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 : Nadir Ramla

SUJET

Un problème d'ordonnancement de type Job Shop dans un environnement dynamique

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

Dr.Moussaoui Adel	**Université de M'sila Président**
Dr. Nasser Eddine MOUHOUB	**Université de M'sila Rapporteur**
Dr.Boudaa Abdelghani	**Université de M'sila Examinateur**

Promotion : 2018 /2019

REMERCIEMENTS

Tout d'abord El-hamdoulillah, c'est Lui qui m'a donné la force et la patience pour réaliser ce mémoire.

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 réserve le dernier remerciement très chaleureux à mes parents et à toute ma famille pour leur présence constante à mes côtés et leur soutien.

Je tiens à remercier tous les enseignants et mes amis qui ont ma donnée des aides importantes.

Table des matières

Introduction générale 1

Chapitre I les problèmes d'optimisation combinatoire

1. Introduction 3
2. Définition 3
3. Complexité 3
   3.1Les class P et NP 4
      3.1.1 La classe P 4
      3.1.2 La classe NP-complet 4
      3.1.3La classe NP-difficile 4
4. La typologie des problèmes d'ordonnancement 5
5. Les éléments des problèmes d'ordonnancement 6
   5.1 Les tâches 6
   5.2Les opérations 7
   5.3Les ressource 7
   5.4Les contraintes 8
      5.4.1 Inégalité des potentielles 9
      5.4.2 Inégalité des disjonctions 10
      5.4.3 Les contraintes cumulatives 10
   5.5Les objectifs ou les critères d'évaluation (La fonction économique). 10
6. Notations et classification des problèmes d'ordonnancement 11
   6.1 Champ α : Organisation des ressources 11
   6.2 Champ β : les types de contraintes et caractéristiques du système 11
   6.3Champ γ : fonction objectives 11
7. La représentation du résultat 12

7.1Le diagramme de Gantt 12
7.2 Graphe Potentiel-Tâches 12

Chapitre II les méthodes de résolutions

1. Introduction 13
2. Les algorithmes classiques de résolution 13
2.1 Méthode des Potentiels Métra (MPM) 13
2.2 La méthode Pert (Program Evaluation and Review Technique) 14
3. Les algorithmes exacts 15
3.1 Procédure par Séparation et Évaluation (Branch and Bound) 15
3.2 Programmation dynamique 16
3.3 Programmation par contraintes 16
3.4 Programmation linéaire 17
4. Les méthodes approchées 17
4.1 Les heuristiques 18
4.2 Les méta-heuristiques 18
4.2.1 Méthodes à solution unique 18
4 .2 .1 .1 les méthodes de descente 18
4 .2 .1 .2 le recuit simulé 19
4 .2 .1 .3 Recherche tabou 19
4.2. 2 Méthodes à population de solutions 19
4 .2 .2 .1 Les algorithmes génétiques 20
4 .2 .2 .2 l'algorithme d'optimisation par essaim de particules PSO 21
4 .2 2 .3 l'optimisation par colonie de fourmis 23
4 .2 2. 4 l'optimisation par colonie d'abeille 24

Chapitre III Les problèmes des ateliers

1. Introduction 25
2. Définitions et notions fondamentales 25

3. Modèle de base 26
4. Notion d'objectifs et de critères 26
5. Les problèmes d'ordonnancement d'atelier 27
5.1 Les types de problème atelier 27
5.1.1 Problèmes à une machine 28
5.1.2 Problèmes à machines parallèles 28
5.2 Les problèmes d'atelier multi-machines 29
5.2.1 Problème flow shop 29
5.2.2 Problème flow shop avec permutation 30
5.2.3 Problème Job Shop 30
5.2.4 Problème Open Shop 31
5.3 La notion de flexibilité 31
5.3 .1 Flow shop hybrid 31
5.3.2 Job Shop hybride 32

Chapitre IV Problème job shop dans un environnement dynamique

1. Introduction 33
2. Système à étudier et le thème proposé 33
2.1 Définition formelle 33
2.2 Les machine 34
2.3 Les taches 35
2. 4 Représentations d'un problème job-shop 35
3. Formulation du problème 36
3.1 Formulation linéaire 36

3.2 Formulation mathématique 37

4. Les modèles du problème 38

4.1 Le modèle statique 38

4.1.1 Ordre de problème job shop 38

4.1.2 Algorithme de Johnson 38

4.1.3 Ordonnancement d'atelier à deux machines 38

4.1.4 Ordonnancement d'atelier à >2 machines 43

4.1.5 Calcul du Makespan (Cmax) 45

4.2 Modèle dynamique 46

4.2.1 Ordonnancement d'atelier en temps réel 46

4.2.2 Approches pour l'ordonnancement temps réel d'atelier 46

4.2.3 Méthode d'insertion de tâches 47

4.2.3.1 Définition d'une position admissible 47

4.2.3.2 Procédure de décalage 48

4.2.3.3 Création d'une position admissible et détermination du nouvel ordre 48

Chapitre V Conception et Réalisation

1. Introduction 49

2. les éléments matériels 49

3. les éléments logiciels 49

3.1 Le langage de programmation java 49

3.2 L'environnement NetBeans 50

4. Application de l'algorithme Johnson 51

5 .Interface graphique de l'application 55

5.1 Modèle statique 55

5.2 Modèle dynamique 57

Conclusion générale 61

List des figures

Figure I.1 : Les class P et NP 4
Figure I.2 : la représentation des types d'ordonnancement 5
Figure I.3 : la représentation de Caractéristiques d'une tâche i 6
Figure I.4 : La typologie des problèmes d'ordonnancement par les ressources 8
Figure I.5 : Représentation du schéma des contraintes 8
Figure I.6 : Représentation du schéma des contraintes de ressources 9
Figure I.7 : Représentation du schéma des contraintes potentielles 9
Figure I.8 : la représentation d'un exemple de diagramme de Gantt 12

Figure II.1 : la représentation d'un exemple du graphe MPM. 14
Figure II.2 : la représentation d'un exemple du graphe Pert 15
Figure II.3 : La représentation du fonctionnement de l'algorithme génétique 21
Figure II.4 : l'organigramme de la méthode des essaims particulaires 23
Figure II.5 : Classification des méthodes de résolution 25

Figure III.1 : La représentation d'ordonnancement d'atelier 25
Figure III.2 : Domaine concerné par l'ordonnancement 26
Figure III.3 : La représentation d'ordonnancement à une machine 28
Figure III.4 : La représentation d'ordonnancement à machines parallèles 29
Figure III.5 : La représentation à cheminement unique. 29
Figure III.6 : La représentation un flow-shop de permutation à 3 jobs et 4 machines. 30
Figure III.7 : La représentation d'ateliers à cheminement multiple (job-shop) 31
Figure III.8 : La représentation d'un flow show hybride à « k » étages 32
Figure III.9 ; La représentation d'un job shop hybride 32

Figure IV.1 : Un problème job shop simple et hybride 34

Figure IV.2 : diagramme de Gantt d'un problème job-shop 36

Figure IV.3 : premier algorithme de deux machines, la règle de Johnson 39

Figure IV.4 : exemple de premier algorithme de deux machines..... 32

Figure IV.5 : deuxième algorithme de la règle de Johnson 41

Figure IV.6 : Représentation de diagramme de Gantt de 2 machines 43

Figure IV.7 : Représentation de diagramme de Gantt de 3 machines 45

Figure IV.8 : Exemple d'une opération de décalage effectué pour insérer la tâche O_i^a 48

La figure V.1 : la page principale de NetBeans. 50

La figure V.2 : l'Algorithme de Procédure Convert..... 51

La figure V.3 : l'Algorithme de Procédure findminmax 52

La figure V.4 : l'Algorithme de Procédure insertion 53

Figure V.5 : Le schéma de réalisation de problème 54

La figure V.6 : L'interface principale de l'application. 55

La figure V.7 : la page de l'interface. 55

La figure V.8 : Un exemple applique sur l'interface. 56

La figure V.9 : Interface après exécution. 57

La figure V.10 : le diagramme de Gantt représentant la solution optimale 57

La figure V.11 : les durées de la nouvelle tâche. 58

La figure V.12 : interface après insertion statique. 58

La figure V.13 : le diagramme de Gantt après insertion statique 59

La figure V.14 : Le diagramme de Gantt après insertion statique..... 59

La figure V.15 : le diagramme de Gantt après insertion dynamique. 60

Liste des tableaux

Tableau II.1 : Exemple de méthode MPM. 14

Tableau III.1 : La représentation de la gamme opératoire des jobs. 30

Tableau IV.1 : La représentation les gammes opératoires des jobs 36

Tableau IV.2 : Une représentation l'ordre de passage des différents jobs 36

Tableau IV.3 : Présentation d'un exemple de premier algorithme de deux machines 41

Tableau IV.4 : La représentation d'un exemple de 3 machines 44

Tableau IV.5 : Présentation de trois machines réduite au cas de deux machines 44

Tableau IV.6 : Présentation l'ordonnancement optimal. 45

Tableau IV.7 : Présentation de calcul le Cmax d'un exemple 46

Introduction générale

La réactivité des ateliers de la production et le taux de satisfaction clients en termes de délais reposent pour toute ou partie sur la capacité de ces ateliers à faire face aux événements imprévus auxquels ils sont soumis.

La flexibilité en gestion de production confère aux ateliers cette capacité potentielle en utilisant des méthodes adéquates d'affectation et d'ordonnancement des tâches. Les industriels doivent donc maîtriser leur système de production au niveau opérationnel et être capables de réagir sur le très court terme aux événements imprévus tels qu'une modification ou une annulation d'un ordre de fabrication, l'arrivée d'une commande urgente.

Un problème d'ordonnancement est défini par un ensemble de travaux à réaliser sur un ensemble de ressources ; de sorte qu'une fonction objectif soit optimisée.

Le problème d'ordonnancement est réellement consiste à adapter en permanence les modalités d'exécutions d'ensemble des tâches par un ensemble des ressources à la situation réel du système considéré. On rencontre souvent ce type de problème dans les ateliers de fabrication qui travaille à la commande où le délai de livraison représente une des difficultés majeur.

Le problème de type Job Shop est l'un des problèmes de la théorie de l'ordonnancement le plus étudié et le plus difficile à résoudre. Il est indispensable de savoir, dans la résolution de ce type de problème, si l'on doit privilégier la qualité de la solution recherchée, la rapidité du temps de calcul ou trouver un compromis. La résolution de manière optimale s'avère donc dans la plupart des cas impossible à cause de son caractère fortement combinatoire. Alors on fait le recoure généralement aux méthodes dite méthodes approchées qui donnent des solutions approchées dans un temps raisonnable.

Dans notre projet, nous nous intéressons plus particulièrement aux problèmes d'ordonnancement en temps réel pour un system de type Job Shop dans un environnement dynamique. Ce système est soumis à une perturbation de l'environnement représentée par l'occurrence des nouvelles commandes urgentes, qu'il doit les exécuter. Le problème imposé ici est comment le système doit il réagir lors de l'apparition aléatoire d'une nouvelle tâche.

Ce mémoire est composé de cinq chapitres dont nous vous 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 la résolution des problèmes d'ordonnancement, leurs notations et leurs classifications.

Le deuxième chapitre présente les méthodes de résolution d'un problème d'ordonnancement qui analyse trois méthodes, les méthodes classiquesde résolution, les méthodes exactes et les méthodes approchées.

Le chapitre troisième présente les définitions et les notions fondamentales et aussi les critères de problème d'ordonnancement avec un rappel de quelques concepts sur les types d'atelier.

Dans le quatrième chapitre, on abordera le problème d'ordonnancement job shop, 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.

Les problèmes d'optimisation combinatoire

1. Introduction

Les problèmes d'ordonnancement font partie des problèmes d'optimisation combinatoire apparaissent dans tous les domaines de l'économie, l'informatique, la construction, l'industrie, l'administration .et considéré comme un objet de recherche intensive.

Dans ce chapitre on va vous présenter 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 la résolution des problèmes d'ordonnancement, leur notation et leurs classifications.

2. Définition

Un problème d'optimisation combinatoire (POC) est un problème qui consiste à rechercher une solution optimisée sur un ensemble de solution réalisable, Dans un POC on cherche un objectif (max, min...). Cet objectif étudie un ensemble de modèles et de méthodes de résolution (gloutonnes, programmation dynamique, métaheuristique ….).

Un facteur majeur qui permet la classification des problèmes d'ordonnancement c'est la complexité.

3. Complexité

La théorie de la complexité est la mesure de la solution de POC qu'elle a été développée mathématiquement. On distingue deux types de problèmes : problème facile et problème difficile. [13]

- Lorsque on dit un problème facile si en le résoudre par un algorithmeefficace.
- un algorithme est dit efficace est un algorithme qui permet de résoudre un problème en même temps d'exécution qui ne soit pas tropimportant.
- Si on ne connait pas un algorithme efficace pour un problème alors il est dit difficile.
- la complexité d'un algorithme (A) est une fonction $C(_{An})$ qui donne le nombre d'instructions exécutées par A dans le pire des cas pour une donnée de taillen.
- la complexité est notée d'un algorithme parO.
- un problème est dit décidable lorsque il existe ou moins un algorithme pour

lerésoudre.

- le problème est dit non décidable lorsqu'il n'a aucun algorithme trouvé lerésoudre.
- dans les problèmes décidables on trouve la familleNP.

La classe NP est la classe des problèmes décidables, elle est décomposée on 3 partie.

3.1 Les class P et NP

3.1.1 La classe P

C'est la classe qui rejoint tous les problèmes, ont des algorithme des solution de complexité polynomiale, 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. [13]

3.1.2 La classe NP-complet

Elle contient le problème du plus difficile de la class NP, elle contient la classe NP tell que n'importe quelle problème de la classe NP leur est polynomialement réductible.

3.1.3 La classe NP-difficile

Contient les problèmes pas forcement de la classe NP telle que un problème d'optimisation est dit NP-Difficile, si le problème de décision associé est NP-complet.

