, elle vise aussi a classifier les problèmes en plusieurs niveaux de difficultés. Une étude a prouvé que les problèmes d'ordonnancement sot des problèmes difficiles.

En général, la complexité algorithmique se mesure par rapport a deux paramètres :

- Le temps alloué pour l'exécution de l'algorithme : il est relatif au nombre d'instruction à exécuter ainsi qu'à la taille des données manipulés.
- Espace mémoire requis : associé à la taille d'instance d'un problème donné. [15].

3.2 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 aux critères retenus.

Un problème de décision comprend deux parties : une partie données 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 solution. Donc , la résolution d'un problème de recherche consiste a trouver pour chaque ensemble de données D des solutions S asso- ciées . Un problème d'optimisation est un problème de recherche en associant à chaque solution une valeur quantitative. à chaque problème d'optimisation, on peut associer un problème de décision (par exemple d'exclusion u l'inclusion d'une solution dans les futures génération

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 complexité permet de classer les problèmes en deux classes P et NP. La classe P regroupe les problème qui peuvent être résolus par les 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 est résolu au mieux , en un temps exponentiel .

- Un problème d'optimisation est dit NP-DIFFICILE, si le problème de décision associé est NP-COMPLET. [15].
-

2. Les méthodes de résolution

Comme on l'a rappelé dans le paragraphe sur la complexité, la plupart des problèmes d'ordonnancement sont NP-difficiles. On ne peut donc pas espérer trouver un ordonnancement optimal en un temps raisonnable pour des problèmes de taille industrielle. Les méthodes utilisées sont donc des méthodes approchées.

Pour des problèmes de petite taille, nous pouvons obtenir une solution exacte .Une méthode exacte peut servir pour résoudre des sous-problèmes d'un problème de grande taille de manière optim ale.

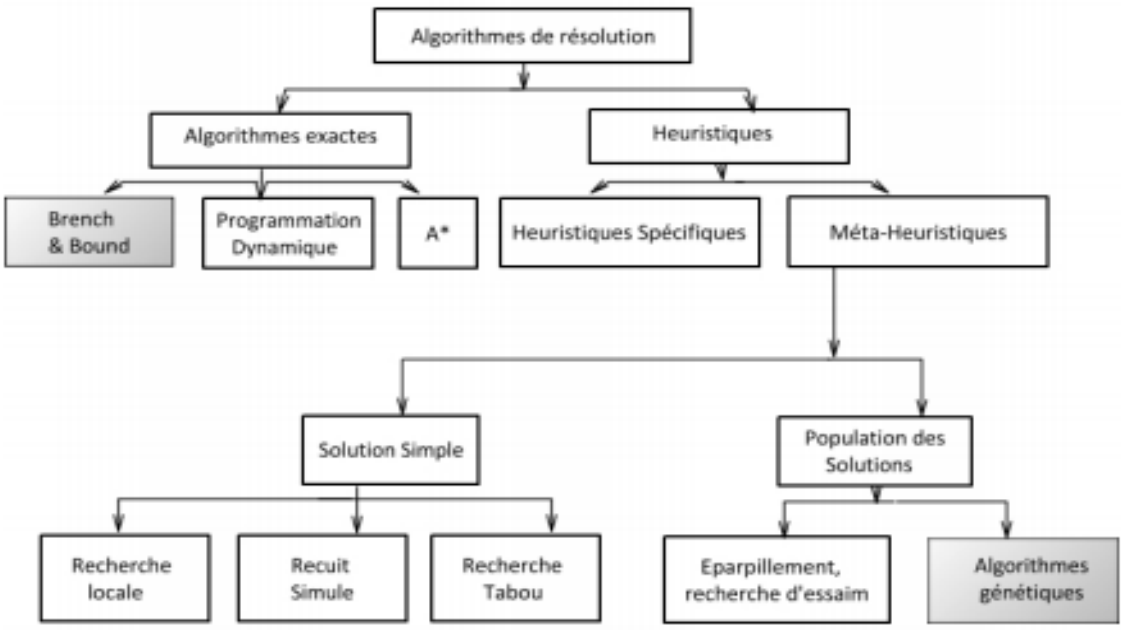


Figure 1.1 : Classification des méthodes d'optimisation combinatoire

4.1 Les méthodes exactes

Elles sont basées sur une recherche exhaustive et en évaluant chaque solution S de X, jusqu'à trouver la solution optimale globale, comme : Brunch and Bound et la programmation dynamique. Les méthodes de cette classe sont vite devenues inutilisables pour des instances de grandes tailles.[A2].

4.1.1 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 effectue une recherche complète de l'espace des solutions d'un problème donné, pour trouver 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. [6].

Algorithme générale

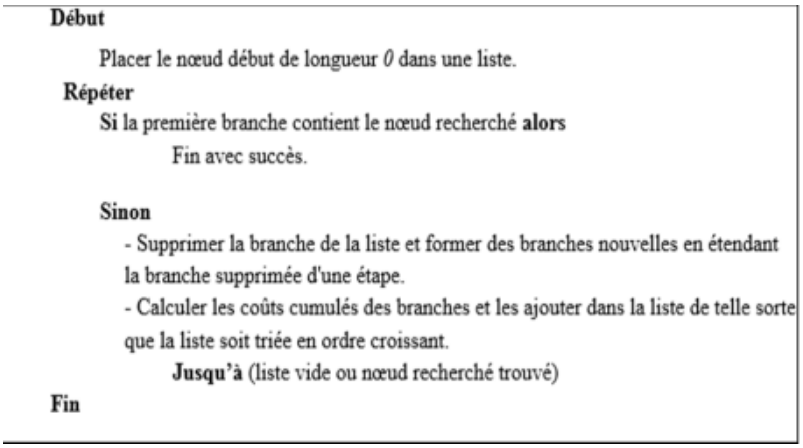


Figure 1.2 : Algorithme générale de Branch and Bound

4.1.2 La programmation Dynamique

La programmation dynamique consiste à placer le problème dans une famille de problèmes de même nature mais de difficulté différente, puis à trouver une relation de récurrence liant les solutions optimales de ces problèmes.

Elle est basée sur le principe de Richard BELLMAN (1949) « Toute politique optimale ne peut être formée que de sous politiques optimales ».

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. [1]

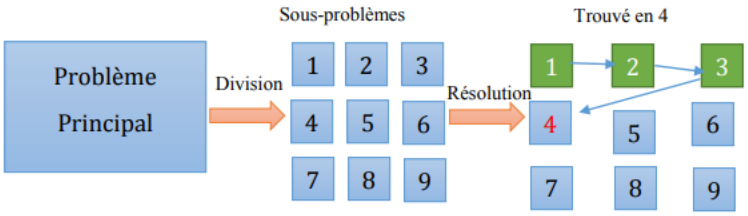


Figure 1.3 : Programmation dynamique

4.2.3 La programmation linéaire

C'est l'une des techniques classiques de recherche opérationnelle. Elle repose sur la méthode du simplexe et les algorithmes de points intérieurs de Karmarkar. Elle consiste à minimiser une fonction coût en respectant des contraintes, le critère et les contraintes étant des fonctions linéaires des variables du problème.

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^TX» 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. [9]

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. [2]

4.2 Les méthodes approchées

Cette section comporte une présentation de quelques méthodes approchées. Celles-ci sont connues sous le nom d'heuristiques ou de métaheuristiques. Ces méthodes sont utilisées pour résoudre des problèmes de grandes tailles qui ne peuvent pas l'être de manière exacte en des temps de calcul acceptables en sachant néanmoins qu'elles ne garantissent pas l'optimalité de la solution trouvée.

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. [3].

4.2.2 Les métaheuristiques

Les méthodes dites métaheuristiques sont des méthodes générales, des heuristiques polyvalentes applicables sur une grande gamme de problèmes. Elles peuvent construire une alternative aux méthodes heuristiques lorsqu'on ne connaît pas l'heuristique spécifique à un problème donné. [10].

Les propriétés fondamentales des métaheuristiques

Les métaheuristiques sont des stratégies qui permettent de guider la recherche d'une solution optimale.

- Le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer dissolutions (presque) optimales.

- Les techniques qui constituent des algorithmes de type métaheuristiques vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.

- Les métaheuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité.

- Les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.

- Les concepts de base des métaheuristiques peuvent être décrit de manière abstraite, sans faire appel à un problème spécifique.

- Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.

- Les métaheuristiques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche. [7]

3. Classification des méthodes métaheuristiques

On peut regrouper les métaheuristiques en deux grandes classes : les métaheuristiques à solution unique (c.à.d. évoluant avec une seule solution) et celles à solutions multiples ou population de solutions. Les méthodes d'optimisation à population de solutions améliorent, au fûr et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

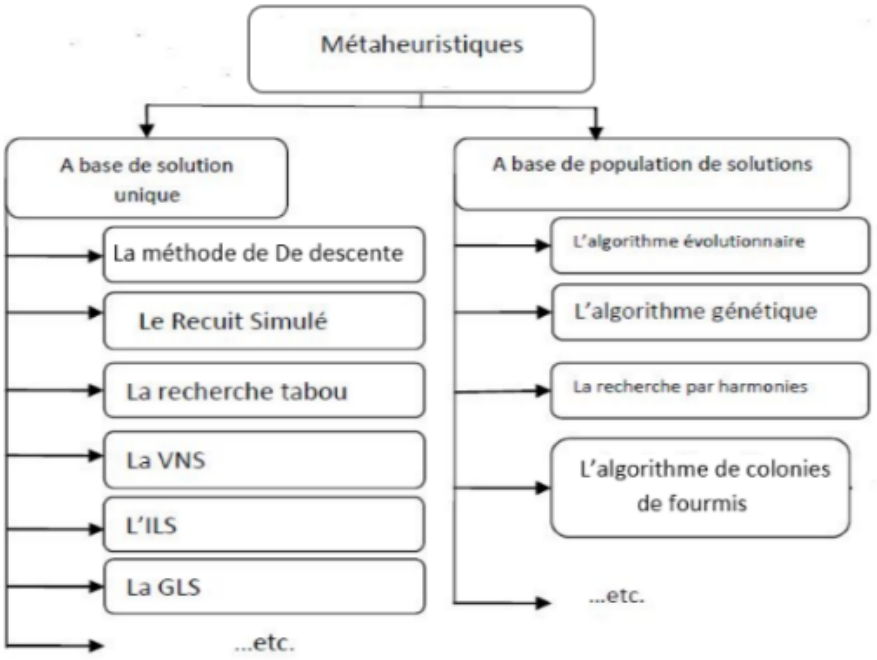


Figure 1.4 : Classification des métaheuristiques.

5.1 Les métaheuristiques à solution unique

Dans cette section, nous présentons les métaheuristiques à base de solution unique, aussi appelées méthodes de trajectoire. Contrairement aux métaheuristiques à base de population, les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, le recuit simulé, la recherche tabou,…..

5.1.1 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. [19].

5.1.2 La recherche tabou (TS : Tabu Search)

La recherche tabou est une métaheuristiques originalement développée par Glover, 1986 et indépendamment par Hansen, 1986 14 . Elle est basée sur des idées simples, mais elle est néanmoins très efficace. Cette méthode combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes permettant à celle-ci de surmonter l'obstacle des optima locaux, tout en évitant de cycler. Elle a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire : problèmes de routage de véhicule, problèmes d'ordonnancement, problèmes de coloration de graphes, etc. Dans une première phase, la méthode de recherche tabou peut être vue comme une généralisation des méthodes d'amélioration locales. En effet, en partant d'une solution quelconque x appartenant

à l'ensemble de solutions S, on se déplace vers une solution x' située dans le voisinage $N(x)$. Donc l'algorithme explore itérativement l'espace de solutions S. [7].

5.2 Les métaheuristiques à population de solutions

Contrairement aux algorithmes partant d'une solution singulière, les métaheuristiques à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions .On distingue dans cette catégorie, les algorithmes évolutionnaires, qui sont une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle, énoncée par Charles Darwin 20 et les algorithmes d'intelligence en essaim qui, de la même manière que les algorithmes évolutionnaires, proviennent d'analogies avec des phénomènes biologiques naturels. [A6].

5.2.1 Algorithmes génétiques

Les algorithmes génétiques fonctionnent sur une analogie avec la reproduction des êtres vivants. Une population d'individus (correspondants à des solutions) évolue en même temps comme dans l'évolution naturelle en biologie. Pour chacun des individus, on mesure sa faculté d'adaptation au milieu extérieur par le fitness. Les algorithmes génétiques s'appuient alors sur trois fonctionnalités :

- La sélection : cette fonction permet de favoriser les individus qui ont un meilleur fitness (pour nous le fitness sera le plus souvent la valeur de la fonction objectif de la solution associée à l'individu).
- Le croisement : le rôle de cette fonction combine deux solutions parents pour former un ou deux enfants (offspring) en essayant de conserver les "bonnes" caractéristiques des solutions parents.
- La mutation : qui permet d'ajouter de la diversité à la population en mutant certaines caractéristiques (gènes) d'une solution. [A2].

Chapitre 02 Les problèmes d'ordonnancements

1- Introduction

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation d'un en- semble de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.), et de contraintes portant sur l'utilisation et la disponibilité des ressources requises pour les tâches, et visant à minimiser (resp. maximiser) un certain critère d'optimalité .

Dans ce chapitre on va vous présenter les différents éléments d'un problème de ordonnancement, leurs objectifs, leurs classifications . Nous verrons également comment résoudre et représentation des problèmes 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 ». [16]

3- Les éléments d'un problème d'ordonnancement

Dans un problème d'ordonnancement, quatre notions fondamentales interviennent : les tâches, les ressources, les contraintes et les objectifs. Dans ce qui suit, on donnera une définition détaillée de chacun de ces notions.

Les tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début et une date de fin d'exécution et qui consomme des ressources avec des quantités déterminées. Un coût (ou poids) est attribué à une tâche pour estimer sa priorité, son degré d'urgence, ou son coût d'immobilisation dans les système.

On distingue deux types de tâches :

- les tâches morcelables (préemptives) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.
- les tâches non morcelables (indivisibles) qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

On note en général N= {J1, J2,…, Jn} l'ensemble des tâches, chaque tâche est caractérisée par :

- La durée opératoire de la tâche i sur la machine j : (pij).
- La date de disponibilité de la tâche i : (ri).
- La date de début d'exécution de la tâche i :(si).
- La date de fin d'exécution de la tâche i :(Ci).
- La date d'achèvement souhaitée de la tâche i :(di).
- Le facteur de priorité ou poids de la tâche i : (wi).
- Le retard algébrique de la tâche i : (Li = Ci– di)
- Le retard vrai de la tâche i : (Ti = max (Ci– di, 0)).
- L'indicateur de retard de la tâche i : (Ui = 1, si Ti>0, Ui = 0, sinon.

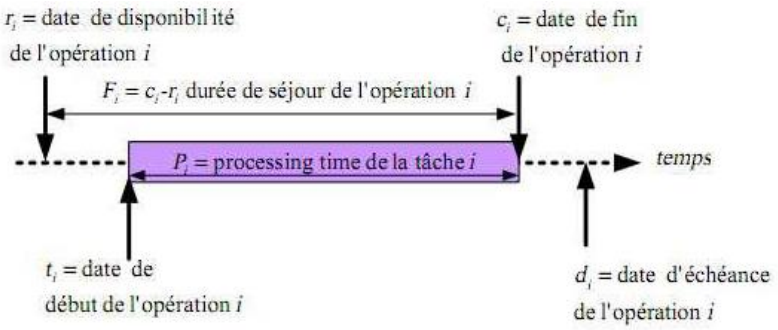


Figure 2.1 : La caractéristique d'une tache

Les ressources

Une ressource est un moyen technique ou humain utilisé pour la réalisation d'une tâche et disponible en quantité limitée. Dans un atelier, plusieurs types de ressources sont distingués.

1. Selon leurs disponibilités au cours du temps, on trouve :
   - Les ressources renouvelables, comme c'est le cas pour les machines, personnels, équipements, etc. La ressource est dite renouvelable si après avoir été utilisé par une ou plusieurs opérations, elle est à nouveau disponible en même quantité.
   - Les ressources non-renouvelables, souvent appelées ressources consommables ou bien ressources financières. On dit que la ressource est non-renouvelables si ça disponibilité décroît après avoir été allouée à une opération. C'est le cas pour la matière première, budget.
   - Les ressources doublement contraintes, ces ressources combinent les contraintes liées aux deux catégories précédentes. Leur utilisation instantanées et leur consommation globale sont toutes les deux limitées. C'est le cas des ressources d'énergie (pétrole, électricité, etc.).
2. Selon leurs capacités, on trouve :
   - Les ressources disjonctives (ou non-partageables), il s'agit des ressources qui ne peuvent exécuter qu'une seule opération à la fois c'est le cas par exemple de machine-outil ou robot manipulateur.
   - Les ressources cumulatives (ou partageables), il s'agit des ressources qui peuvent être utilisé par plusieurs opérations simultanément (équipes d'ouvriers, poste de travail, etc.).[11]

Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre conjointement les variables de décision 1 . En d'autres termes, les contraintes représentent les conditions à respecter lors de la construction de l'ordonnancement pour qu'il soit réalisable. Plus les contraintes sont nombreuses, plus le problème d'ordonnancement devient plus difficile. .[11].

Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distingués : contraintes de ressources et contraintes temporelles

- les contraintes de ressources :

plusieurs types de contraintes peuvent être induits parla nature des ressources. A titre d'exemple, la capacité limitée d'une ressource implique un certain nombre, à ne pas dépasser, de tâches à exécuter sur cette ressource.

Les contraintes relatives aux ressources peuvent être disjonctives, induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource, ou cumulatives impliquant la limitation du nombre de tâches à réaliser en parallèle.

- les contraintes temporelles :

elles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. Ces contraintes peuvent être :

- des contraintes de dates butoirs, certaines tâches doivent être achevées avant une date préalablement fixée, des contraintes de précédence, une tâche i doit précéder la tâche j
- des contraintes de dates au plus tôt, liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

4- Les objectifs de l'ordonnancement

Les objectifs dits aussi les critères d'évaluation sont les indicateurs de performance sur lesquels se base le choix d'un ordonnancement satisfaisant. En ordonnancement, les critères à optimiser consistent à minimiser ou maximiser une fonction objectif. Cette fonction objectif est générale- ment liée aux temps, aux ressources ou bien aux coûts. [11].

- 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.

5- La typologie des problèmes d'ordonnancement

Une typologie des problèmes d'ordonnancement dans un atelier peut s'opérer selon le nombre et la nature des machines ainsi que l'ordre d'enchaînement des opérations (gamme defabrication). Deux grandes familles de problèmes d'ordonnancement se présentent. La première famille regroupe les problèmes pour lesquels chaque job nécessite une seule opération. La deuxième regroupe ceux dont les jobs nécessitent plusieurs opérations.

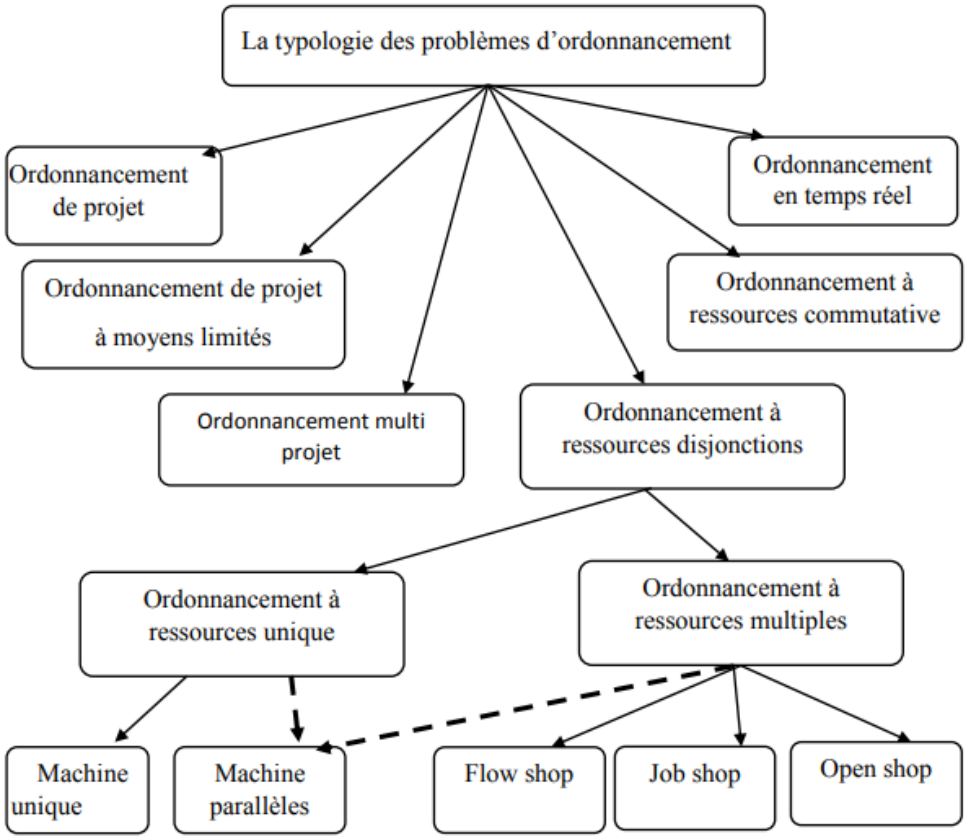


Figure 2.2 :la représentation des types d'ordonnancement.

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.

➢ 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. [19]

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.