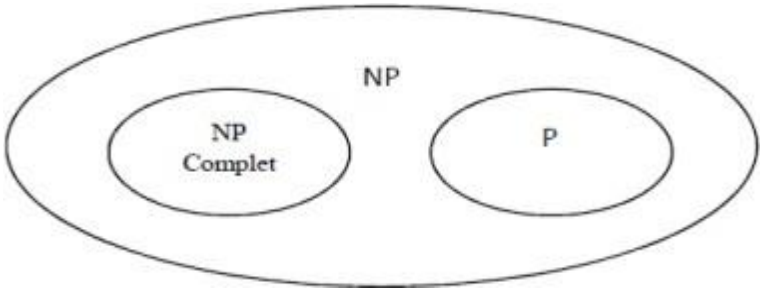


Figure 1.1 : Les class P et NP tirée de [13].

4. La typologie des problèmes d'ordonnancement

Selon la nature des variables lise en jeux, la nature des contraintes…. plusieurs classifications des problèmes d'ordonnancement sont proposées ou leurs complexité et de l'ordre de NP difficile une typologie des problèmes peut être dressée dans l'architecture suivante :

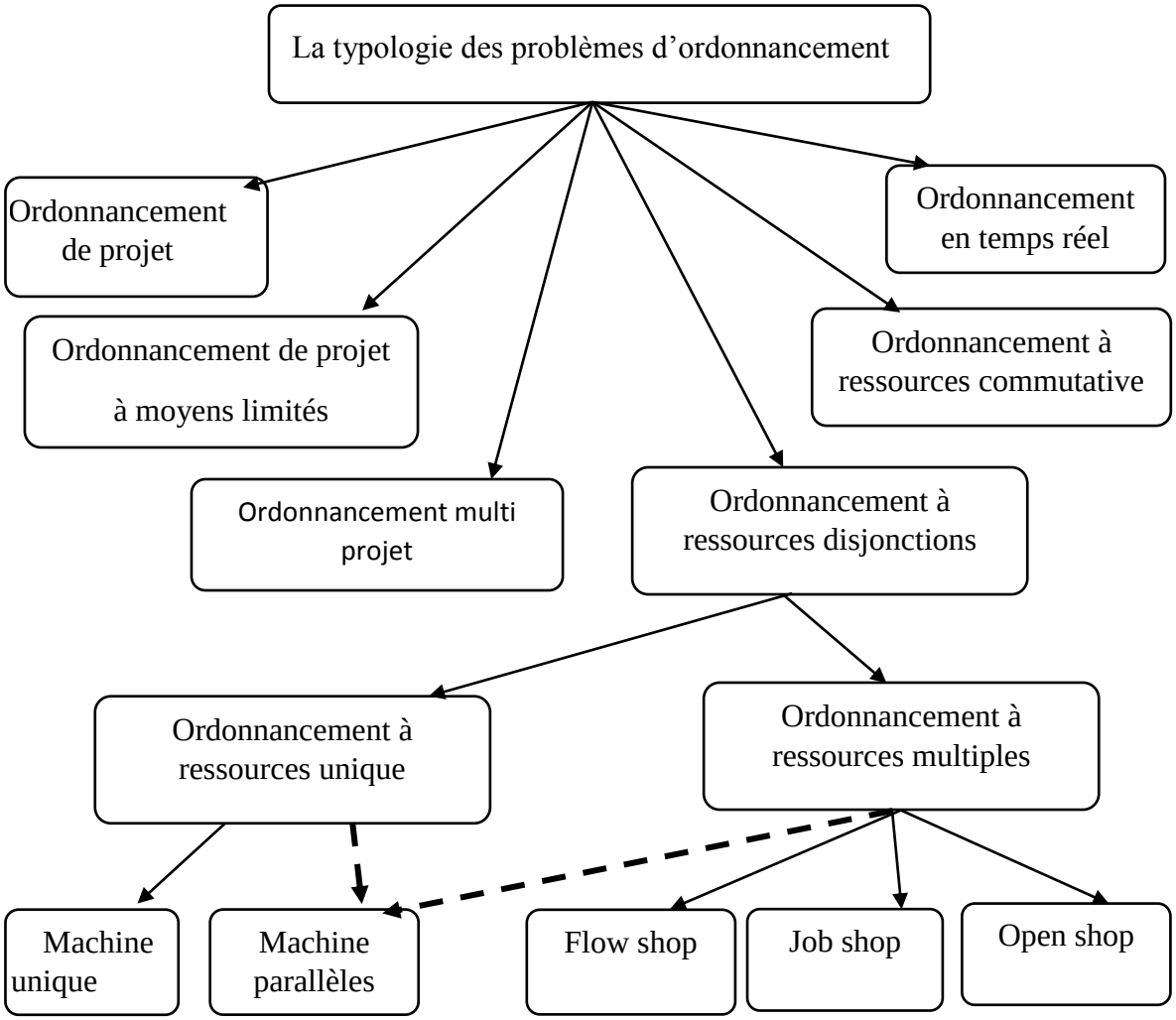


Figure 1.2 : la représentation des types d'ordonnancement.

5. Les éléments des problèmes d'ordonnancement

Un problème d'ordonnancement est composé d'un ensemble défini d'éléments, on trouve ces composants dans tous les problèmes d'ordonnancement :

5.1 Les tâches

Une tâche est un ensemble d'opérations (programme) mobilisant des ressources et réalisant un progrès significatif de l'état d'avancement du projet compte tenu du niveau de détail retenu dans l'analyse du problème. [19]

Une tâche est une entité élémentaire de travail démarquée dans le temps par une date de début de la tache t_i ou de fin de la tache c_i, dont la réalisation est caractérisée par une durée de traitement $p_i = c_i - t_i$, et par l'intensité aik avec laquelle elle consomme certains moyens k.

Il y a 3 types de tâche : séquentielle, parallèle, convergente.

Généralement, les notations utilisées pour caractériser une tâche, qu'on note 'i', sont les suivantes :

- Une date de disponibilité r_i : l'exécution de la tâche i ne peut pas débuter avant cette date.
- Une date échue notée d_i : la tâche i doit être achevée avant cette date.
- La durée opératoire de traitement- notée p_i.

On note $[r_i, d_i]$, l'intervalle temporel dans lequel la tâche i devrait s'exécuter.

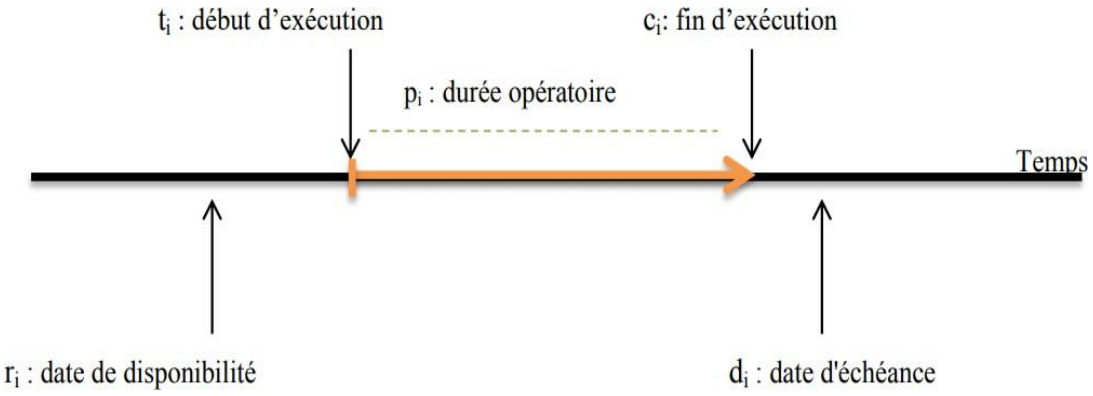


Figure 1.3 : la représentation de Caractéristiques d'une tâche i. [12]

5.2 Les opérations

Généralement la plupart des tâches, donnent lieu à l'utilisation de différentes ressources et se décomposent donc en plusieurs opérations. Dans le cas d'un problème d'ordonnancement sur une machine unique ou atelier unique, on parle de tâches mono-opération. L'opération est caractérisée par son temps d'exécution, appelé aussi durée opératoire. Parfois, une opération peut également être caractérisée par son temps de préparation sur la machine, ou encore les temps de montage et de démontage de la pièce (matérialisant l'opération ou l'outillage), les temps de transport et d'attente. La séquence des opérations est décrite dans ce qu'on appelle la gamme d'une tâche. [18]

5.3 Les ressource

Une ressource k est un moyen technique et/ou humain, destiné à être utilisé pour une réalisation d'une tâche et disponible en quantité limitée, sa capacité a_{ik} (supposée constante). On distinguer plusieurs types de ressources sont [18] :

- Ressources renouvelables si elles sont utilisées par une ou plusieurs tâches, elles seront à nouveau disponibles en même quantités (les hommes, les machines, l'espace, l'équipement en général, …etc.), les quantités des ressources utilisables à chaque instant sont limitées.

- Ressources consommables lorsque on utilisée pour la réalisation d'une ou plusieurs tâches à la fois elles ne sont pas disponible telle que (matière première, budget,…etc), la consommation globale (ou cumul) au cours du temps est limitée.

- Ressources doublements contraintes lorsque son utilisation instantanée et sa consommation globale sont toutes les deux limitées (source d'énergie, financement,… etc.).

- Ressources disjonctives (ou non partageables) principalement dans le cas de ressources renouvelables, sont des ressources qui ne peuvent exécuter qu'une tâche à la fois (machine-outil, robot manipulateur, … etc.).

- Ressources cumulatives (ou partageables) qui peuvent être utilisées par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail) [18].

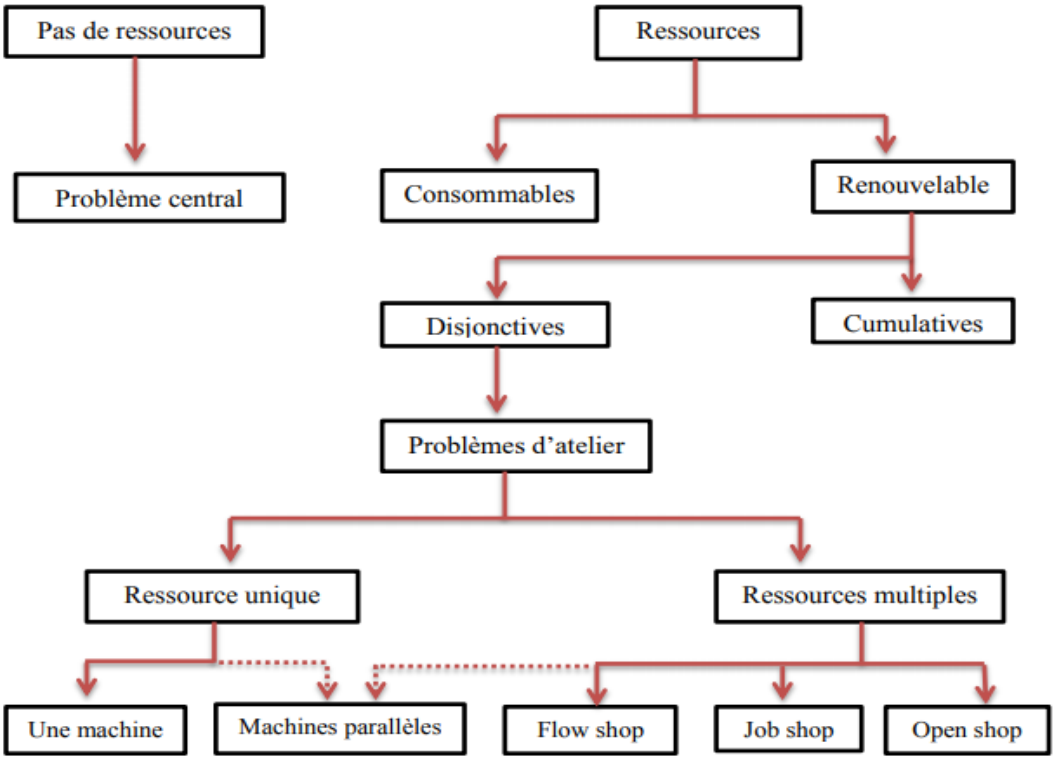


Figure 1.4 : La Représentation typologie des problèmes d'ordonnancement par les ressources tirée de [18].

5.4 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décisions.

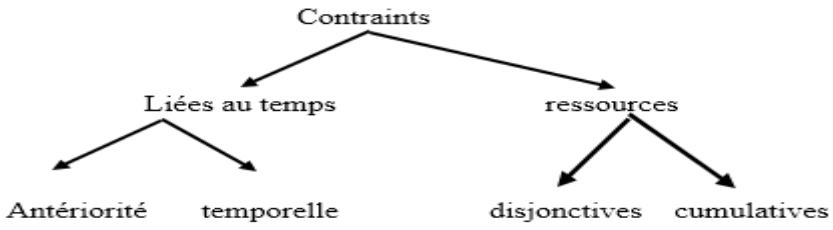


Figure 1.5 : Représentation du schéma des contraints

- Contrainte Liées au temps :
  - Les contraintes Antériorité décrivent les positionnements des taches qui doivent être respectés.
  - Les contraintes temporelles (de la durée) c'est une restriction sur l'exécution

de la tâche et qui dépend des temps.

- Contraintes d'enchaînement :

Nous qualifions de contrainte d'enchaînement ou de succession, une contrainte qui lie le début ou la fin de deux activités par une relation linéaire. Ce sont des contraintes imposées généralement par la cohérence technologique (les gammes opératoires dans le cas d'ateliers) qui décrivent des positionnements relatifs devant être respectés entre les tâches. [1]

- Contrainte ressources :

Exprime la nature et la quantité des moyens utilise des taches disponibles au cours du temps ainsi que la caractéristique de l'utilisation.

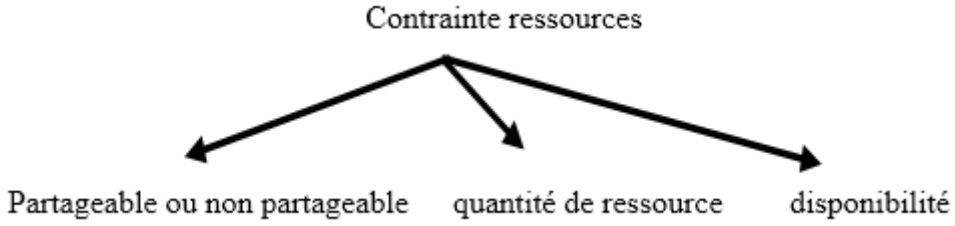


Figure 1.6 : Représentation du schéma des contraints ressources

5.4.1 Inégalité des potentielles

Soit deux taches ij une inégalité des potentielles et de la forme t_i-t_j>(ou égale) bij (b_{ij} constante réelle). Dans la contrainte antériorité généralement i<j (i avant j) (i précède j) b_{ij}> 0.

Soit ts la date de référence associé au début de projet ou prise pour origine des temps.

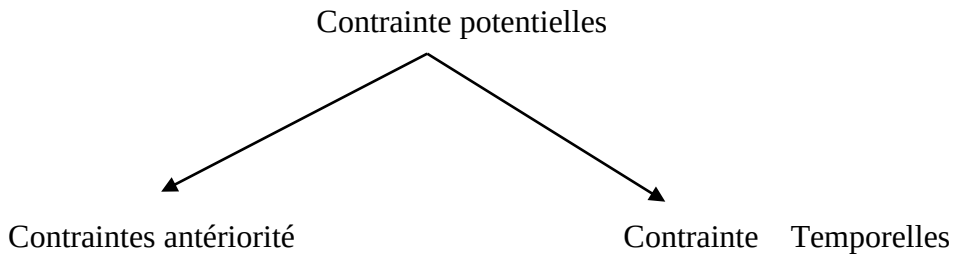


Figure 1.7 : Représentation du schéma des contraints potentielles

5.4.2 Inégalité des disjonctions

Les contraints disjonctives sont partie des contrainte de limitation de ressources renouvelable. Elles obligent à réaliser toute paire de taches sur les intervalles de temps disjoints.

5.4.3 Les contraintes cumulatives

Les contraints cumulatives font partie des contraints de limitation de ressources renouvelables.

Elles interdisent la réalisation simultanée d'un nombre trop important deux taches compte tenu la disponibilité maximal de la ressource à chaque instant et des quantités requise individuellement.

5.5 Les objectifs ou les critères d'évaluation (La fonction économique)

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi [18]. Les critères qui doivent satisfaire un ordonnancement sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement.

- Les objectifs liés au temps : On trouve par exemple la minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales de réglage ou des retards par rapport aux dates de livraison.
- Les objectifs liés aux ressources : maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches sont des objectifs de ce type.
- Les objectifs liés au coût : ces objectifs sont généralement de minimiser les coûts de lancement, de production, de stockage, de transport, etc. [1]

On peut disposer les critères en critères réguliers et en critères irréguliers :

- Les critères réguliers : un critère est dit régulier s'il est une fonction décroissante des dates de fin d'exécution des opérations. Par exemple, la durée globale de fabrication d'un produit [18], minimisation du maximum des retards sur les dates

d'achèvement, minimisation de la moyenne des retards sur les dates d'achèvement.

- Les critères irréguliers : Ils ne sont pas des fonctions monotones des dates de fin d'exécution des opérations. Parmi ces types de critères, on cite :

  -La minimisation des encours ;

  -La minimisation du coût du stockage des matières premières ;

  -Délai de fabrication, avances, retard,

  -L'équilibrage des charges des machines. [7]

6. Notation et classification des problèmes d'ordonnancement

Etant donné la différence des problèmes d'ordonnancement, nous utilisons couramment un formalisme de classification, permettant de distinguer les problèmes d'ordonnancement entre eux et de les classer. Ce formalisme, comporte trois champs : α, β, γ. Permettant de décrire les différentes entités d'un problème d'ordonnancement.

6.1 Champ α : Organisation des ressources

Le champ α décrit la structure des problèmes ordonnancement et se décompose on deux sous champs : a1 indique la nature des problèmes et a2 désire le nombre des machines.

Le paramètre α1 ∈ {1 ou ∅, P, Q, R, O, F, J, FH, JG, OG} caractérise le type de machines utilisées. Le paramètre α2 ∈ {∅, k} caractérise le nombre de machines utilisées dans le problème.

6.2 Champ β : les types de contraintes et caractéristiques du système

Le second champ β = β1 β2 β3 β4 β5 β6 β7 β8 décrit les types de contraintes prises en compte et les caractéristiques de la ressource.

6.3 Champ γ : fonction objectif

Le troisième champ, le champ γ, indique les fonctions objectivesconsidérées ou les descriptions des critères d'évaluation d'un ordonnancement. Les objectifs visés sont liés à une bonne utilisation des ressources, une minimisation du délai global ou encore le respect d'un

maximum de contraintes. Nous donnons ici les critères les plus couramment utilisés :Cmax (Makespan), TMax, …

7.La représentation du résultat

7.1 Le diagramme de Gantt

Le graphique de Gantt, encore appelé diagramme de Gantt, est une technique de visualisation de l'utilisation de moyens productifs et/ou de l'avancement de l'exécution de tâche [20].

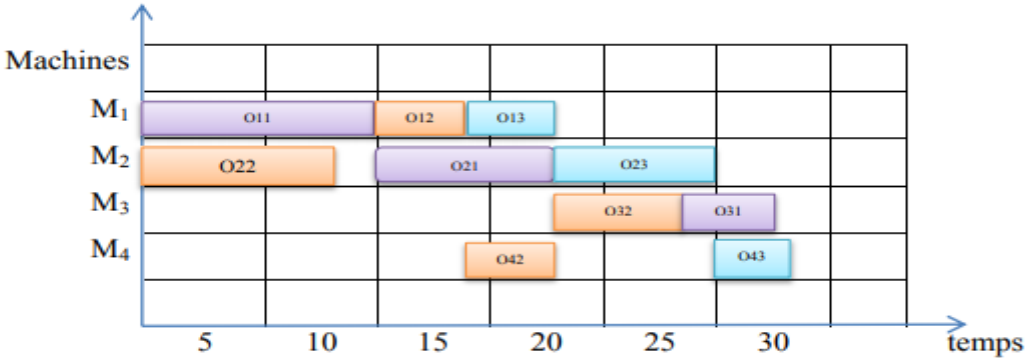


Figure 1.8 : la représentation d'un exemple de diagramme de Gantt. [20]

7.2 Graphe Potentiel-Tâches

Ce graphe a été développé grâce à la théorie des réseaux de Pétri qui ont particulièrement servi à modéliser les systèmes dynamiques à évènements discrets [7].

Les méthodes de résolutions

1. Introduction

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

Dans ce chapitre nous présentons les méthodes de résolution d'un problème d'ordonnancement qui sont trois méthodes : les méthodes classiques, les méthodes exactes et les méthodes approchées et qui sont défini comme suit :

2. Les algorithmes classiques de résolution

Les problèmes appartenant à la classe P ont des algorithmes efficaces et sont de complexité polynomiale telle que la méthode de chemin critique (CPM, MPM, PERT).

2. 1 Méthode des Potentiels Métra (MPM)

La méthode MPM a été Crée en 1958 par le chercheur français Bernard Roy, elle utilise systématiquement des relations d'ordre initiales [8].

Elle fait partie des méthodes potentiel-tâches où les Activités (tâches) sont représentées par les relations d'ordre entre activités successives par des arcs.

Le principe de MPM qui est déterminé comme suit :

- Les tâches sont représentées par des sommets et les contraintes de succession par des arcs.
- Chaque tâche est renseignée par la date à laquelle elle peut commencer (date au plus tôt) et celle à laquelle, elle doit se terminer (date au plus tard).
- A chaque arc est associée une valeur numérique, qui dessine soit une durée d'opération, soit un délai.

La rénovation du séjour d'un appartement se décompose en plusieurs tâches décrites dans le tableau ci-dessous. Ce dernier donne également les précédences à respecter lors de la planification des travaux ainsi qu'une estimation de la durée de chacune des tâches. [8]

Tâches		Durée (en jours)	Antériorité
A	Enlèvement des portes	1	-
B	Ponçage et peinture des portes	3	A
C	Pose des portes	1	A
D	Tirage des fils électriques	1	A
E	Pose des prises	1	B,D
F	Ragréage des murs	3	B,C
G	Peinture du plafond	2	B
H	Peinture des cadres	1	F,G
I	Peinture du balcon	2	E,H

Tableau 2.1 : un exemple de méthodes MPM. [8]

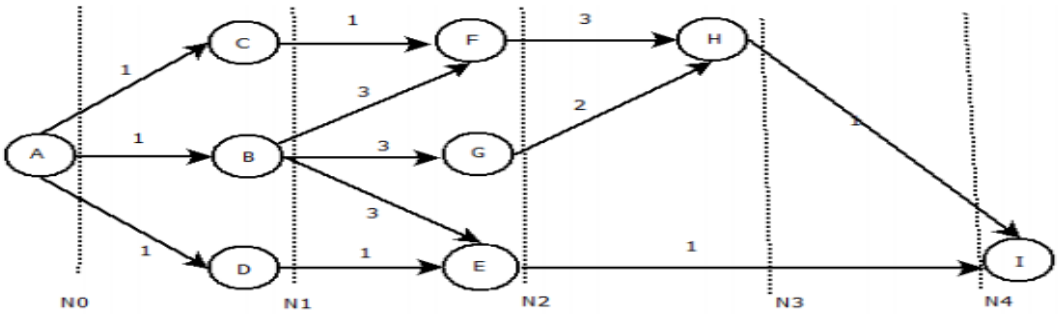


Figure 2.1 : la représentation d'un exemple du graphe MPM. [8]

2.2 La méthode PERT (Program Evaluation and Review Technique) :

La méthode PERT est une technique permettant de gérer l'ordonnancement dans un projet. Elle a été créée par la marine américaine dans les années 1950. [8]La méthode PERT consiste à représenter sous forme de graphe, un réseau de tâches dont l'enchaînement permet d'aboutir à l'atteinte des objectifs d'un projet.

Le but est de trouver la meilleure organisation possible pour qu'un projet soit terminé dans les meilleurs délais, et d'identifier les tâches critiques, c'est-à-dire les tâches qui ne doivent souffrir d'aucun retard sous peine de retarder l'ensemble du projet.

Ainsi, la méthode PERT implique : un découpage précis du projet en tâches. Pour chaque tâche, sont indiquées une date de début et de fin au plus tôt et au plus tard ; l'estimation du la durée de chaque tâche ; la nomination d'un chef de projet chargé d'assurer le suivi du projet,

De rendre compte si nécessaire et de prendre des décisions en cas d'écart par rapport aux prévisions.

Le principe Dans un graphe PERT : chaque tâche est représentée par un arc, auquel on associe un chiffre entre parenthèses qui représente la durée de la tâche. Entre les arcs figurent des cercles appelés « sommets » ou « événements » qui marquent l'aboutissement d'une ou plusieurs tâches. Ces cercles sont numérotés afin de suivre l'ordre de succession des divers évènements [8].

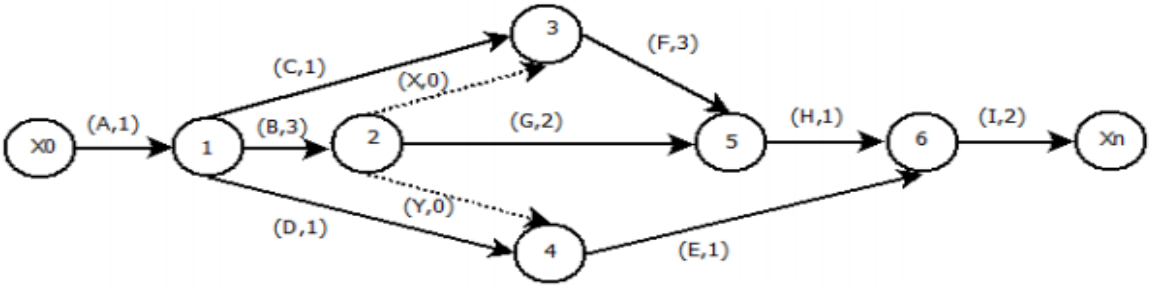


Figure 2.2 : la représentation d'un exemple du graphe Pert. [8]

3. Les algorithmes exacts

Les méthodes exactes font partie des méthodes génériques qui permettent de trouver une solution optimale à un problème donné. Toutefois, ces méthodes sont généralement utilisées pour résoudre des problèmes de petite taille et peuvent devenir rapidement couteuses en temps d'exécution, notamment pour les problèmes NP-difficiles.

En effet, le temps de traitement et la complexité du problème sont généralement liés (plus c'est complexe, plus le temps d'exécution sera important). Parmi ces méthodes, on peut citer : Procédure par Séparation et Évaluation, Programmation dynamique, Programmation par contraintes, Programmation linéaire.

3. 1 Procédure par Séparation et Évaluation (Branch and Bound)

L'algorithme Branch and Bound (Séparation et Évaluation) consiste à placer progressivement les tâches sur les ressources en explorant un arbre de recherche décrivant toutes les combinaisons possibles. Il se procède 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 réalise 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, éliminer les mauvais sous-espaces, et reproduire les étapes précédentes jusqu'à l'obtention de l'optimum global.

3. 2 Programmation dynamique

La programmation dynamique base sur le principe de Bellman [Bellman, 86] : « Si 'C'est un point qui appartient au chemin optimal entre A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C ». C'est une méthode qui consiste donc à construire d'abord les sous chemins optimaux et ensuite par récurrence le chemin optimal pour le problème entier. Cette méthode est destinée à résoudre des problèmes d'optimisation à vocation plus générale que la méthode de séparation et d'évaluation (branch and bound) sans permettre pour autant d'aborder des problèmes de tailles importantes.

Le principe de la méthode dynamique c'est composer une solution optimale du problème en combinant les solutions (optimales) de ses sous problèmes. En pratique : décomposer le problème en des sous-problèmes plus petits, calculer les solutions optimales de tous ces sous problèmes et les garder en mémoire et enfin, calculer la solution optimale à partir des solutions optimales des sous-problèmes.

3. 3 Programmation par contraintes

(PPC, ou CP pour constraint programming en anglais) est un paradigme de programmation apparu dans les années 1970 et 1980 permettant de résoudre des problèmes combinatoires de grandes tailles tels que les problèmes de planification et d'ordonnancement. En programmation par contraintes, on sépare la partie modélisation à l'aide de problèmes de satisfaction de contraintes (ou CSP pour Constraint Satisfaction Problem), de la partie résolution dont la particularité réside dans l'utilisation active des contraintes du problème pour réduire la taille de l'espace des solutions à parcourir (on parle de propagation de contraintes) [a].

3. 4 Programmation linéaire

La programmation Linéaire fait partie des techniques classiques de recherche opérationnelle. Elle repose sur la méthode du simplexe.Le problème mathématique consiste à optimiser (maximiser ou minimiser) une fonction linéaire de plusieurs variables qui est reliées par des relations linéaires appelées contraintes.