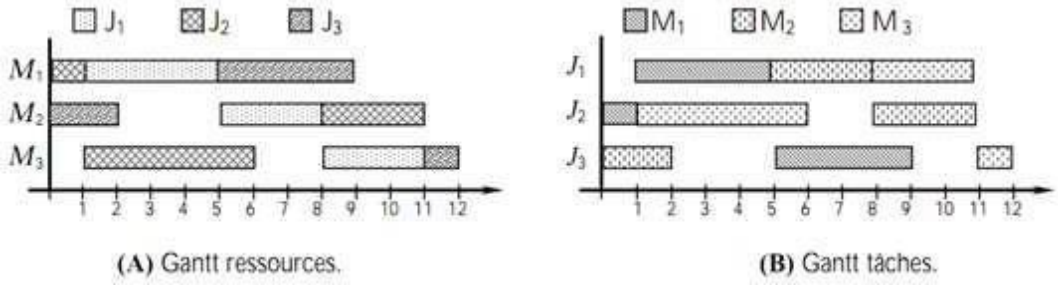


Figure 2.3 Le diagramme de Gantt

➢ 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

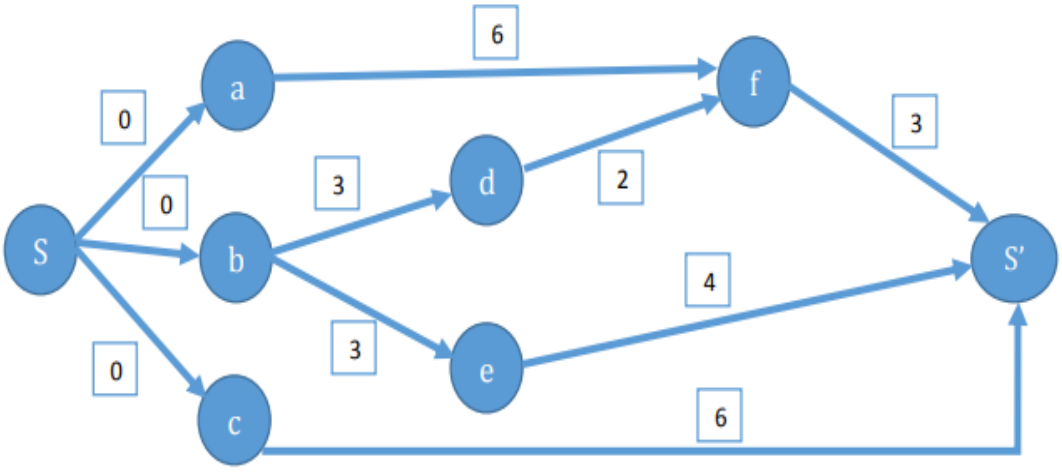


Figure 2.4 Graphe Potentiel-Tâches d'un ordonnancement

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).

➢ 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

La classification des problèmes d'ordonnancement dans un atelier peut se fait selon le nombre de machines et leur ordre d'utilisation pour fabriquer un produit, qui dépend de la nature de l'atelier considéré. Un atelier est caractérisé par le nombre de machines qu'il contient et par son type.

Dans ce chapitre, nous présenterons la définition du problème de l'atelier, quelques concepts sur les types d'ateliers, et présenterons également les relations entre ces types et à la fin nous pouvons voir la complexité et les types de problèmes.

.

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).[6]

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

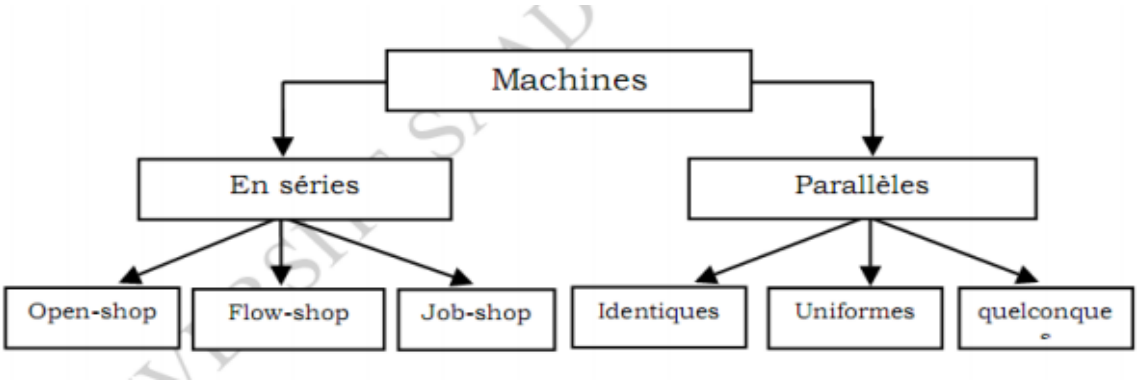


Figure 3.1 : Les types des machines

3. Les types des Problèmes

Les problèmes d'ordonnancement sont généralement classés en deux principaux modèles dépendamment du nombre d'opérations que requièrent les jobs: des modèles à une opération (machine unique et machines parallèles) et des modèles à plusieurs opérations (flow-shop, open shop et job shop).

3.1 Modèles à une opération

3.1.1 Modèle à machine unique

Dans un modèle à machine unique, l'ensemble des tâches à réaliser est exécuté par une seule 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. .[7].

Ce modèle est illustré dans la Figure 3.2.

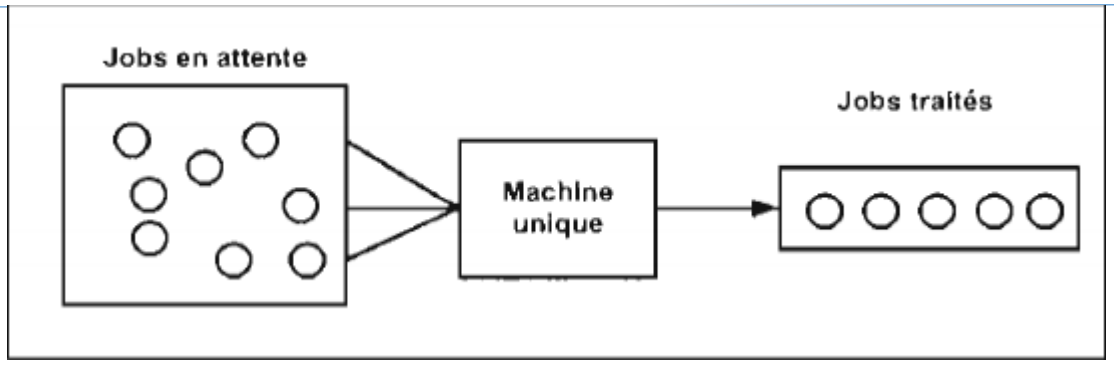


Figure 3.2 : Modèle à machine unique.

3.1.2 Modèle à machines parallèle

Le deuxième modèle est le modèle à machine parallèle. Ce modèle est utilisé surtout dans les secteurs industriels tels que: l'industrie alimentaire, les industries plastiques, les fonderies et en particulier l'industrie textile. Le processus de déroulement de ce système de production est le suivant: à chaque fois qu'une machine i se libère, ont lui affecte un job j comme illustré à la Figure 3.3.

Dans le cas d'un processus d'assemblage industriel, par exemple, si l'une des étapes d'assemblage nécessite beaucoup de temps, il serait très intéressant alors d'avoir plusieurs machines parallèles qui effectuent la même tâche.

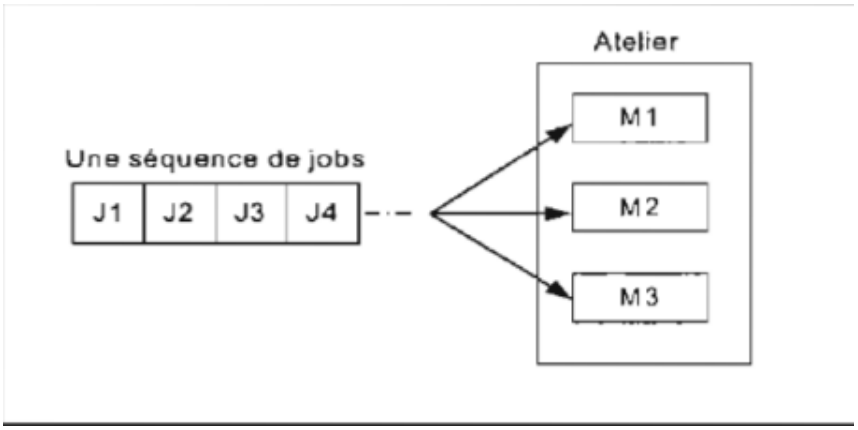


Figure 3.3 : Modèle à machines parallèle.

Ce type d'atelier peut être divisé en trois sous-catégories selon la vitesse d'exécution des machines :

- les ateliers à machines identiques : toute tâche peut s'exécuter sur n'importe quelle machine avec une même durée opératoire.

- les ateliers à machines uniformes : chaque machine possède sa propre vitesse, indépendamment de la durée de la tâche à exécuter.

- les ateliers à machines indépendantes : la vitesse de chaque machine dépend de l'opération à effectuer.[1].

3.2 Modèles à plusieurs opérations

Le modèle à plusieurs opérations est constitué des cas où un job, pour se réaliser, doit passer par plusieurs machines, chacune de ces machines ayant ses spécificités. On distingue trois modèles selon l'ordre de passage des jobs sur les machines, à savoir les modèles de flow-shop, job-shop et open-shop.

3.2.1 Modèle flow-shop

Dans le modèle de flow-shop, les ordres de fabrication visitent les machines dans le même ordre, avec des durées opératoires pouvant être différentes. Chaque job va être s'exécuter sur les M machines en série et tous les jobs vont suivre le même ordre de passage sur ces machines. Ce type de modèle est aussi appelé modèle linéaire.

La figure 3.4 illustre le cas d'un flow-shop avec quatre machines et quatre jobs. Les quatre jobs suivent le même ordre de traitement sur les quatre machines.[7].

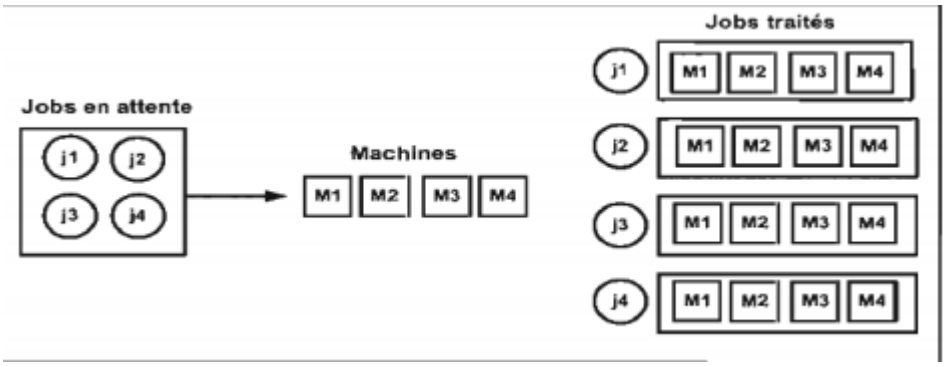


Figure 3.4 : Modèle flow-shop.

3.2.2 Modèle job-shop

Concernant le modèle de job-shop, chaque job à un ordre à suivre et chacun d'entre eux peut s'exécuter plusieurs fois sur la même machine; ce qui n'est pas le cas du flow-shop. Il s'agit dans ce cas de déterminer les dates de passage sur différentes ressources d'ordres de fabrication ayant des trajets différents dans l'atelier.

Ces ordres de fabrication partageant des ressources communes, des conflits sont susceptibles de survenir, résultant des croisements de flux illustrés à la Figure 3.5.

Dans cette figure, l'ordre de passage de chaque job est indiqué par un chemin de même couleur que celui du job. Par exemple, le job J1 passe par toutes les machines, alors que le job J3 ne passe que par la deuxième et la quatrième machine. Dans son expression la plus simple, le problème consiste à gérer ces conflits tout en respectant les contraintes données, et en optimisant les objectifs poursuivis.

Les types de ressources et de contraintes prises en compte peuvent toutefois considérablement compliquer le problème. Plus on intégrera de contraintes, plus on se rapprochera d'un cas réel, mais moins on disposera de méthodes de résolution satisfaisantes.[7].