4. les méthodes approchées

Dans la majorité des cas où les problèmes sont NP-difficile de grande taille, on cherche à les simplifier pour réduire le volume des calculs. Bien entendu, cela se fait au prix d'une dégradation de la qualité de la solution.

Il existe plusieurs familles de méthodes qui permettent d'approcher efficacement la solution optimale. Parmi ces méthodes on trouve les méthodes heuristiques et les méta-heuristiques.

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

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

4. 1 Les 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 [2].

4. 2 Les méta-heuristiques

Un algorithme de résolution métaheuristique est un algorithme heuristique "générique" qu'il faut ajuster à chaque problème.Une métaheuristique est une heuristique généraliste, ils peuvent être appliqués à de nombreux problèmes d'optimisation.Le but d'une heuristique et réussir à trouver un optimal globale pour cela, l'idée est à la fois de parcourir l'espace de recherche, d'exploit les zones qui percent prometteuse mais sans être piégé Par un optimum local. Elles sont inspirées des processus naturels. La classification habituelle des méta-heuristiques : en fonction du nombre de solution qu'ils traitent :

4 .2 .1 Méthodes à solution unique

Les méta-heuristiques à base de solution unique débutent la recherche avec une seule solution initiale. Elles se basent sur la notion du voisinage pour améliorer la qualité de la solution courante. En fait, la solution initiale subit une série de modifications en fonction de son voisinage. Le but de ces modifications locales est d'explorer le voisinage de la solution actuelle afin d'améliorer progressivement sa qualité au cours des différentes itérations [10].

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

4.2.1.1 les méthodes de descente

La méthode de descente c'est une méthode de recherche locale les plus simples. Elle consiste rechercher à chaque étape la meilleure solution voisine de la solution courante. C'est une solution de coût très faible. Ce procédé est répété aussi longtemps que la valeur de la fonction objective diminue. Les méthodes de descente ont toujours compté parmi les méthodes heuristiques les plus populaires pour traiter les problèmes d'optimisation combinatoire. Cette méthode a l'avantage d'être rapide, mais s'arrête dès qu'un optimum local est atteint, même si celui-ci n'est pas de bonne qualité.

Toutefois elles comportent deux obstacles majeurs qui limitent considérablement leur

efficacité : suivant la taille et la structure du voisinage (s) considéré, la recherche de la meilleure solution voisine est un problème qui peut être aussi difficile que le problème (P) initial ; une méthode de descente est incapable de progresser au-delà du premier minimum local rencontré. Or les problèmes d'optimisation combinatoire comportent typiquement de nombreux optimums locaux pour lesquels la valeur de la fonction objective peut être fort éloignée de la valeur optimale [14].

4.2.1.2 Le recuit simulé

Le recuit simulé est une métaheuristique s'inspirant de la métallurgie créée au début des années 1980. C'est une méthode bien éprouvée. Elle a été présentée par S. Kirkpatrick,C. Gelatt et M. Vecchi dans. C'est une méthode empirique (métaheuristique) destinée à résoudre au mieux les problèmes dits d'optimisation. Cette méthode est une technique de recherche locale inspirée du processus utilisé en métallurgie utilisant la température T pourguider la recherche en passant d'une solution à sa voisine.

4.2.1.3 Recherche tabou

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

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

4 .2. 2 Méthodes à population de solutions

Les méta-heuristiques à base de population de solutions débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure (l'optimum global) qui représentera la solution du problème traité.

L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité.

Une grande variété de méta-heuristiques basées sur une population de solutions a été proposée dans la littérature, algorithmes génétiques, et les algorithmes à base d'intelligence par essaims : l'algorithme d'optimisation par essaim de particules, l'algorithme de colonies de fourmis et l'algorithme de colonies d'abeille [10].

4.2.2.1 Les algorithmes génétiques

Les algorithmes génétiques utilisent la théorie de Darwin sur l'évolution des espèces. Elle repose sur trois principes : le principe de variation, le principe d'adaptation et le principe d'hérédité. [b]

Le principe de variation :

Chaque individu au sein d'une population est unique. Ces différences, plus ou moins importantes, vont être décisives dans le processus de sélection.

Le principe d'adaptation :

Les individus les plus adaptés à leur environnement atteignent plus facilement l'âge adulte. Ceux ayant une meilleure capacité de survie pourront donc se reproduire davantage.

Le principe d'hérédité :

Les caractéristiques des individus doivent être héréditaires pour pouvoir être transmises à leur descendance. Ce mécanisme permettra de faire évoluer l'espèce pour partager les caractéristiques avantageuse à sa survie.

Il y a trois opérateurs d'évolution dans les algorithmes génétiques :

La sélection : La sélection consiste à choisir les individus les mieux adaptés afin d'avoir une population de solution la plus proche de convergence vers l'optimum global.Cet opérateur est l'application du principe d'adaptation de la théorie de Darwin.

Le croisement : Le croissement ou enjambement, crossing-over, est le résultat obtenue lorsque deux chromosomes partage leurs particularités.Celui-ci permet le brassage génétique

de la population et l'application du principe d'hérédité de la théorie de Darwin.

La mutation : La mutation consiste à altérer un gène dans un chromosome selon un facteur de mutation. Ce facteur est la probabilité qu'une mutation soit effectuée sur un individu. Cet opérateur est l'application du principe de variation de la théorie de Darwin et permet, par la même occasion, d'éviter une convergence prématurée de l'algorithme vers un extremum local.

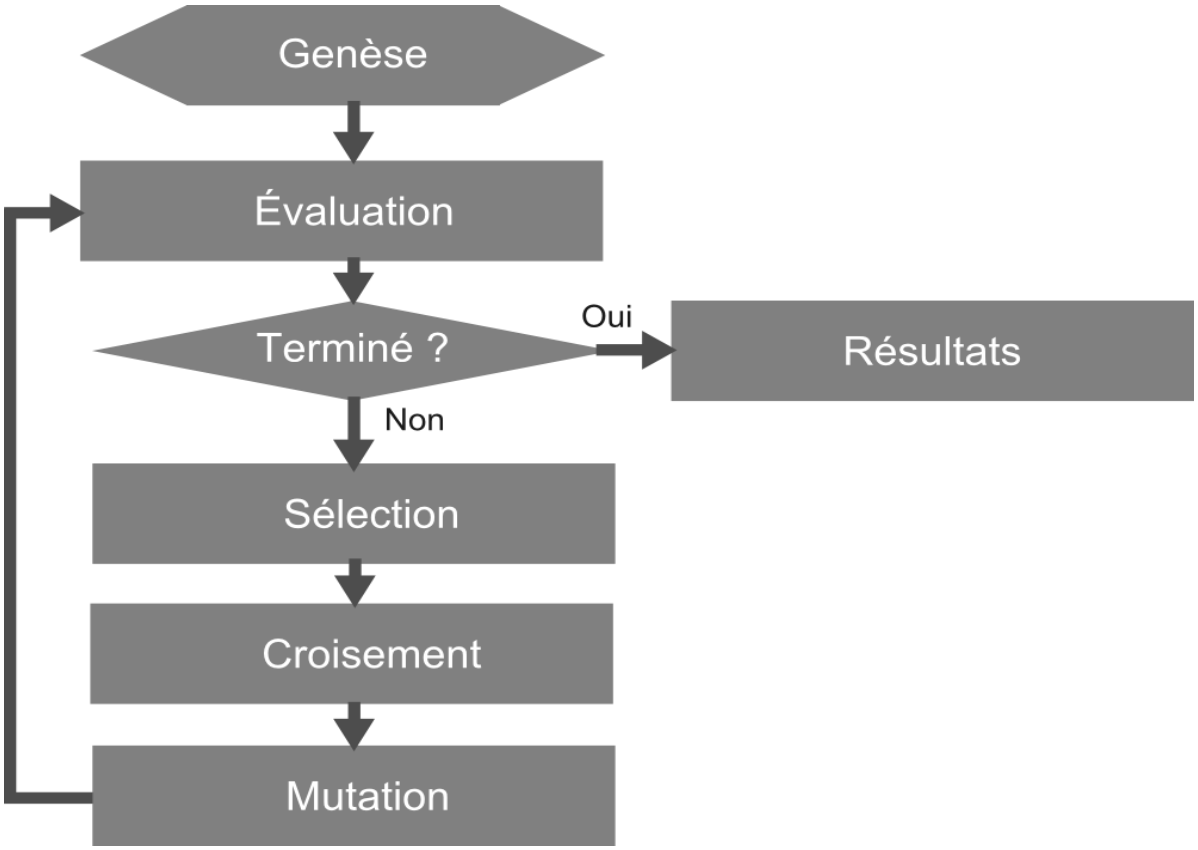


Figure 2.3 : La représentation du fonctionnement de l'algorithme génétique. [b]

4.2.2.2 Algorithme d'optimisation par essaim de particules PSO

La méthode L'optimisation par essaim de particule a été Crée en 1995 par le Russel Eberhart (ingénieur en électricité) etJames Kennedy (socio-psychologue). PSO une méthode d'optimisation stochastique, pour des fonctions non _linéaire, basé sur la reproduction d'un comportement social.Elle est utilisée pour explorer l'espace de recherche d'un problème quelconque pour trouver l'ensemble des paramètres qui (maximiser /minimiser) un objet particule.

PSO est une technique évolutionnaire qui utilise (une population) de solution candidates pour développer une solution optimale au problème d'optimisation. Le degré d'optimalité est mesuré par une fonction fitness définie par l'utilisateur. Chaque particule se déplace et à chaque itération, la plus proche de l'optimum communique aux autres sa position pour qu'elles modifient leur trajectoire. Cette idée veut qu'un groupe d'individus peu intelligents puisse posséder une organisation globale complexe.

Le principe d'essaim particule de chaque particule est en train de :

- D'une position, c'est-à-dire ses coordonnées dans l'ensemble de définition.
- D'une vitesse qui permet à la particule de se déplacer. De cette façon, au cours des itérations, chaque particule change de position. Elle évolue en fonction de son meilleur voisin, de sa meilleure position, et de sa position précédente. C'est cette évolution qui permet de tomber sur une particule optimale.
- D'un voisinage, c'est-à-dire un ensemble de particules qui interagissent directement sur la particule, en particulier celle qui a le meilleur critère.
- A tout instant, chaque particule connait :
- Sa meilleure position visitée. On retient essentiellement la valeur du critère calculée ainsi que ses coordonnées.
- La position du meilleur voisin de l'essaim qui correspond à l'ordonnancement optimal.
- La valeur qu'elle donne à la fonction objectif car à chaque itération il faut une comparaison entre la valeur du critère donnée par la particule courante et la valeur optimale [15].

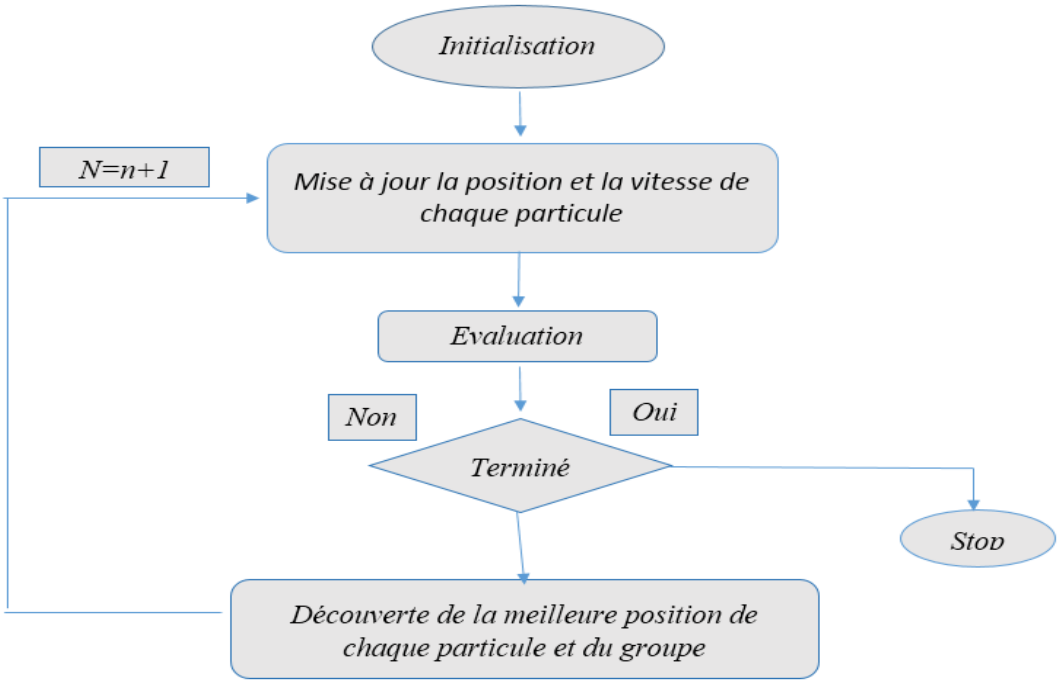


Figure 2.4 : La représentation de l'organigramme de la méthode des essaims particulaires [15].

4.2.2.3 Optimisation par colonie de fourmis

L'optimisation par colonie de fourmis (ACO) a été proposée en 1992 par Colorni, c'est une méthode d'optimisation fondée sur la manipulation d'un ensemble de solutions. Cette méthode représente toute une classe des méta-heuristiques qui reposent sur la notion de l'intelligence courante. La solution finale est plus complexe que celle d'un composant simple.

Le principe d'une optimisation par colonie de fourmis chaque fourmi est mise aléatoirement sur une ville et elle a une mémoire qui stocke la solution partielle qu'elle a construit jusqu'ici (au commencement la mémoire contient seulement la ville du début). À partir de sa ville de début, une fourmi se déplace itérativement d'une ville vers une autre mais avec une règle de probabilité.

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

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

4.2.2.4 Optimisation par colonie d'abeille

L'une des insectes les plus organisées et les plus rigoureuses dans leur travails est l'abeille. Les abeilles possèdent une très grande capacité de communication. Et grâce à son intelligence, une méthode appelée méthode des abeilles a été développer. Dans cette méthode, les abeilles artificielles représentent des agents qui en collaborant les unes avec les autres, résolvent des problèmes complexes d'optimisation combinatoire.

On tire de cette information l'idée de base de cette méthode : créer un système multi agent capable de résoudre avec succès les problèmes complexes.

En résumé, nous présentons dans la figure suivante (figure 3.5) une classification des Classification des méthodes de résolution des problèmes d'ordonnancement :

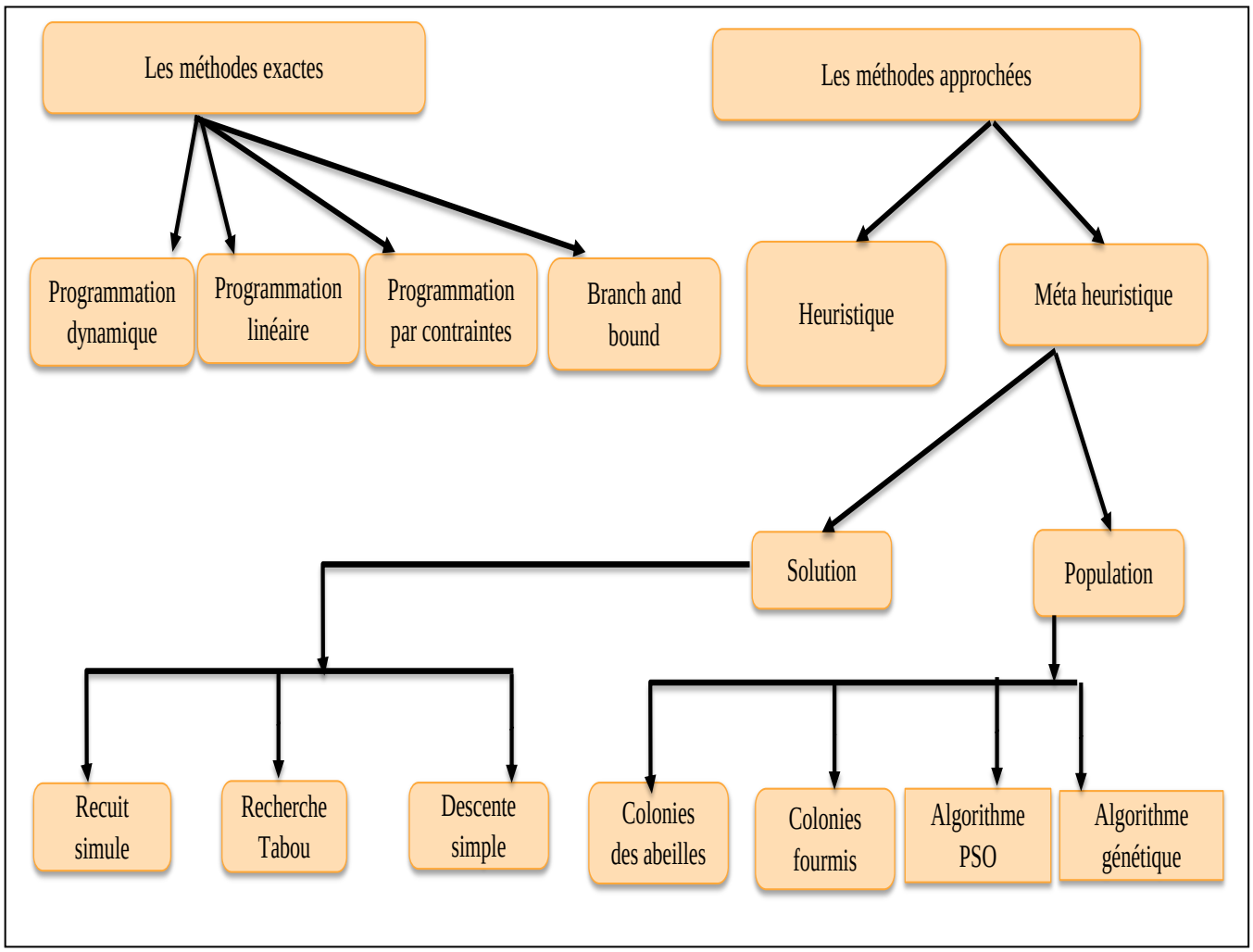


Figure 2.5 : Classification des méthodes de résolution des problèmes d'optimisation combinatoire.

Les problèmes des ateliers

1. Introduction

Dans un atelier spécialisé, les problèmes d'ordonnancement des ressources qui sont des machines ne réalisent qu'une tâche (opération) à la fois. Ce sont des problèmes à ressource disjonctives. Chaque travail concerne une entité physique invisible appelée : produit ou lot lorsque plusieurs produits identiques sont regroupés.

Un produit ne peut pas se trouver en deux lieux différents à la fois et un même travail ne peut être exécuté qu'une seule opération à la foi. Pour chaque travail il y'a un ordre strict.

Dans ce chapitre on va présenter les définitions, les notions fondamentales et les critères des problèmes d'ordonnancement avec rappel de quelques concepts sur les types d'ateliers.

2. Définitions et notions fondamentales

Un problème d'ordonnancement peut être considéré comme un sous problème de planification dans lequel il s'agit de décider de l'exécution opérationnelle des tâches (jobs) planifiées, et ainsi d'établir leur planning d'exécution et leur allouer des ressources visant à satisfaire un ou plusieurs objectifs sous une ou plusieurs contraintes.

En se basant sur les concepts de tâche, ressource, contrainte et objectif, l'ordonnancement peut également être déterminé comme :

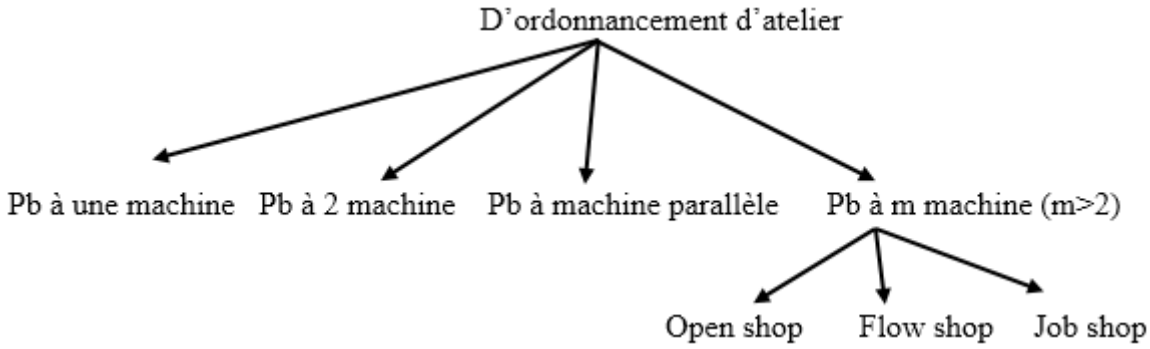


Figure 3.1 : La représentation d'ordonnancement d'atelier

Les problèmes d'ordonnancement existent dans de nombreux secteurs d'activités comme la gestion de production, dans l'industrie manufacturière on encore les systèmes informatiques, la figure 3.2 montre les différents domaines concernés par l'ordonnancement.

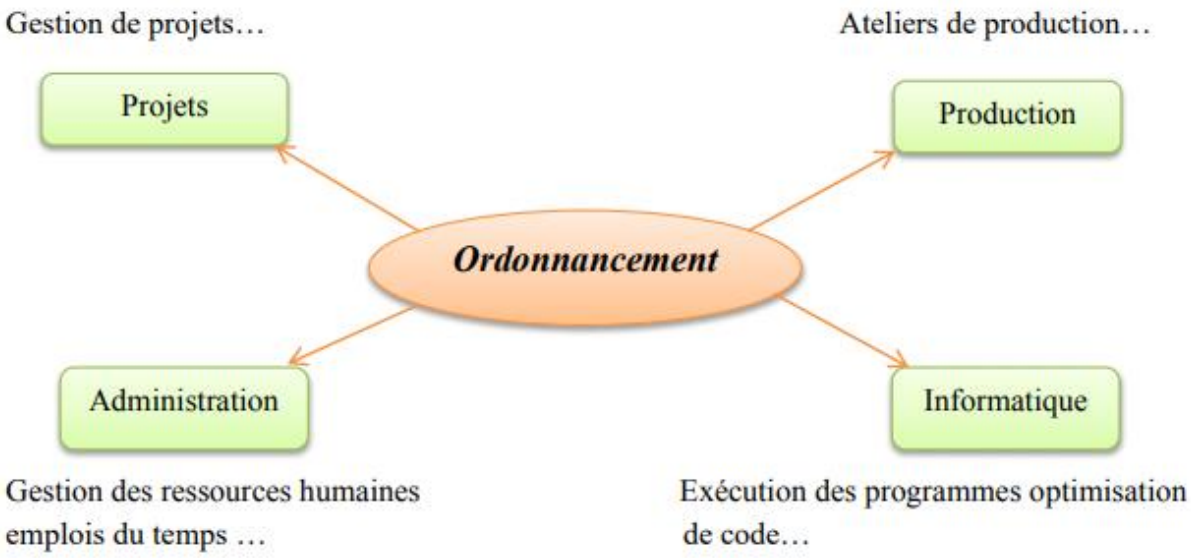


Figure 3.2 : Domaines concernés par l'ordonnancement

3. Modèle de base

Dans le modèle de base de l'ordonnancement d'atelier, l'atelier est constitué de m machines, n travaux (jobs), disponibles à la date t= 0, doivent être réalisés, un travail i est constitué de n_i opérations, l'opération j du travail i est notée (i, j) avec $(i, 1) < (i, 2) < \ldots < (i, n_i)$ si le travail i possède une gamme ($A<B$ signifie A précède B) [18].Une opération $(i,)$ utilise la machine m_i, $j$pendant toute sa durée p_i.L'objectif du problème d'ordonnancement est fixe, les date des débuts T_{ij} des opérations (ij). Il ne faut pas oublier que les ressources sont disjonctives, donc il faut déterminer l'ordre de passage de l'ensemble des travaux sur chaque machine.

4. Notion d'objectifs et de critères

Les objectifs des entreprises ont été diversifiés et l'opération d'ordonnancement été de plus en plus multicritère. Après cette opération Les critères doivent satisfaire un ordonnancement qui est varié.

Les fonctions économiques ou critères d'optimalité les plus utilisées font intervenir la

durée totale de l'ordonnancement, le délai d'exécution, les retards de l'ordonnancement et le coût des stocks d'encours. La durée totale de l'ordonnancement notée $Cmax$ est égale à la date d'achèvement de la tâche la plus tardive : $Cmax$= max c_J. C'est la longueur de l'ordonnancement (schedule length ou makespan) [18].

Le Flow time moyen $\bar{F}$ = 1 n $\sum F_j n_j$=1 ou le flow time moyen pondéré

$$F_{\overline{W}} = \sum \text{nj} = 1\ WjFj / \sum nj = 1\ Wj \text{ [5].}$$

Le critère, flow time, $\sum n_i$=1 $WiCi$ permet d'estimer le coût des stocks d'encours. En effet la tâche " i " est présente dans l'atelier entre les instants ri et Ci, et donc les stocks dont elle a besoin doivent être disponibles entre ces deux dates ; d'où le coût$\sum Wini$=1 $(Ci - ri)$ est égale à un constant pré à $\sum ni$=1 WiC_{i}.

Dans beaucoup de problèmes, il faut respecter les délais, donc les dates au plus tard d_i; on peut chercher à minimiser le plus grand retard T_{max}= max T_i, ou bien la somme des retards$\sum n_i$=1 T_i.ou encore la somme pondérée des tâches en retard $\sum n_i$=1 $W_i T_i$. Le décalage maximum L_{max}= max $\{L_J\}$ [18].

D'autres critères peuvent être utilisés :

- Le retard moyen T_{moy} = 1 /n $\sum n_j$=1 T_j. Le retard moyen pondéréT_w = $\sum n_j$=1 $W_j T_j$/ $\sum n_j$=1 W_j.
- Le nombre de tâches en retard U = $\sum Uj$. n_j=1 où U_j= 1 si C_j> d_j et 0 sinon. - Le nombre de tâches en retard pondéré U_w = $\sum W_j n_j$=1 U_j [5].

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

5. Les problèmes d'ordonnancement d'atelier

5.1 Les types de problèmes d'atelier

Une classification des problèmes d'ordonnancement dans un atelier peut s'opérer 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

On spécifie deux catégories. La première regroupe les problèmes pour lesquels chaque

tâche nécessite une seule machine et la deuxième, chaque tâche demande plusieurs machines pour son exécution.

5.1.1 Problèmes à une machine

C'est le cas le plus simple dans problèmes d'ordonnancement, l'ensemble des tâches à réaliser est fait par une seule machine(ou ressource unique). Si on ne dispose qu'une seule ressource pour réaliser un ensemble de travaux alors les tâches sont composées d'une seule opération qui nécessite la même machine.Dans ce type de problème la résolution consiste à trouver l'ordre optimal d'exécution de ces tâches vis-à-vis d'un critère donné.

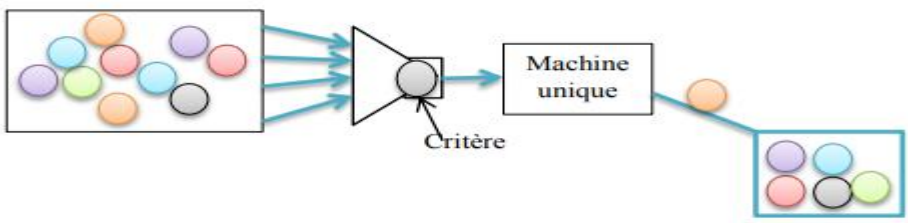


Figure 3.3 La représentation d'ordonnancement à une machine [20]

5.1.2 Problèmes à machines parallèles

Les problèmes d'ordonnancement à machines parallèles se caractérisent par l'existence de plus d'une ressource. Elles représentent un cas particulier de problèmes multi-machines où les machines sont disposées en parallèle. Pour résoudre le sous problème, l'ordonnancement s'effectue en deux phases :

La première phase décide sur quelle machine effectuer chaque opération et la deuxième phase déterminer la séquence d'opération sur chaque machine.