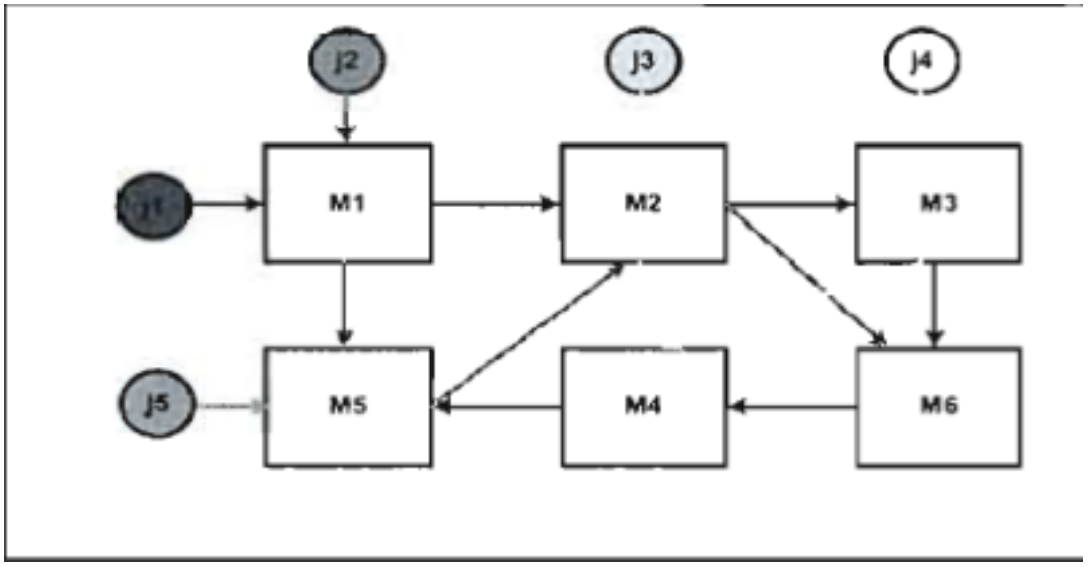


Figure 3.5 : Modèle job-shop.

3.2.3 Modèle open-shop

Dans le modèle d'open shop, l'ordre de passage des n jobs sur les m machines n'est pas connu à l'avance. Cet ordre est déterminé lors de la construction de la solution. Chaque job j peut avoir son propre ordre de passage sur toutes les machines.

Le fait qu'il n'y ait pas d'ordre prédéterminé rend la résolution du problème d'ordonnancement de ce type plus complexe, mais offre cependant des degrés de liberté intéressants.

À la Figure 3.6, nous avons un ensemble de quatre jobs et un ensemble de quatre machines. À droite de la figure nous pouvons remarquer que chaque job a suivi un ordre de passage différent sur les quatre machines.

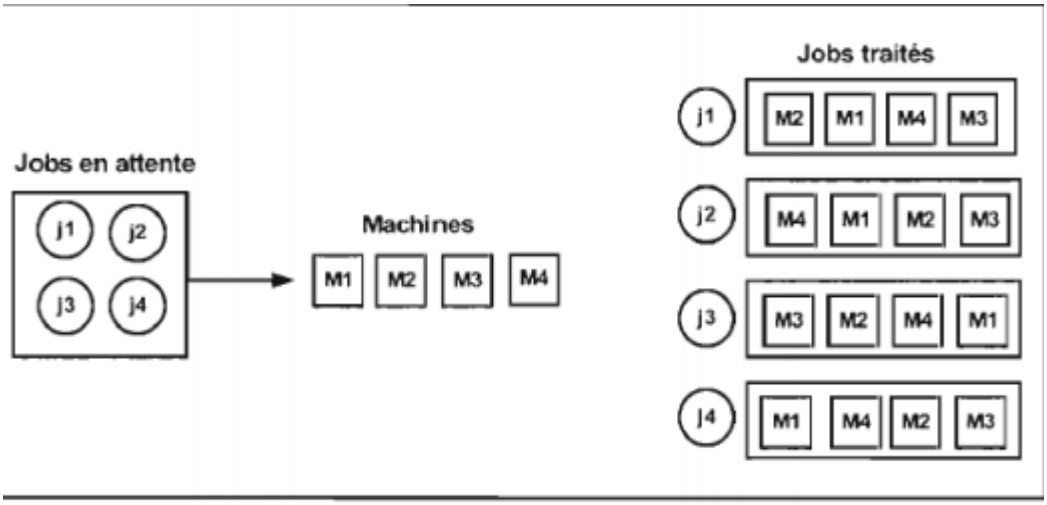


Figure 3.6 : Modèle open-shop.

3.2.4 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).[6].

– ***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 [16] (voir la figure 3.7).

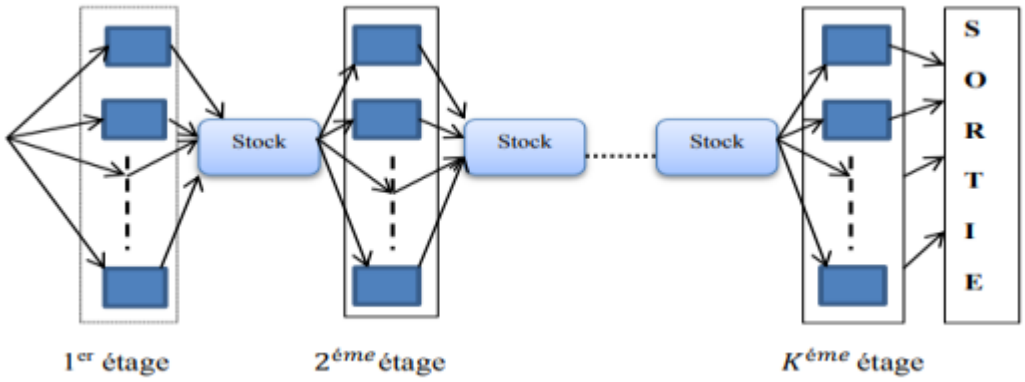


Figure 3.7 : La représentation d'un flow show hybride à « k » étages.

– ***Job Shop hybride***

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 un sous ensemble d'opérations. Plus précisément, une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération.

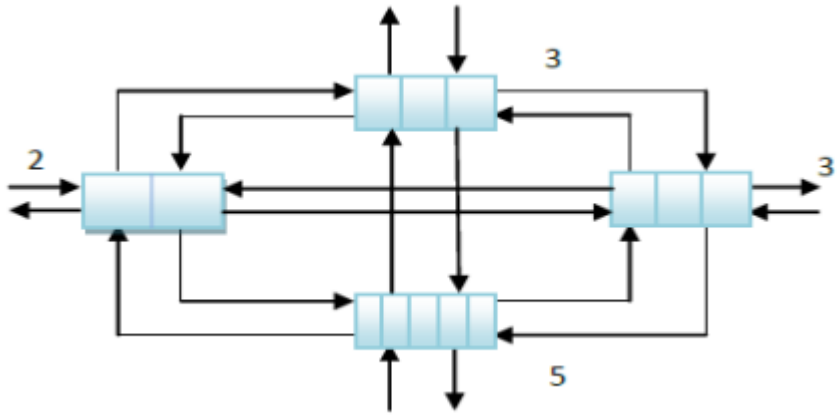


Figure 3.8 : La représentation d'un job shop hybride.

4. Relations entre les types

Les organisations présentées précédemment ne sont pas indépendantes. En effet, des relations de ressemblance, de réduction et de différentiation existent entre elles.

L' étude des relations qui existent entre les différents problèmes d' ordonnancement revêt un grand intérêt, dans la mesure où cela permet d' appliquer des algorithmes de résolution connus pour certaines classes de problèmes à d' autres classes qui leurs sont réductibles.

La figure 3.9 ci-dessous illustre ces différentes relations : chaque arc de *A* à *B*s' interprète que *B* est un cas particulier de *A* en considérant la condition associée. Par exemple, le Flow Shop simple est un cas particulier du Flow Shop hybride, où toutes les machines existent en un seul exemplaire. Il est également, un cas particulier du Job Shop simple avec gammes identiques pour toutes les tâches.

Dans certains cas, ce schéma peut être lu en terme de domaine de solutions réalisables. Par exemple, l' ensemble des solutions réalisables du problème de "machine unique" appartient également à l' ensemble des solutions réalisables du problème de Flow Shop simple.[13].

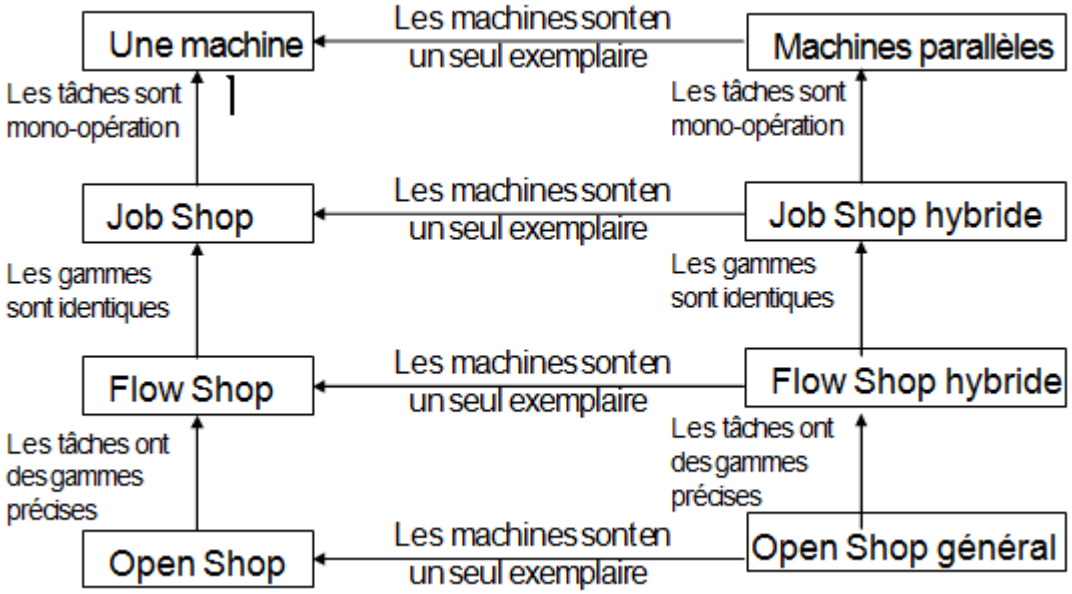


Figure 3.9: Relations entre les différentes organisations

5. 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 tâches indépendantes dans les problèmes d'ordonnancement à machines parallèles

1- Introduction

Le but de ce chapitre, introduire quelques notion sur les problèmes d'ordonnancement à machines parallèles, Le problème de Machine identique en parallèle, la représentation de cet problème.

Nous terminons ce chapitre par deux méthodes de résolution de ces types de problème : Les algorithmes génétiques et L'algorithme glouton.

2 – Définition (les problèmes d'ordonnancement à machines parallèles)

Ces problèmes sont une généralisation de ceux à une seule machine, plusieurs machines sont disponibles pour l'exécution d'une tâche.

L'utilisation du parallélisme pour la réalisation de projets de grande taille et pour le traitement d'applications réclamant une puissance de calcul de plus en plus importante est aujourd'hui une réalité. Il est aussi intéressant de considérer le parallélisme pour des raisons de fiabilité. D'autres part, le parallélisme pose des problèmes non rencontrés dans l'exécution séquentielle, comme l'extraction du parallélisme d'un projet et sa décomposition en tâches, la détection des contraintes chronologiques (délais de communication et les dépendances entre les tâches (contraintes de précédence) ou entre les tâches et les machines (contraintes de placement) . Le processus d'ordonnancement consiste en l'affectation des tâches aux machines et leur séquencement sur chacune d'elles.[4].