Le problème se divise généralement à 3 classes selon les machines qui sont :

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

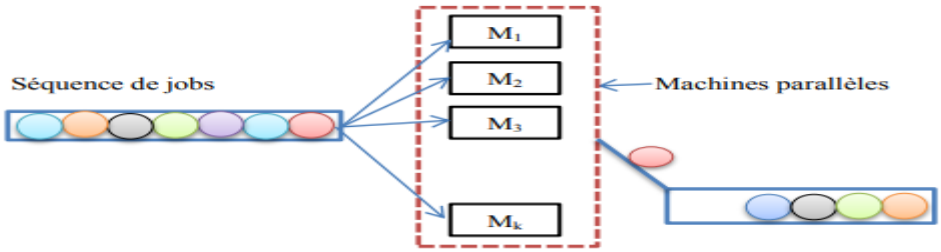


Figure 3.4 La représentation d'ordonnancement à machine parallèles [20]

5.2 Les problèmes d'atelier multi-machines

Pour ce type de problème les ressources nécessaires pour réaliser l'ensemble des travaux sont multiples. L'organisation de ces ressources et le type de passage des produits entre eux génèrent trois sous types de problèmes :

5.2.1 Problèmes flow shop

Appelés également ateliers à cheminement unique, ce sont des ateliers où une ligne de fabrication est constituée de plusieurs machines en série.

Un travail est constitué des opérations qui visitent les es machines et avec des manières linéaire suivant une chaine. Tout travail visite chaque machine de l'atelier et l'ordre de passage d'un travail sur les déférentes machines et le même pour tous les travaux cette gamme unique et une donnée du problème.

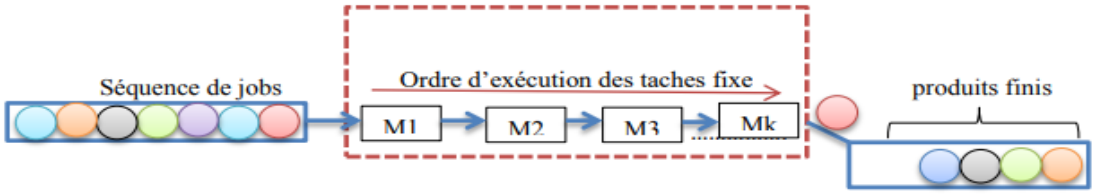


Figure 3.5 La représentation à cheminement unique. [20]

5.2.2 Problèmes flow shop avec permutation

Un cas particulier important, c'est celui du flow shop simplifié où la séquence des travaux visitant à une machine est la même pour toutes les machines, il s'agit du (flow shop permutation).Dans le cas général, le problème est NP-difficile au sens fort mais dans le cas Fm//prmu /Cmax on peut les résoudre par les méthodes arborescentes.

Figure 3.6 La représentation un flow-shop de permutation à 3 jobs et 4 machines. [3]

5.2.3 Les problèmes de type Job Shop

Appelés également ateliers à cheminement multiples qui consiste à réaliser un ensemble de n Job (travail) sur un ensemble de m machines en cherchant d'atteindre certain objectifs. Chaque Job j est composé de nj tâches devant être exécutées sur les différentes machines selon un ordre préalablement défini.

Les travaux ne s'exécutent pas sur toutes les machines et de plus n'ont pas le même ordre d'exécution. En effet chaque travail emprunte le chemin qui lui est propre et la majorité dans les cas de Job Shop NP difficile.

	1	2	3	4	1	2	3	4	1	2	3	4
J_1	M_1	M_2	M_3		10	8	4		O_{11}	O_{21}	O_{31}	
J_2	M_2	M_1	M_4	M_3	8	3	5	6	O_{22}	O_{12}	O_{42}	O_{32}
J_3	M_1	M_2	M_4		4	7	3		O_{13}	O_{23}	O_{43}	
	Machines				Durée opératoire (p_{ij})				Opérations			

Tableau 3.1 : La représentation de la gamme opératoire des jobs. [20]

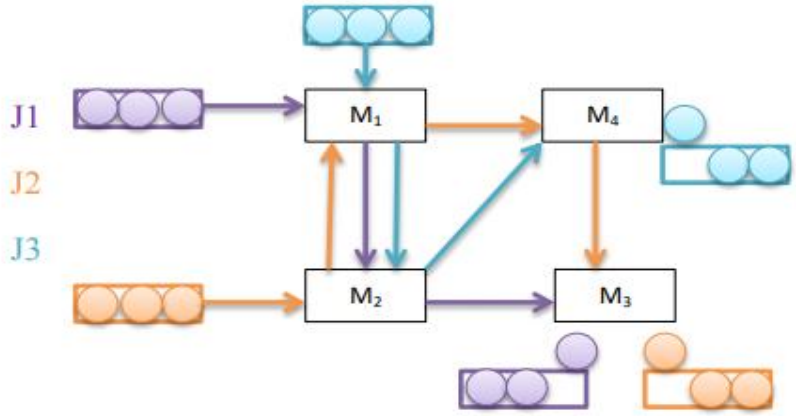


Figure 3.7 La représentation d'ateliers à cheminement multiple (job-shop) [20].

5.2.4 Problèmes de type Open Shop

Appelés également ateliers à cheminement libre où l'ordre d'exécution de tâches d'un travail est totalement libre, c'est-à-dire aucune contrainte est remarquée sur la précédence entre tâches. Les produits passent alors dans n'importe quelle direction (les gammes sont libres).

5.3 La notion de flexibilité

La flexibilité est un mode de gestion de la main d'œuvre qui permet aux entreprises d'ajuster rapidement la production et l'emploi (l'offre) aux fluctuations rapides des commandes des clients (demande).

Afin d'être toujours plus réactives et productives, les entreprises ont cherché à augmenter la flexibilité de leurs systèmes de production. Pour atteindre ce but, il est possible de multiplier le nombre des machines qui peuvent réaliser une même opération. Ces machines, considérées comme identiques dans le cadre de ce mémoire, sont regroupées en étage ou cellule. Et pour cela on peut distinguer autres types d'atelier :

5.3.1 Flow shop hybride

Le Flow Shop hybride est une généralisation du Flow Shop classique au cas où nombreuse machines sont disponibles sur un ou plusieurs étages pour exécuter les différentes tâches du Flow Shop. Ces problèmes présentent alors une difficulté supplémentaire par

rapport aux problèmes sans flexibilité des ressources. En effet, la machine qui sera utilisée pour exécuter une opération n'est pas connue d'avance, mais doit être sélectionnée parmi un ensemble donné pour construire une solution au problème.

Chaque étage k (un ensemble des machine parallèle) contient $E_k$contient M_k machines parallèles identiques et tous les jobs exigent le même ordre des opérations qui doit être exécuté selon le même processus de fabrication l'opération O_{ik} à besoin d'un temps d'exécution P_{ik} sur l'étage E.

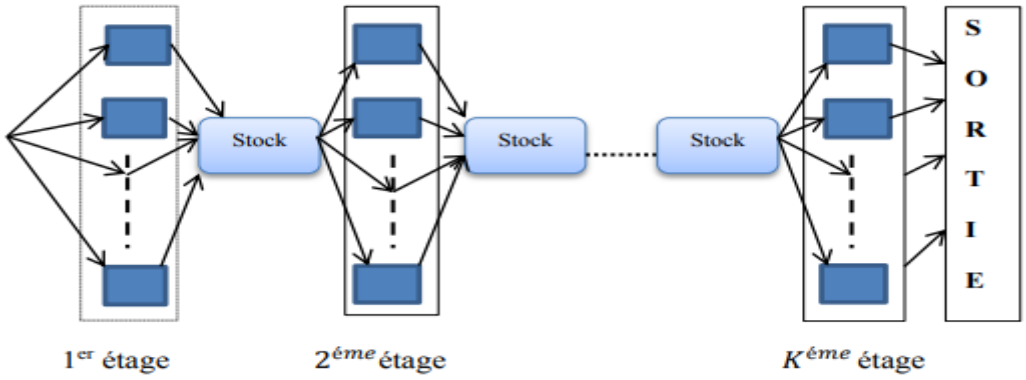


Figure 3.8 La représentation d'un flow show hybride à « k » étages [17].

5.3.2 Problème de Job Shop hybride

Le job shop flexible est une extension du modèle job shop classique et le problème de machine parallèle. 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. [5]

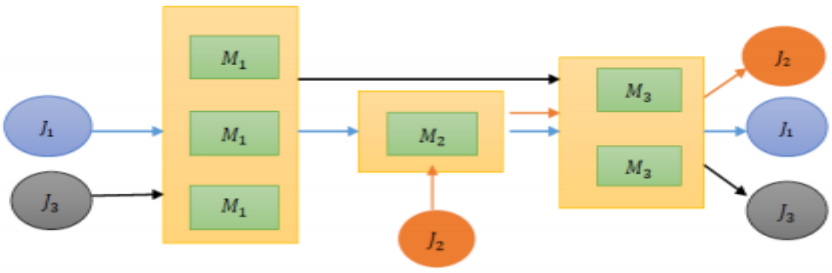


Figure 3.9 La représentation d'un job shop hybride. [15]

Le problème Job shop dans un environnement dynamique

1. Introduction

L'objectif de ce chapitre est de définir et de situer le problème d'ordonnancement job shop visé par ce travail. Nous présentons ensuite le système à étudier et la formulation du problème, Les modèles du problème, à savoir le modèle statique et le modèle dynamique (l'ordonnancement job shop en temps réel).

2. Système à étudier et thème proposé

2.1 Définition formelle

Dans ce thème, nous travaillons exclusivement sur l'ordonnancement d'atelier de type job shop (atelier à cheminements multiples).

Les problèmes d'ordonnancement en job shop sont des problèmes classiques en littérature d'ordonnancement.

Le job shop est classé dans les systèmes de production au flux discret où chaque job a sa propre gamme. Une gamme consiste à visiter un ensemble de machines M dans un ordre donné. Le Job-Shop ou JSSP pour (Job-Shop Scheduling Problem) est plus général que le Flow Shop car chaque job possède une gamme opératoire différente.

Le problème de type Job Shop est classé sous le terme « job-shop J×M » : Ils sont composés d'un ensemble de J jobs (produits), chacun composé d'un ensemble de nj opérations qui peuvent être exécutées sur m machines. On note que le système n'autorise pas les stocks intermédiaires. Alors le résultat immédiat est qu'une machine n'est pas disponible dès la terminaison d'une tâche. Celle-ci doit attendre que la machine suivante dans la gamme opératoire soit libérée pour lui transférer la pièce.

De cette contrainte, on a emporté la propriété « succession sans interruption » qui impose toujours le non séparation entre tâches. Pour chaque machine, on dispose d'un et un seul exemplaire, on dit que l'organisation est un Job Shop simple. Au contraire, pour un au moins des machines (postes de travail), on dispose de plus d'un exemplaire, on l'appelle Job Shop Hybride.

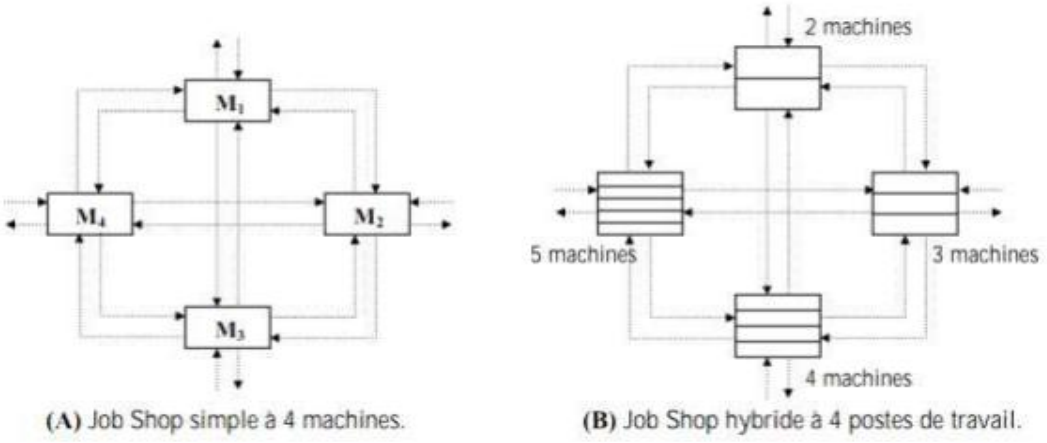


Figure 4.1 : Une représentation d'un problème job shop simple et hybride. [17]

Les hypothèses suivantes sont communément retenues pour le problème de job-shop :

- Tous les jobs sont disponibles dès le début de l'ordonnancement.
- Les machines sont disponibles sur tout l'horizon d'ordonnancement (pas de panne, pas d'état initial non vide).
- Les temps de traitement pi sont déterministes et connus à l'avance.
- Les temps de transport sont négligeables.
- Chaque machine ne peut réaliser à un instant donné qu'une seule opération.
- Un job peut être traité au plus par une machine à un instant donné.
- Les produits peuvent attendre dans les zones de stockage de capacité illimitée.
- Une machine ne peut exécuter qu'une seule opération à la fois et à l'instant initial toutes les machines sont disponibles.

2.2 Les machines

Les machines représentent l'élément nécessaire du système, ce sont des outils contenant des pièces réalisant des tâches spécifiques. Chaque machine est une ressource disjonctive, c'est -à-dire qu'elle ne peut exécuter qu'une seule tâche à la foi. La phase d'exécution considère la disponibilité de ces machines. Il y a deux états caractérisant le fonctionnement des machines :

- l'état occupé : la ressource est entrainé de réaliser une certaine tâche (état de marche).
- l'état non occupé ou libre : la ressource n'exécute aucune tâche (état de repos).

2.3 Les tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début ou de fin, dont la réalisation est caractérisée par une durée défini. Les taches sont regroupées en n entités appelées travaux ou lots. Un ensemble de propriétés, sur les tâches sont à souligner :

- La durée de chacune de ces tâches est déterminée par l'union des temps suivant :
  - le temps de transport, c'est le temps de déplacement de la/les pièces vers la machine concernée.
  - le temps d'exécution, qui représente le temps dans lequel une pièce doit rester dans la machine.
  - le temps de préparation des ressources.

On note que s'il s'agit d'un lot, ce temps est calculé par la somme des temps de toutes les pièces de ce lot.

- La notion de préemption des tâche est non permit, c'est-à-dire que les tâches n'acceptent aucune interruption une fois sont commencées.
- La propriété succession sans interruption impose de ne pas avoir une séparation entre les tâches.

Dans certains cas, il est possible d'étendre la durée d'exécution d'une tâche, la ressource concernée n'est pas donc disponible pour exécuter une autre tâche pendant cette durée. Une opération ne peut être exécutée que sur une seule machine et sans interruption [4].

2.4 Représentations d'un problème de type job-shop

Dans le cadre d'un problème d'ordonnancement, le diagramme de Gantt prend en ordonnée les différentes ressources utilisées et donne une représentation d'un problème du job-shop.

Soit un problème de job-shop consistant à usiner sur 4 machines 3 pièces (3 jobs). Le Tableau suivant (Tableau 4.1) représente les gammes opératoires des jobs et les dures des opérations [16] :

Job	Opération 1	Opération 2	Opération 3
Job 1	(M 3, 1)	(M 1, 3)	(M 2, 7)
Job 2	(M 3, 5)	(M 4, 4)	(M 1, 1)
Job 3	(M 2, 5)	(M 1, 5)	(M 3, 3)

Tableau 4.1 : la représentation les gammes opératoires des jobs. [16]

Une solution de ce problème consiste à définir un ordre des opérations sur les machines. Le tableau suivant définit un ordre de passage des différents jobs sur les différentes machines.

M1	Op. 2 du Job 3	Op. 2 du Job 1	Op 3 du Job 2
M2	Op. 1 du Job 3	Op. 3 du Job 1	
M3	Op. 3 du Job 3	Op. 1 du Job 1	Op. 1 du Job 2
M4			Op2 du Job 2

Tableau 4.2 : la Une représentation l'ordre de passage des différents jobs. [16]

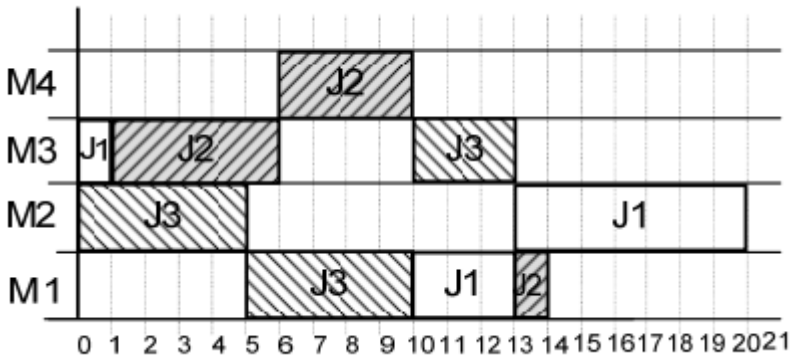


Figure 4.2 : Une représentation sous forme de diagramme de Gantt d'un problème du job-shop [16].

3 Formulation du problème

3.1 Formulation linéaire

Plusieurs formulations linéaires existent pour le job-shop et certaines d'entre elle se basent

sur la formulation de Manne. Dans Pham (2008), une évaluation des formulations linéaires du job-shop est proposée. Pour un problème de Job-Shop de njobs et m machines, on considère les notations suivantes [16] :

$\tilde{J}$ L'ensemble de tous les jobs ; J = {2 ; 1 ;...; n} ;

I L'ensemble de toutes les opérations ;

M L'ensemble de machines M = {2 ; 1 ;...; m} ;

A_j L'ensemble de tous les couples d'opérations consécutives pour le job j$\in\tilde{J}$;

B L'ensemble de tous les couples d'opérations (i, j) $\in I$, i $\neq$ j exécutées sur la même machine ;

I_k L'ensemble des opérations exécutées sur la machine k $\in$ M ;

P_i Le temps opératoire de l'opération i $\in$ I ;

H Un nombre entier positif suffisamment grand ;

x_i La date de début de l'opération i $\in$ I ;

x_τ La date de début de l'opération * ;

3.2 Formulation mathématique

A fin décrire le problème de façon univoque, nous le décrivons à l'aide du formalisme mathématique .la formulation proposée par (Manne, 1960) peut aisément être adaptée aux notations ci-dessus pour le problème de job shop [16] :

$$x_j - x_i \geq P_i \forall (i,j) \in A_k, \forall k \in \tilde{J} \quad (1)$$

$$x_j + H(1 - y_{ij}) - x_i \geq P_i \forall (i,j) \in B \quad (2)$$

$$x_i + H y_{ij} - x_j \geq P_j \forall (i,j) \in B \quad (3)$$

$$x_\tau - x_i - P_i \geq 0 \qquad \forall i \in I \quad (4)$$

$$y_{ij} \in \{0,1\} \forall (i,j) \in B \quad (5)$$

$$x_i \geq 0 \qquad \forall i \in I \quad (6)$$

$$x_\tau \geq 0 \quad (7)$$

Les contraintes (1) assurent qu'aucune opération ne peut commencer avant la fin d'exécution de l'opération qui la précède.

Les contraintes (2) et (3) sont des contraintes de disjonction machine et assurent que deux opérations s'exécutant sur la même machine doivent être ordonnées grâce à l'utilisation de la variable binaire $y_{ij} \in \{1,0\}$ de la contrainte (5).

Les contraintes (4) fixent la date de début de l'opération *, ceci est assuré par le fait que $x\tau$est supérieur à toutes les dates de début plus le temps de traitement xi+ pide chaque opération pari ∈ I.

Les contraintes (6) et (7) imposent que toutes les dates de débuts des opérations sont positives ou nulles. [16]

4 Les modèles du problème

4.1 Le modèle statique

4.1.1 Ordre de problème job shop

Dans les problèmes à cheminement multiples (job shop), L'ordre d'exécution des opérations au sein des travaux, est linéaire, fixée à l'avance et identique pour tous les travaux, ce qui permet de numéroter les machines dans l'ordre d'exécution des opérations à traiter. La production continue est caractérisée par la fluidité de son processus et l'élimination du stockage.

4.1.2 Algorithme de Johnson

L'algorithme de Johnson proposé par S.M Johnson qui calcule l'ordonnancement minimisant le temps total d'exécution des taches. Cet algorithme, et ses variantes, est un moyen rapide d'optimiser l'ordonnancement de processus simples. Son objectif est de minimiser la durée de réalisation d'une file d'attente de n pièces devant toutes passer selon le même ordre sur deux machines. La complexité de l'algorithme de Johnson est de l'ordre : $O(n \log(n))$

4.1.3 Ordonnancement d'atelier à deux machines

Il s'agit ici d'un autre cas particulier de l'ordonnancement de projet à contraintes disjonctives. Il y a seulement deux ressources en un exemplaire, que l'on appelle machine 1 et machine 2.Les tâches sont couplées par deux et correspondent à des travaux ou jobs. A tout travail correspond une tâche ou opération sur la machine 1, suivie d'une tâche ou opération sur la machine 2.Le

critère à minimiser est toujours la durée totale. Un théorème, démontré en 1954 par Johnson, permet de construire un algorithme rapide (polynomial) qui résout ce problème de manière optimale. Le calcul des dates au plus tôt permet d'obtenir l'ordonnancement précis correspondant à l'algorithme de Johnson. [d]

Premier algorithme construit à partir de la règle de Johnson

A_i : processing time ou durée du travail i sur la première machine.

B_i : processing time ou durée du travail i sur la seconde machine.

Classer les travaux en deux groupes.

Dans le premier groupe G_1, mettre tous les travaux tels que $A_i<B_i$

Dans le second groupe G_2, mettre tous les travaux tels que $B_i \leq A_i$

Classer G_1 par A_i croissants

Classer G_2 Bi décroissants

La solution optimale est constituée de la séquence G_1 suivi de G_2.

Cette séquence est optimale.

Figure 4.3 : Présentation de premier algorithme de deux machines, la règle de Johnson. [d]

Soit un problème de job-shop consistant à usiner sur 5taches soient à exécuter sur 2 machines (2 pièces). Le Tableau suivant a représenté les durées des jobs :

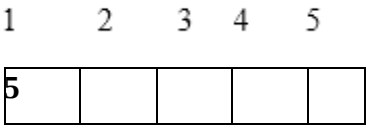
Tâches (i)	1	2	3	4	5
$t_{i,A}$	50	150	80	200	30
$t_{i,B}$	60	50	150	70	200

Figure 4.4 : Présentation d'un exemple de premier algorithme de deux machines

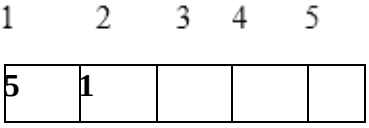
On applique le premier algorithme :

- Dans le premier groupe G_1, $A_i<B_i$:

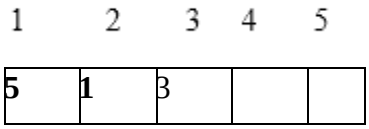
- la tache **5** (30<200) donc mise en première position.



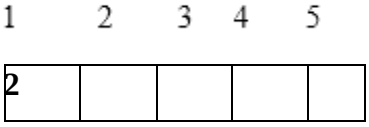
- la tache **1** (50<60) donc mise en deuxième position.



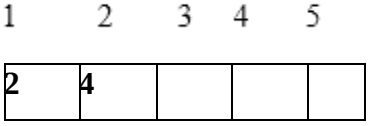
- la tache **3** (80<150) donc mise en troisième position.



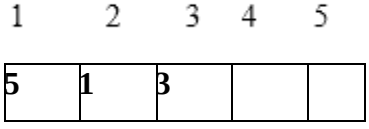
- Dans le deuxième groupe G_2, $B_i \leq A_i$:
  - la tache **2** (50<150) donc mise en première position



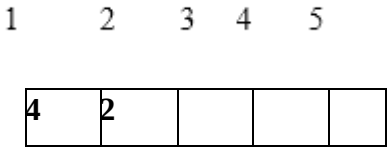
  - la tache **4** (70<150) donc mise en deuxième position



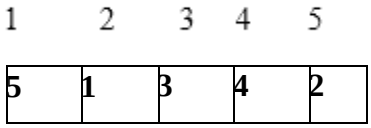
- Classer G_1 par A_i croissants :



- Classer G_2 B_i décroissants :



- La solution optimale est constituée de la séquence G_1 suivi de G_2.



Deuxième algorithme construit à partir de la règle de Johnson

On considère le cas où toutes les tâches sont `à exécuter sur le premier centre puis sur le second. Soient t_{iA} et t_{iB} les temps d'exécution de la tâche i sur les machines (ou centres de production) A et B respectivement. On va utiliser comme critère d'ordonnancement la minimisation du temps total d'exécution des tâches sur les deux machines [11].

Rechercher la tâche i de temps d'exécution T_{ij} minimum.

Si m = A, placer cette tâche à la première place disponible.

Si m = B, placer cette tâche à la dernière place disponible.

Supprimer la tâche i des tâches encore à programmer, retour en 1.

Figure 4.5 : Représentation de deuxième algorithme de la règle de Johnson [11].

Supposons le même exemple, 5 tâches soient à exécuter sur les machines A puis B on Applique l'algorithme [11] :

Tâches (i)	1	2	3	4	5
$t_{i,A}$	50	150	80	200	30
$t_{i,B}$	60	50	150	70	200

Tableau 4.3 : Présentation d'un exemple de premier algorithme de deux machines.

• la tâche 5 (t_{5A} = 30) est mise en première position.

1	2	3	4	5
5				

• Puis la tâche 1 (t_{1A} = 50) est mise en deuxième position.

1	2	3	4	5
5	**1**			

•Puis la tâche 2 (t_{2B} = 50) est mise en dernière position.

1	2	3	4	5
5	**1**			**2**

• Puis la tâche 4 (t_{4B} = 70) est mise en avant-dernière position.

1	2	3	4	5
5	1		4	2

• Puis la tâche 3 est mise `à la dernière place disponible.

1	2	3	4	5
5	**1**	**3**	**4**	**2**

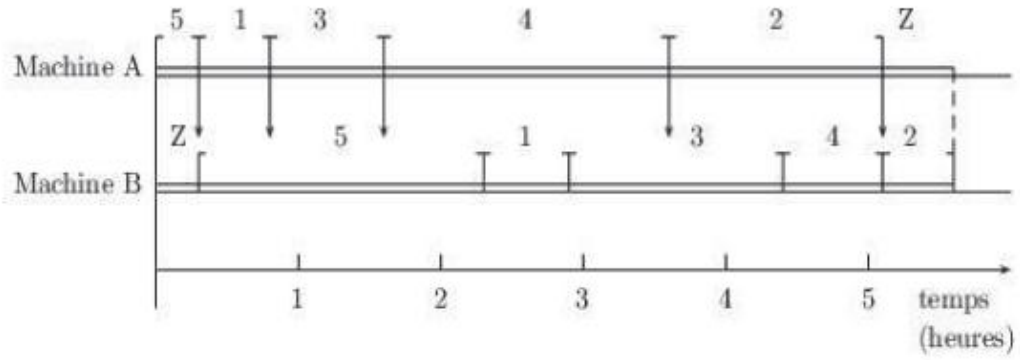


Figure 4.6 : Représentation de diagramme de Gantt de 2 machines [11].

L'ordonnancement obtenu est optimal. Le passage d'une tâche d'une machine à l'autre est visualisé `à l'aide d'une flèche verticale. On notera qu'une machine au repos est indiquée par un Z.

Si l'on veille à aligner verticalement l'origine du temps pour chaque machine, une ligne verticale indique donc à tout moment à quelle tâche est occupée chacune des machines. Un tableau mural peut ˆêtre ainsi d'un grand recours pour les agents de maitrise responsables de l'affectation des moyens humains et matériels. [11]

4.1.4 Ordonnancement d'atelier à >2 machines

L'algorithme de Johnson ne s'applique qu'en présence de deux machines. Cependant, le cas de m machines peut se ramener au cas de deux machines. C'est-`a-dire si l'on se trouve dans le cas où [11] :

Minimum Temps de traitement de M1 ≥ maximum Temps de traitement de M2, M3,… M $(n-1)$

Soit dans le cas où :

Minimum Temps de traitement de M(n) ≥ maximum Temps de traitement de M_2, M_3,… M $(n-1)$

Ces conditions évitent de fausser l'algorithme de Johnson, si la condition vérifie elle le

ramènera les machines au cas de deux machines comme suit :

Machine 1 :$\sum(M1, M2 \ldots \ldots \mathrm{M}\ (n-1))$.

Machine 2 :$\sum(M2, M3 \ldots \ldots \mathrm{M}\ (n))$.

Dans le cadre de l'exemple ci-dessous, on a le cas de trois machines peut se réduire au cas de deux machines [11].