3- Tâches indépendantes (Machines identiques)

Considérons n tâches Ji avec des temps d'exécution pi (i = 1, . . . , n) à exécuter sur m ma- chines parallèles identiques M1, . . . , Mm.

P|pmtn|Cmax

– Une borne inférieure pour P|pmtn|Cmax est donnée par

$$LB := \max\left\{\max_i p_i, \left(\sum_{i=1}^{n} p_i\right)/m\right\}$$

– Un ordonnancement qui atteint cette borne peut être obtenu en un temps O(n) .

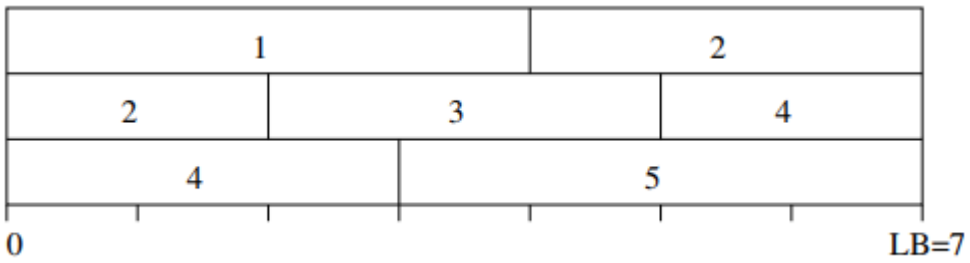
– Remplir les machines successivement avec les tâches dans un ordre arbitraire.

– Quand la borne est atteinte, diviser la tâche en deux parties et ordonnancer la deuxième partie de la tâche préemptée au temps 0 sur la machine suivante.

– Vu que pi ≤ LB pour tout i, les deux parties d'une tâche préemptée ne se chevauchent pas.

Exemple : m = 3

i	1	2	3	4	5
p_i	4	5	3	5	4



P|pmtn; ri |Cmax et P|pmtn|Lmax

– Cas particuliers de Q|pmtn; ri |Lmax.

– O(n log n + mn) (voir plus loin).

P|pmtn; ri |Lmax

Considérons tout d'abord la version "décision" du problème : Etant donné une constante L, existe-t-il un ordonnancement tel que

$$\max_{i=1}^{n} L_i = \max_{i=1}^{n}\{C_i - d_i\} \le L ?$$

De plus, nous voulons trouver un tel ordonnancement s'il existe.

– La condition est satisfaite ssi :

$$C_i \leq d^L := L + d_i \text{ pour tout } i = 1, \ldots, n.$$

– Chaque tâche Ji doit être exécutée dans l'intervalle $[r_i, d^L]$ (fenêtres de temps, time windows).

– Nous allons montrer que le problème d'ordonnancer avec préemption n tâches sur m ma- chines identiques de sorte que les tâches sont exécutées pendant leur fenêtre de temps

$[r_i, d_i]$ se ramène à un problème de flot maximum.

– Soient t1 < t2 < · · · < t_r la séquence ordonnée des différentes valeurs de r_i et d_i.

– Considérons les intervalles

$$I_K := [t_K, t_{K+1}] \text{ de longueur } t_{K+1} - t_K \text{ pour } K = 1, \ldots, r-1.$$

– Construction du graphe :

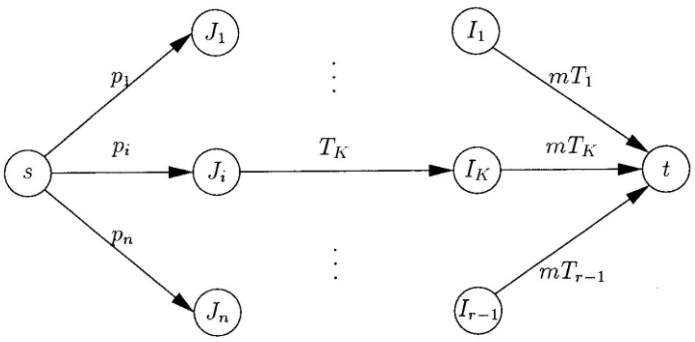
Nœuds : Un nœuds par tâche J_i et par intervalle I_K, plus deux nœuds artificiels *s* et *t*.

Arcs :

– (s, J_i) de capacité p_i pour tout $i = 1, \ldots, n$.

– (I_K, t) de capacité mT_K pour tout $K = 1, \ldots, r-1$.

– (J_i, I_K) si J_i peut être exécuté pendant I_K ($r_i \leq t_K$ et $t_{K+1} \leq d_i$), de capacité T_K.



– Il existe un ordonnancement satisfaisant toutes les fenêtres de temps ssi le flot maximum a une valeur $\sum p_i$.

– Complexité : $O(n^3)$.

– Pour résoudre $P|pmtn; r_i|L_{max}$, recherche par bissection sur la valeur de L.

Supposons que $d_i \leq r_i + n \max p_j$. ce qui implique

$$-n \max_{j=1}^{n} p_j \leq L_{max} \leq n \max_{j=1}^{n} p_j.$$

– Algorithme en

$$O(n^3(\log n + \log \frac{1}{s} + \log(\max_{j=1}^{n} p_j))$$

– pour résoudre le problème avec une erreur absolue d'au plus s.

P p_i = 1; **r_i L_{max}**

– On suppose que les tâches sont ordonnées de sorte que $r_1 \leq r_2 \leq \cdots \leq r_n$.

– r_i entiers : ordonnancer dans l'ordre des dates limites d_i croissantes.

– Plus précisément : si au temps courant toutes les machines ne sont pas occupées et qu'il y a une tâche J_i avec r_i t_i, on ordonnance une telle tâche avec la date limite la plus petite.

– Complexité $O(n \log n)$.

Cette approche ne fonctionne pas si les r_i ne sont pas entiers. Considérons l'exemple suivant avec $m = 2$.

i	1	2	3	4
r_i	0	0.2	0.5	0.5
d_i	2	3	1.9	3

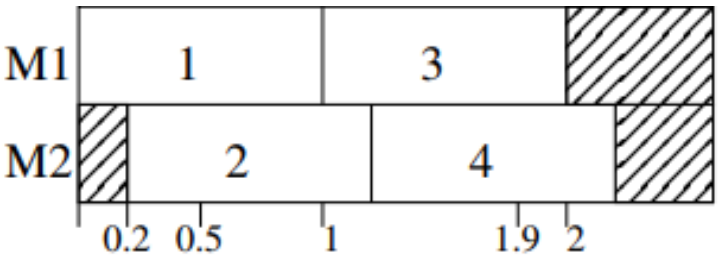


Figure 4.1 : – Earliest Due Date — $L_{max} = 0.1$.

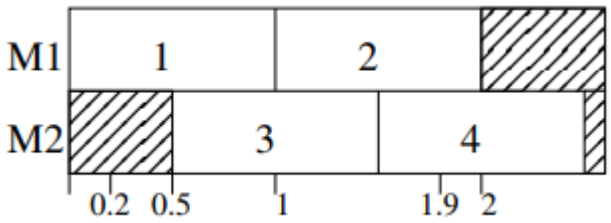


Figure 4.2 : – Un meilleur ordonnancement — $L_{max} = -0.4$.

8- Le problème de 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,…..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 :

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

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

$P_{[kr]} = \sum_{j=1}^{n} Xjkr\ Pjk\ r_{ij}$ k=1,2,….,m r=1.2,….. nk (4)

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

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

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

$C[kr] = \max(C[k,r\text{-}1] + S[kr], r[kr]) + P_{[kr]}$ k=1,2,.,m r = 1,2... nk (8)

$T_{[kr]} = \max(C[kr] - d[kr], 0)$ k=1,2.. , m r = 1,2,., nk (9)

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

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

Xjkr =0 or 1 k = 1,2,….. m r = 1,2, … n_k j=1,2,…..n (12)

Yij =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 [6].

9- 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.

10- Difficulté du problème P|prec|C_{max}

Les problèmes d'ordonnancement intéressants se montrent difficiles. P| |C_{max}, le problème de tâches indépendantes et de durées d'exécution arbitraires, est montré NP-difficile même pour le cas simple à deux machines (P_2| |C_{max}).

Théorème 1.1 : Le problème d'ordonnancement $P_2| |C_{max}$ est NP-difficil

Preuve : Un POC est NP-difficile si le problème décisionnel lui correspondant est NP- complet. Le problème $P_2| |C_{max}$. « Existe-t-il une affectation des tâches, tel que la durée totale des tâches affectées à chaque machine, est au plus k ? » est la version décision du problème $P_2| |C_{max}$, où k est un entier.

Considérons le problème de partition connu comme étant NP-complet.

Le problème Partition : Soit un ensemble fini E d'entiers positifs a_j. Existe t-il un sous ensemble E' de E tel que :

$$\sum_{a_j \in E} a_j = \sum_{a_j \in E-E'} a_j .$$

Etant donnée une instance quelconque du problème de partition, on peut créer une instance de la version décision de $P_2| |C_{max}$, de la manière suivante :

i. n = |E|, où n est le nombre de tâches

ii. $p_j = a_j$, j = 1 .. n

iii. $K= \frac{1}{2}\sum_{j=1}^{n} pj$.

Il existe un sous-ensemble E' pour l'instance de partition si et seulement si il existe un ordonnancement de longueur inférieur ou égale à $K= \frac{1}{2}\sum_{j=1}^{n} pj$

pour $P_2| |C_{max}$. [4].

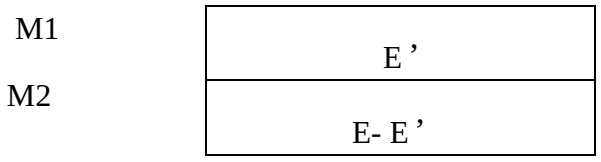


Figure 4.3 : Illustration de la réduction Partition α $P_2| |C_{max}$.

11- Représentation d un problème d'ordonnancement à machines parallèles

Pour représenter un ordonnancement, on utilise couramment une représentation dite *diagramme de Gantt*. Celle-ci utilise un repère orthogonal dans un plan. L'axe des abscisses représente le temps et l'axe des ordonnées représente l'ensemble des machines. Pour chaque machine, on représente la séquence de tâches effectuées dans le temps. Les parties en gris représentent les périodes de oisiveté d'une machine, dues aux éventuelles indisponibilités des tâches.

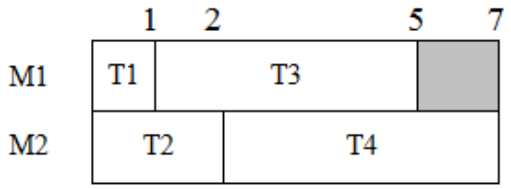


Figure 4.4 : Diagramme de Gantt.

12- L'algorithme glouton

En informatique, un algorithme glouton (greedy algorithm en anglais, parfois appelé aussi algorithme gourmand) est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local. Dans certains cas cette approche permet d'arriver à un optimum global, mais dans le cas général c'est une heuristique. L'illustration ci-contre montre un cas où ce principe est mis en échec.

Principe : En algorithmique, et plus précisément en optimisation combinatoire, de nombreux algorithmes fonctionnent en faisant une série d'étapes, avec des choix. Une méthode classique est la programmation dynamique qui permet de tester tous les choix. Cependant les complexités des algorithmes de ce type sont parfois trop grandes et l'on choisit d'autres méthodes. L'une de ces méthodes est de choisir localement la meilleure solution : ce sont les algorithmes gloutons.

Exemples de problèmes :

- 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.

- Arbre couvrant de poids minimal

Le problème consiste, dans un graphe non orienté connexe et value, à trouver un sous-ensemble d'arêtes, formant un arbre, incluant tous les sommets (c'est-à-dire un arbre couvrant), tel que la somme des poids de chaque arête soit minimale (d'où le terme arbre couvrant de poids minimal). L'algorithme de Prim et l'algorithme de Kruskal sont deux des algorithmes gloutons pour le problème

Dans l'algorithme de Prim, on choisit un sommet et l'on fait grandir l'arbre depuis ce sommet de façon gloutonne. Plus précisément, on choisit un sommet arbitraire puis l'on sélectionne l'arête incidente à ce sommet de poids minimal et on l'ajoute à l'arbre et ainsi de suite. Autrement dit on construit l'arbre de manière itérative, en sélectionnant à chaque pas l'arête de poids minimum qui relie un sommet de l'arbre avec un sommet en dehors de l'arbre.

L'algorithme de Kruskal consiste à faire croître une forêt jusqu'à couverture complète. Chaque augmentation se fait de la manière la plus économique possible.

- Coloration de graphe

le problème de coloration consiste à donner des couleurs aux sommets tel que pour chaque arête, les deux sommets reliés soient de couleurs différentes. L'ensemble de couleur est limité. Le problème est NP-complet. L'algorithme glouton suivant, parfois appelé first-fit algorithm est une heuristique pour ce problème.

Entrée :

liste ordonnée V des n sommets d'un graphe G

liste ordonnée C de couleurs

Pour i variant de 1 à n

v = V[i]

couleur = la première couleur de C non utilisée par les voisins de v

colorier (v, couleur)

Fin pour

Pour un graphe il existe toujours un ordre des sommets qui mène à une bonne solution en utilisant cet algorithme, cependant pour d'autres ordres l'algorithme sera bloqué car toutes les couleurs disponibles auront déjà été utilisées, et l'étape la première couleur de C non utilisée par les voisins de v ne pourra pas être effectuée.[w].

9- Les algorithmes génétiques

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

Le principe d'un algorithme génétique est simple. On part d'une population de solutions du problème représentées par des individus. Cette population choisie aléatoirement constitue la population initiale. Chaque individu x de la population qui est codé sous forme d'une représentation génétique du problème, possède une fonction coût f(x) utilisée pour exprimer son degré d'adaptation à l'environnement ou à l'objectif visé, on dit qu'un individu est d'autant mieux adapté à son environnement ou plus performant que le coût de la solution qu'il représente est plus faible.[17]

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

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

- Le codage binaire

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

!Le codage par valeurs entières

Chaque gène est codé par une valeur entière. Donc, un chromosome est représenté sous forme d'une chaine d'entiers. Ce codage est bien adapté à l'optimisation des problèmes industriels réels.

- Le codage par valeur

  Dans ce type de codage, chaque gène est codé par une valeur qui appartient à un ensemble fini ou infini. Ces valeurs sont liées au problème à résoude.[15]

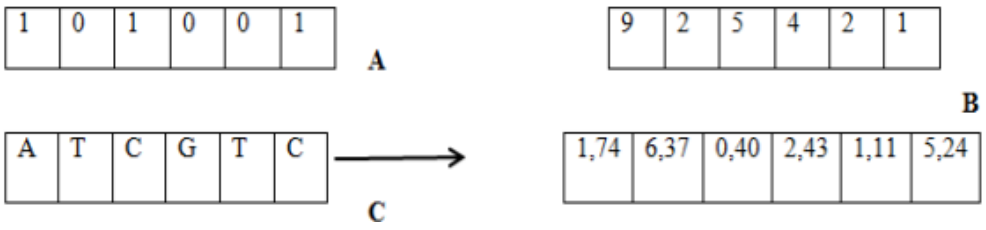


Figure 4.5 : Exemples de codage par valeurs.

(A) : codage binaire.

(B) : codage par permutation de valeurs entières.

(C) : codage par valeur.

Les opérateurs jouent un rôle prépondérant dans la possible réussite d'un AG. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement et de mutation. Le schéma d'un algorithme génétique est présenté dans la figure 4.6.

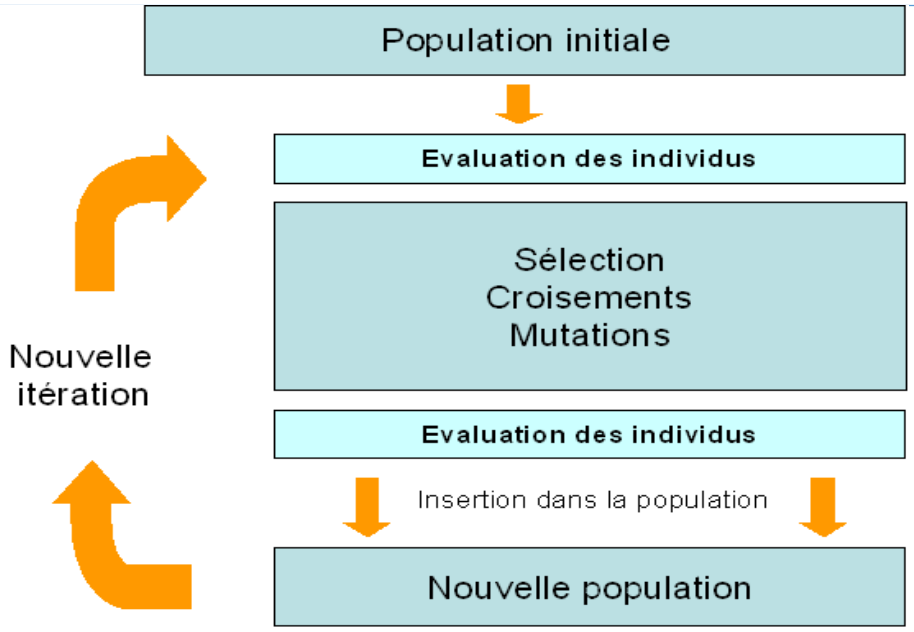


Figure 4.6 : Schéma d'un algorithme génétique .

- **Opérateur de Sélection**

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

- ***Sélection par rang :***

Cette technique de sélection sélectionne toujours les individus ayant le meilleur degré d'adaptation.

- ***Probabilité de sélection proportionnelle à l'adaptation :***

La technique de la roulette ou la roue de la chance pour chaque individu, la probabilité de la choisir est proportionnelle à son adaptation au problème. Afin de sélectionner un individu, on utilise le principe de la roue de la fortune biaisée. Cette roue est une roue de la fortune classique sur laquelle chaque individu est représenté par une portion proportionnelle à son adaptation. On effectue ensuite un tirage au sort homogène sur cette roue.

- ***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.

- **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 4.7).

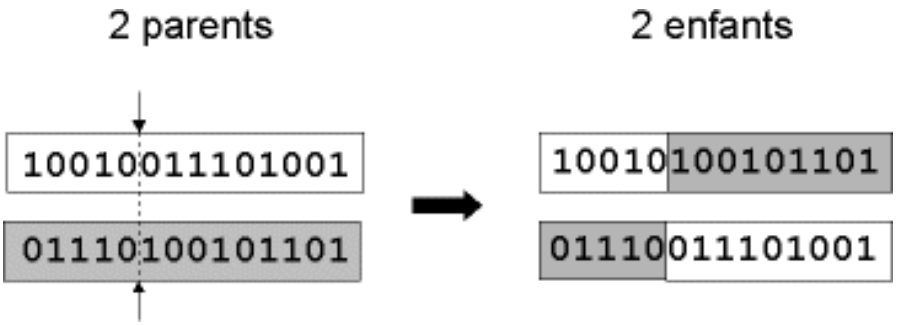


Figure 4.7 : La représentation de croissement a un point.

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 4.8)

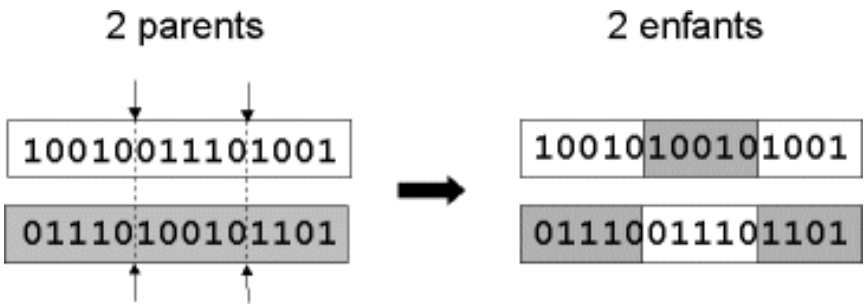


Figure 4.8 : La représentation de croissement a un multipoints

- ***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 4.9).

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

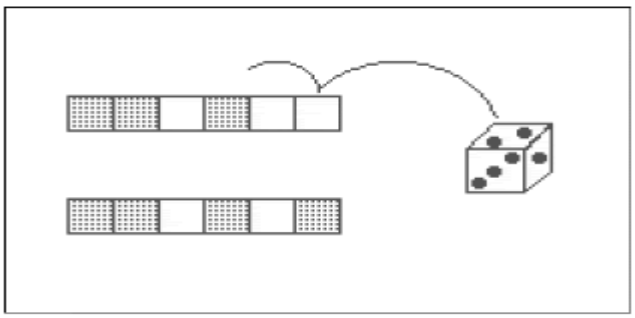
Figure 4.9 : La représentation de croissement uniforme.

- **Opérateur de Mutation**

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

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

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



.

Figure 4.10 : La mutation en codage binaire

- ***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.

Chapitre 5 IMPLEMENTATION DE PROBLEME

1- Introduction

Dans ce chapitre, nous allons passer à la conception et l'implémentation d'un problème machine en parallèle par l'algorithme génétique et l'algorithme glouton .

D'abord, nous présentons langage et les méthodes de résolution de notre problème que nous avons utilisé. Ensuite, on a présenté l'application. Après Nous avons procédé quelque exemples pour tester.

2- Langage et environnement de développement

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

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

JavaFX est un Framework et une bibliothèque d'interface utilisateur issue du projet OpenJFX, qui permet aux développeurs Java de créer une interface graphique pour des applications de bureau, des applications internet riches et des applications Smartphones et tablettes tactiles.

Créé à l'origine par Sun Microsystems, puis développé par Oracle après son rachat et ce, jusqu'à la version 11 du JDK, c'est depuis lors à la communauté OpenJFX que revient la poursuite de son développement.

Cette bibliothèque a été conçue pour remplacer Swing et AWT, qui ont été développés à partir de la fin des années 90, pour pallier les défauts de ces derniers et fournir de nouvelles fonctionnalités (dont le support des écrans tactiles).

Le cycle de sortie d'une nouvelle version de JavaFX correspond à celui de Java, soit tous les 6 mois.

L'environnement NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java. NetBeans permet la prise en charge native de divers langages tels le C, le C, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur avec coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). NetBeans est disponible sous Windows, Linux, Solaris, Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)).

3- Le problème d'une machine parallèle et les méthodes de résolution de notre problème

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.

J'ai utilisé deux algorithmes pour résoudre ce problème, l'algorithme génétique(AG) et l'algorithme glouton.

4- La conception

les procédures que je créé pour réaliser mon problème sont : Tache ,Machine, System, Position ,Génération, Itération, ArrivTache ,Main.

- « 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.

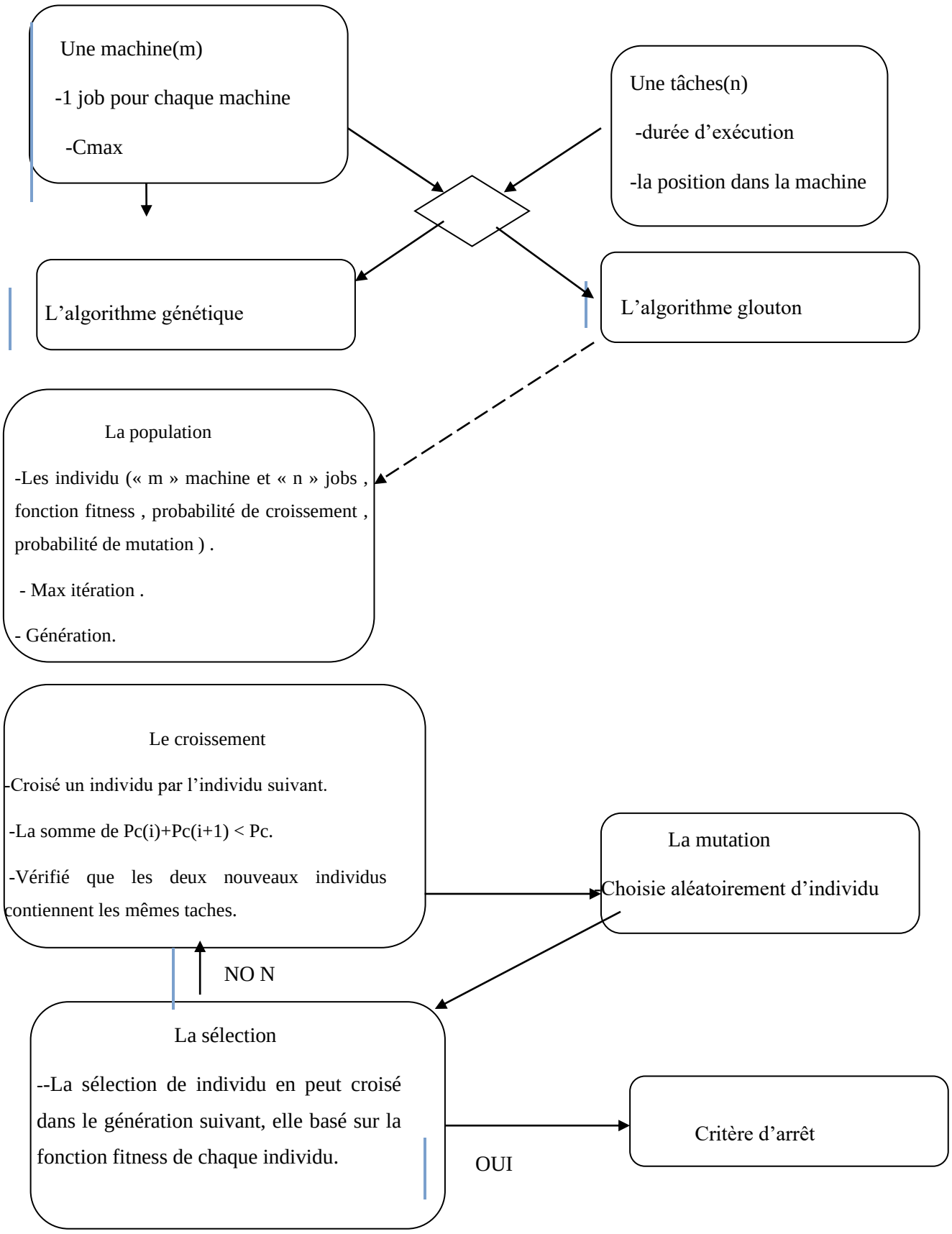


Figure 5.1 Le schéma de réalisation de

5- La réalisation

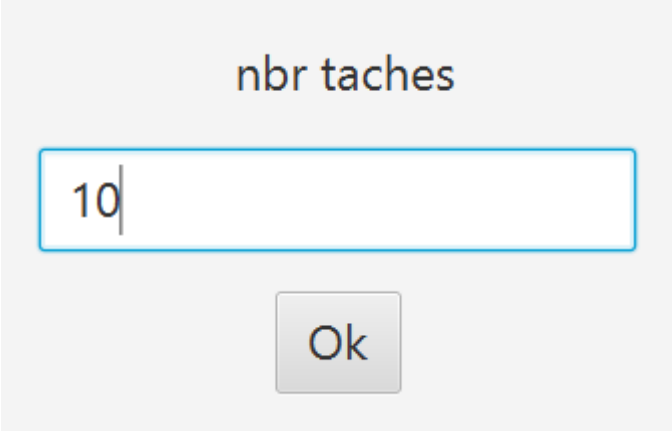


Figure 5.2 :Fenêtre de départ.

La figure 5.2 c'est la première étape d'utilisation est entrez nombre des taches.

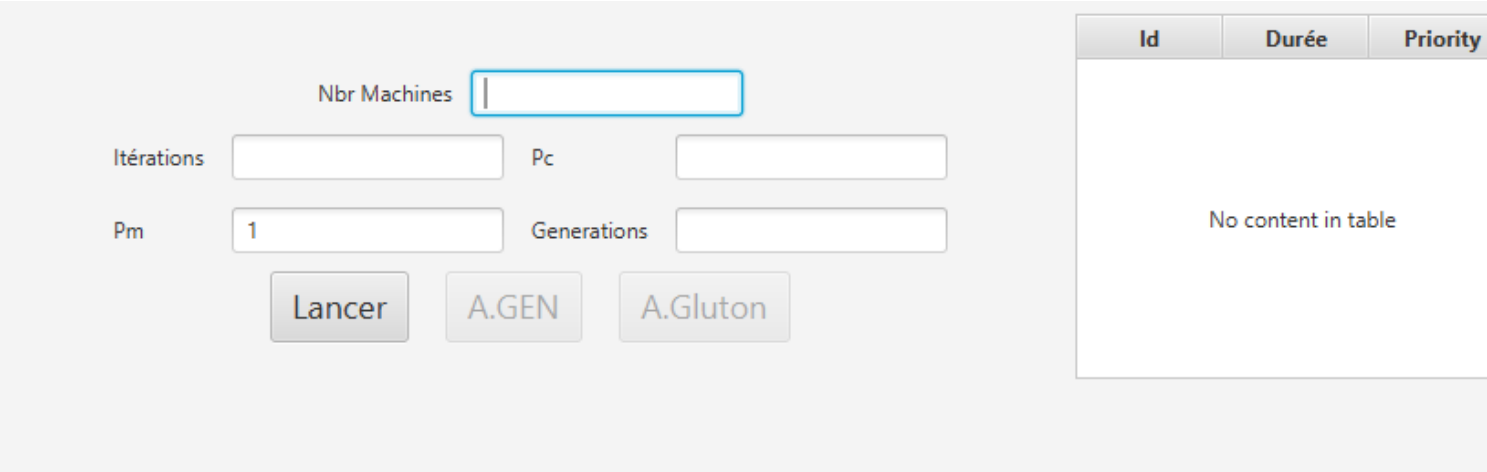


Figure 5.3 : L'interface graphique de l'application.

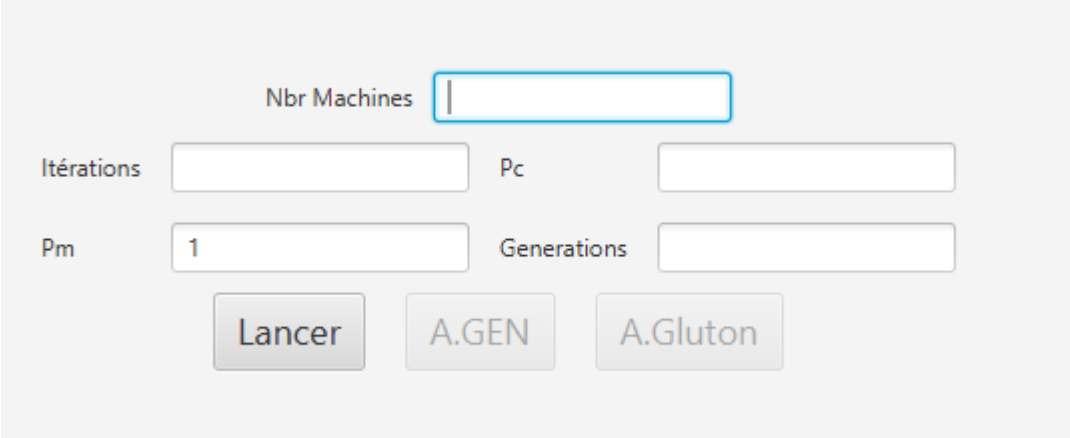


Figure 5.4: les données de l'algorithme.

La figure 5.4 c'est la partie que je entre les donnée de l'algorithme que je utilise pour résoudre mon problème.

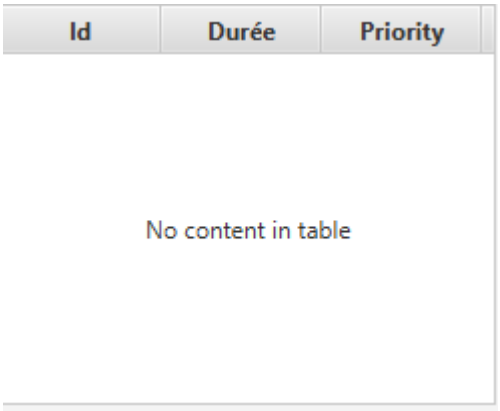


Figure 5.5 : tableau des taches.

La figure 5.5 pour représente le tableau des taches et sont durée des exécutions

6- Conception et résultats de l'AG

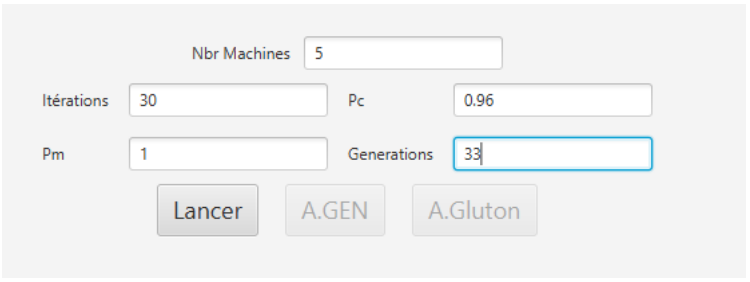
Afin d'étendre le champ d'application des AGs à un problème particulier comme celui de P□prec□C_{max}, il faut prendre en compte les spécificités du problème, en particulier les contraintes de précédence entre les tâches.

1-Codage

On a choisit de coder indirectement une solution (ordonnancement) par la liste de priorité qui donne l'ordonnancement. La permutation des tâches formant la liste est le chromosome représentant l'individu ou la solution. Une population est formée d'un ensemble de permutations de n tâches.

2- initialisation de population

Après la première étape ou entrez nombre des taches, on peut sélectionner les nombres des machines, dans notre problème on peut utiliser plus 5 machines , 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 .



Id	Durée
0	54
1	57
2	15
3	51
4	46
5	34
6	27

Figure 5.6 : L'insertion des données de l'algorithme et le tableau des tâches

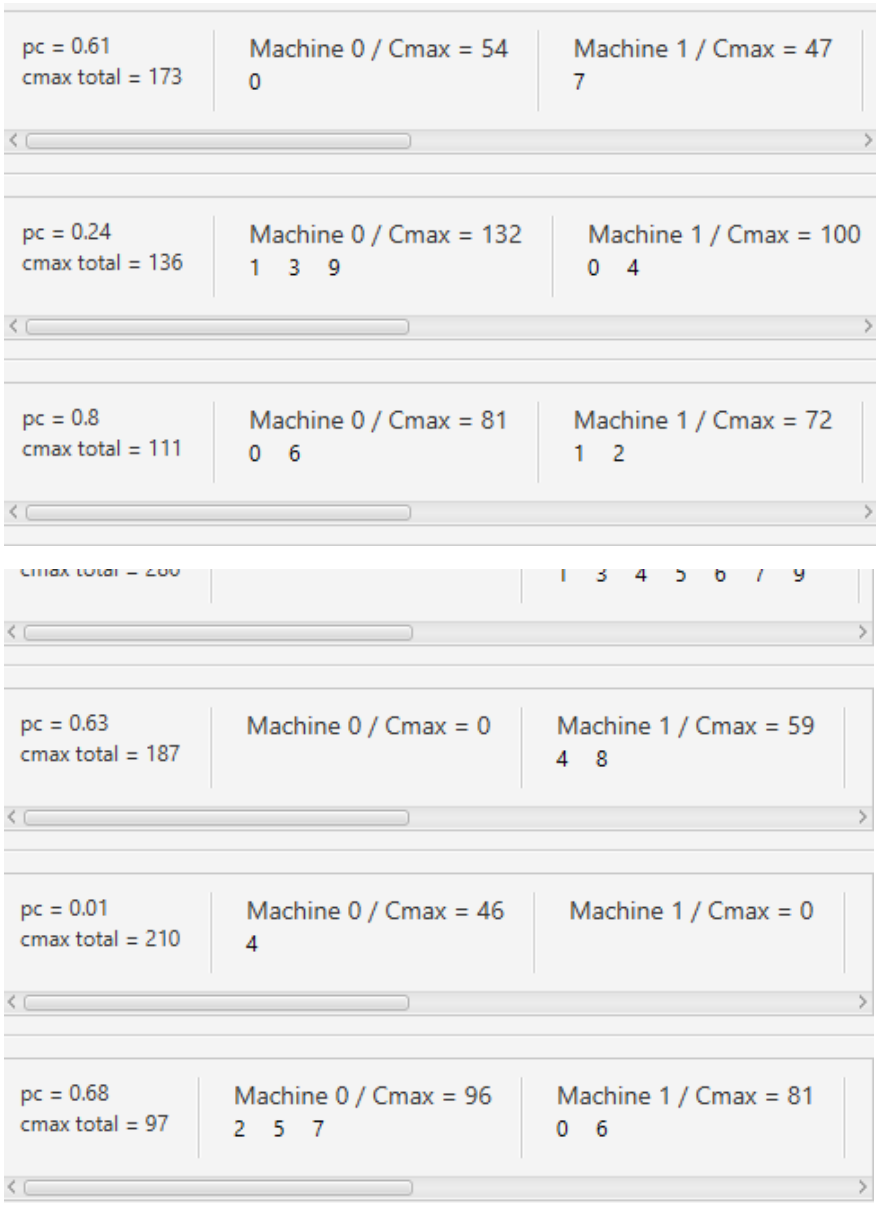


Figure 5.7 : La population.

3- L'évaluation et la sélection

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).

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.

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.

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.

7-Critère d'arrêt

On arrête le déroulement de l'algorithme si on atteint un certain nombre de générations qui ne doit pas être trop grand, sinon l'AG génère une grande partie de l'espace des solutions, ce qui diminue son efficacité. Aussi, ce nombre ne doit pas être trop petit, sinon on ne donne

pas suffisamment de temps à l'algorithme pour améliorer la population. De notre exemple on a sélectionné 33 génération.

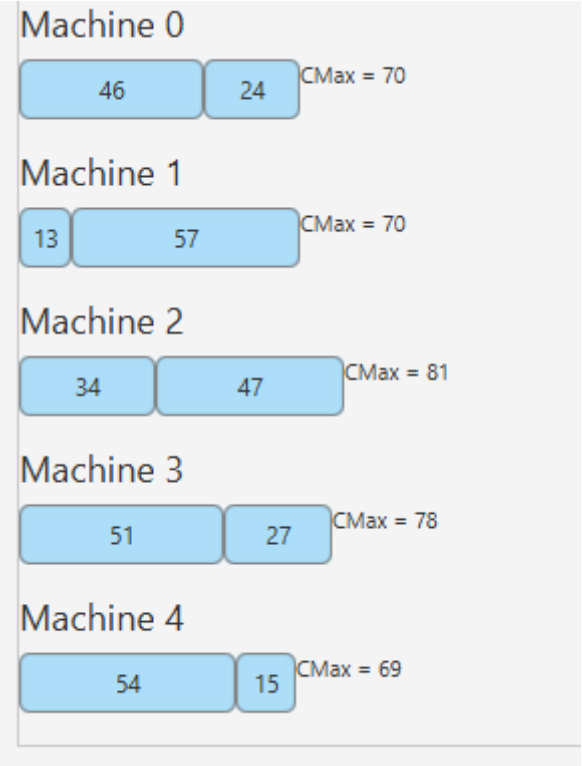


Figure 5.8 : Le diagramme de Gantt de l'AG.

L'implémentation de l'algorithme glouton

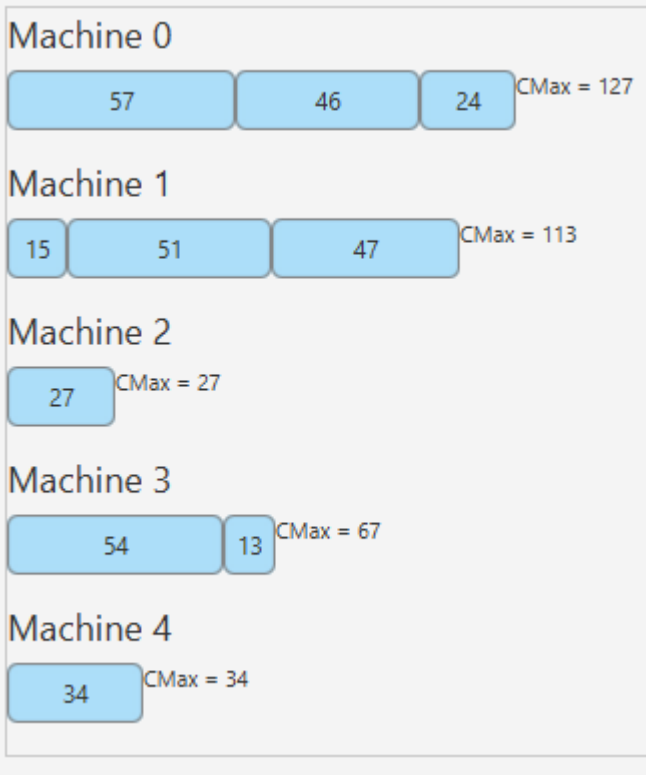


Figure 5.9 : Le diagramme de Gantt de l'AG.

7- Comparaison

Considérons le cas du problème avec deux algorithmes, l'algorithme local qui donne une solution local est l'algorithme glouton (voir la figure 5.2), est l'algorithme global qui donne une solution globale est l'algorithme génétique. 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.

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 Machines identiques en parallèle 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, Branch & Bound, programmation dynamique) 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 : là où ce sont des tâches indépendantes.

La comparaison des deux méthodes a montré la supériorité de la solution globale sur la solution locale, en d'autres termes 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

[1] A.Hidayet Khadidja et Benali Amar Samia . « Algorithme génetique parallèle appliqué a un problème d'optimisation combinatoire , thèse de master, Université Moulay Tahar Saida ».(2017).

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

[3] A.Karray. « Contribution à l'ordonnancement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides , thèse de doctorat ,Ecole centrale de Lille Université de Tunis El Mmanar Ecole Nationale D'ingénieurs de TUNIS ». (2012)

[4] B.Merouane Hocine.«les Problemes d' Ordonnancement a Machines Paralleles, de Taches Dependantes, Universite Saad Dahlab de Blida».(2006)

[5] B.M. E., Optimisation à base de simulation pour le développement des syst-mes décisionnels, Thèse Doctorat 3éme cycle en informatique, université de M'sila.(2015)

[6] B.MOHAMMED, «Problèmes d'Ordonnancement d'Atelier, Memoire de fin d'etude, université Sidi Mohamed Ben Abdellah » .(2016)

[7] C. Abderrahman et Dahmani Abd Naceur«Ordonnancement d'un flow-shop par métaheuristique hybride , Mémoire de Fin d'Etudes , Université Abou Bekr Belkaid Tlemcen » . (2017)

[8] G.Abderrahim «Application d'une approche BIO-INSPIREE au Problème d' Ordonnancement des instructions, diplôme de Magistère, Université de M'sila,».(2012)

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

[10] H.Aboubakr et LEMMOUIS Abdelhamid«Résolution d'un problème d'ordonnancement de type job shop avec contrainte de transport, Projet de Fin d'Etudes, Université Abou Bekr Belkaid Tlemcen ».

[11]. I.LARIBI «Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables, Présentée pour l'obtention du grade de DOCTEUR , Université Tlemcen » .(2018)

[12] J.W. Herrman, & al. Handbaook of Production Scheduling, Springer NY.(2006.)

[13] K.Mebarek «Utilisation des stratégies Métaheuristiques pour l'ordonnancement des ateliers de type Job Shop, Mémoire de Magister**,** Université El-Hadj Lakhdhar atna» **.(**2008)

[14] M.C. Cooper . «Théorie de la complexité IRIT, Université de Toulouse 3» .

[15] M.Meziane 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)

[16] S.Ourari, « L'ordonnancement déterministe à l'ordonnancement , Thèse en vue de l'obtention du garde de docteur de l'Université de TOULOUSE».(2011)

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

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

[19] W.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)

[20] A.Gherboudj « Méthodes de résolution de problèmes difficiles académiques, Thèse du diplôme de Doctorat, Université de Constantine2». (2013)

Les sites web

[1] https://www.techno-science.net/glossaire-definition/Complexite.html#ref_1

[2] https://fr.wikipedia.org/wiki/Algorithme_glouton

الملخص

قمنا بالنظر في مشاكل الجدولة الصعبة ، والمهام المستقلة على أجهزة متوازية متطابقة ، من أجل تقليل وقت االاتمام االاكبر. يمكن إجراء العمليات بشكل متوازٍ على أجهزة متعددة. تم اقتراح طريقة تعتمد على الخوارزميات الجينية ، ثم الخوارزمية الجشعة لحل المشكلات.

الكلمات المفتاحية: الجدولة ، الآلات المتوازية المتطابقة ، المهام المستقلة ، الخوارزمية الجينية ، الخوارزمية الجشعة.

Résumé

Nous considérons les problèmes d' ordonnancement NP-difficiles, de tâches indépendantes sur des machines parallèles identiques, afin de minimiser la longueur d'ordonnancement appelée 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, tâches indépendantes, le makespan, algorithm génétique, algorithm glouton.

Abstract

We deal with the NP-hard problems of scheduling precedence constrained tasks on identical parallel processors expecting to minimize the schedule length or the makespan. The operations can be performed in parallel on multiple machines. A method was suggested based on genetic algorithms and , greedy algorithm, then it was used for solving the problem.

Key words: Scheduling, identical parallel machines, independent tasks, the makespan, genetic algorithm , Greedy algorithm.