Tâches	1	2	3	4	5	6	7
Assemblage	20	12	19	16	14	12	17
Inspection	4	1	9	12	5	7	8
Expédition	7	11	4	18	18	3	6

Tableau 4.4 : La représentation d'un exemple de 3 machines. [11]

On constate que les conditions d'application énoncées précédemment sont vérifiées. Il est à remarquer que les deux conditions ne doivent pas ˆêtre vérifiées simultanément. Dans l'exemple la seconde condition n'est pas vérifiée.

Lorsqu'on se trouve dans un des deux cas, on reformule le problème en un problème à deux machines. Dans le cadre de l'exempleci-dessous, la première regroupe les machines M1 et M2.

Avec $\mathrm{T_i}(1;2) = \mathrm{T_i}(M1) + \mathrm{T_i}(M2)$ et la seconde regroupe les machines M_2 et M_3 avec $\mathrm{T_i}(2;3) = \mathrm{T_i}(M2) + \mathrm{T_i}(n)$ ce qui donne les nouveaux temps opératoires [11].

Tâches	1	2	3	4	5	6	7
Assemblage + Inspection	24	13	28	28	19	19	25
Inspection + Expédition	11	12	13	30	23	10	14

Tableau 4.5 : Présentation de trois machines réduire au cas de deux machines [11].

On applique alors l'algorithme de Johnson à ce problème à deux machines pour déterminer l'ordonnancement optimal.

Place	1	2	3	4	5	6	7
Tâche	5	4	7	3	2	1	6

Tableau 4.6 : Présentation l'ordonnancement optimal.

On peut alors tracer le diagramme de Gantt correspondant au problème original, C'est à-dire celui avec trois machines.

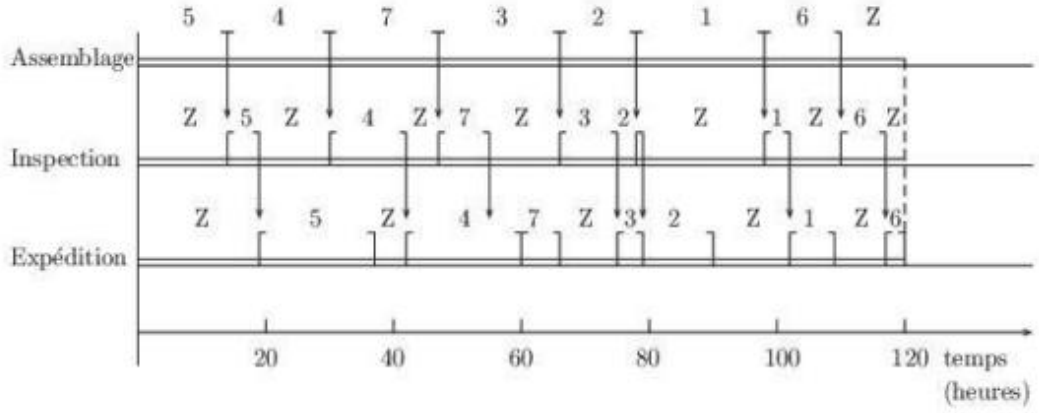


Figure 4.7 : Représentation de diagramme de Gantt de 3 machines [11].

4.1.5 Calcul du Makespan (Cmax)

Notre objectif est de trouver le Cmax minimum. Pour calculer le Makespan (Cmax) nous avons utilisé la méthode de diagramme de Gantt qui représente l'ordonnancement des différents job à exécuter dans les machines Mn dont les jobs ont des routages spécifiques, des temps d'exécution (Processing time) et temps de transport vers chaque machine.

En respectant l'ordre d'exécution des produits le Cmax représente le temps de la sortie du dernier produit exécuté dans le système.

Nous avons calculé le Cmax sur le diagramme de l'exemple précédent on obtient le résultat dans le tableau suivantes :

	5		4		7		3		2		1		6	
Assemblage	0	14	14	30	30	47	47	66	66	78	78	98	98	110
Inspection	14	19	30	42	47	55	66	69	87	79	98	102	110	117
Expédition	19	37	42	60	60	66	69	72	79	90	102	109	117	120

Tableau 4.7 : Présentation de calcule le Cmax d'un exemple précédent.

- Le C_{max} de la machine Assemblage : 110
- Le C_{max} de la machine Inspection : 117
- Le C_{max} de la machine Expédition : 120

4.2 Modèle dynamique

4.2.1 Ordonnancement d'atelier en temps réel

Le problème d'ordonnancement en temps réel consiste à adapter en permanence les modalités d'exécutions d'ensemble des tâches par un ensemble des ressources à la situation réel du système considéré. On rencontre souvent ce type de problème dans les ateliers de fabrication travaillant à la commande où le délai de livraison représente une des difficultés majeures. L'ordonnancement en temps réel offre à ces ateliers avec la prise en compte de ses caractéristiques réelles, une capacité de réagir aux déférents aléas (internes ou externes) auxquels ces ateliers sont soumis. [9]

Dans le domaine informatique la problématique de l'ordonnancement en temps réel revient à définir dans quel ordre exécuter les tâches sur chaque processeur d'une architecture donnée.

Cette problématique est en générale limitée à une ressource ou plusieurs ressources en parallèle (processeurs), de plus l'interruption de tâches (préemption) est souvent autorisée. [9]

4.2.2 Approches pour l'ordonnancement temps réel d'atelier

Pour la résolution des problèmes d'ordonnancement en temps réel deux types d'approches sont utilisées. Des approches basées sur un contexte statique et d'autres basées sur un contexte dynamique. [9]

Dans le contexte statique un ensemble des jobs est supposé connu à l'avance sur un horizon

fini. La séquence de tâches est fixée par une approche prévisionnelle traditionnelle.

L'approche en temps réel consiste alors à contrôler le respect de la séquence proposée et les dates de débuts prévues, tout en prenant en compte les perturbations apparues dans le système. Différents travaux ont été réalisés dans ce contexte, qui a utilisés ce type d'approches en tenant en compte les nouveaux jobs inclus dans le plan de fabrication.

Dans le contexte dynamique les approches proposées considèrent que l'ensemble des jobs à ordonnancer ne sont pas connus d'avance, la solution ne s'effectue que dès qu'un job survient, l'ordonnancement donc consiste à affecter ses opérations aux ressources disponibles. [9]

4.2.3 Méthode d'insertion de tâches

Une approche basée sur le contexte statique et dynamique, c'est la méthode d'insertion des tâches dans un ordonnancement prévisionnel. Cette méthode consiste à sélectionner et choisir le bon endroit (emplacement) où ces nouvelles tâches doivent être inclues. Dans la littérature on trouve que l'utilisation de la méthode d'insertion de tâches n'est pas limitée au problème d'ordonnancement d'une seule ressource, mais aussi pour plusieurs ressources. [9]

L'insertion d'une tâche urgente ne doit pas dépasser son délai de livraison où doit exister une solution avec possibilité de modifier le plan prévisionnel si c'est nécessaire. Pour trouver la solution basée sur la recherche d'une position d'insertion admissible on doit effectuer une procédure de décalage qui sert à modifier l'ordre de passage existant, de tel sort que la position soit admissible.

Le but visé est de définir la bonne solution qui conduit à une modification minimale du plan initial. Deux critères sont utilisés pour sélectionner cette solution, la durée totale d'exécution Cmax (Makespan).

4.2.3.1 Définition d'une position admissible

La détermination d'une position admissible, elle consiste à positionner les tâches de la commande urgente sur le plan prévisionnel par la détermination des dates début et de fin de chaque tâche,

L'obtention d'une position admissible implique la maintenir de solution. Dans le cas contraire celle-ci conduit au passage à la procédure de décalage.

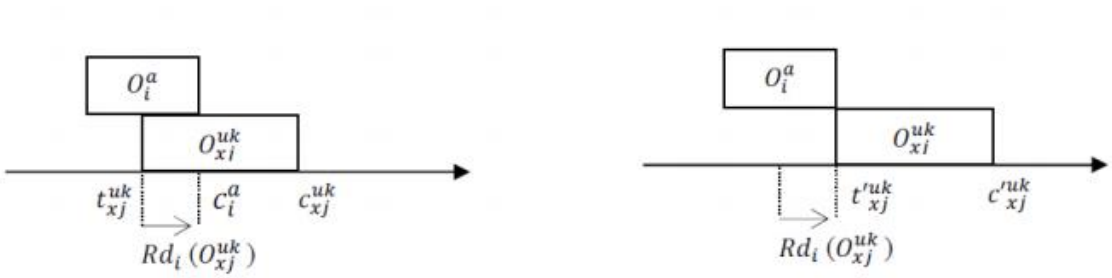
4.2.3.2 Procédure de décalage

La modification du plan prévisionnel se réalise par l'opération de décalage d'une ou plusieurs tâches afin de générer des espaces libres supplémentaires pour qu'une position non admissible devienne admissible. D'une autre manière c'est une quantité du temps additionnée aux dates début et de fin de ces tâches (Figure 4.8).

4.2.3.3 Création d'une position admissible et détermination du nouvel ordre

On définit par $Rd_i(O_{xj}^{uk})$ le temps de décalage (retard) que doit subir la tâche (O_{xj}^{uk}) pour permettre l'insertion de la tâche (O_i^a). Ce retard est déterminé par la différence entre la date début de la tâche concernée par le décalage et la date fin de la tâche à insérer (l'équation).

$$Rd_i(O_{xj}^{uk}) = C_i^a - t_{xj}^{uk}$$



a. Position avant le décalage

b. Position après le décalage

Figure 4.8 : Exemple d'une opération de décalage effectué pour insérer la tâche O_i^a .[9]

La Figure 4.8 explique l'opération de décalage effectuée sur la tâche (O_{xj}^{uk}) où t'^{uk}_{xj} et C'^{uk}_{xj} représentent les nouvelles données de la tâche(O_{xj}^{uk}), leurs calcule se fait par les relations suivantes :

$$t'^{uk}_{xj} = t_{xj}^{uk} + Rd_i(O_{xj}^{uk}) \text{ et}$$

$$C'^{uk}_{xj} = C_{xj}^{uk} + Rdi(O_{xj}^{uk})$$

Conception et Réalisation

1 . Introduction

Dans ce chapitre on va vous présenter la réalisation de cette étude, nous avons conçu une application ayant pour but d'appliquer la règle de résolution que nous avons présentée dans le précédent chapitre,

À savoir l'algorithme Johnson, adapté au problème d'ordonnancement d'atelier job shop dynamique en temps réel.

2. Les éléments matériels

Pour agrandir l'application, nous avons utilisé comme élément matériel un ordinateurs HP qui possède comme particuliers :

- Un processeur Intel (R) Core (TM) i3-3110M CPU@ 2.40Hz 2.40GHz
- Une mémoire vive de 4Go.
- Un disque dur 500 Go.
- Système d'exploitation 64 bit.

3. les éléments logiciels

Plateforme utilisé est Windows 7 professionnel pack1 .le langage de développement choisit est java.

3.1 Le langage de programmation java

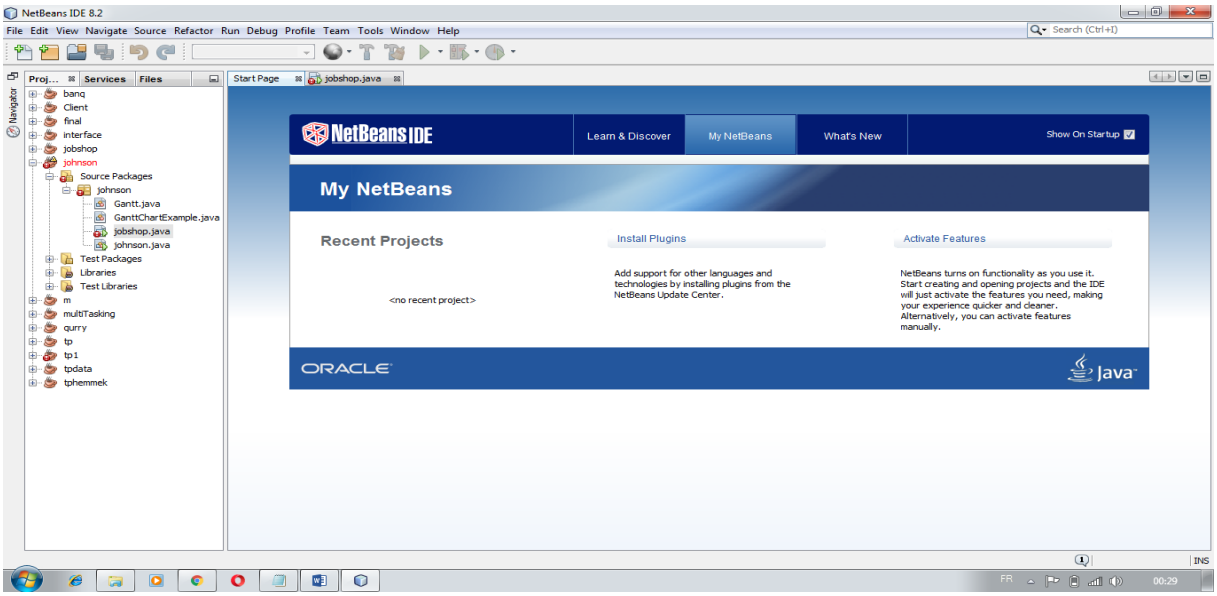
Le langage Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy, cofondateur de Sun Microsystems. Java a été officiellement présentée le 23 mai 1995 au SunWorld. La société Oracle racheta alors la société Sun en 2009, ce qui explique pourquoi ce langage appartient désormais à Oracle. La particularité et l'intérêt de Java réside dans sa portabilité entre les différents systèmes d'exploitations tels que Unix, Windows, ou MacOS. Un programme développé en langage Java, peut ainsi s'exécuter sur toutes les plateformes, grâce à ses Framework associés visant à garantir cette portabilité. [c]

- C'est l'un des langages de programmation les plus populaires au monde
- Il est facile à apprendre et simple à utiliser
- C'est open-source et gratuit

- C'est sécurisé, rapide et puissant
- Il a un énorme soutien de la communauté (des dizaines de millions de développeurs)

3.2 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)).



La figure 5.1 : la page principale de NetBeans.

4. Application de l'algorithme Johnson d'un problème job shop dans un environnement dynamique

Le codage basé sur les opérations d'une tâche donnée porte le même symbole (le numéro de la tâche). L'interprétation de ce symbole se fait suivant l'ordre de son occurrence dans la liste.

Dans notre application, La procédure "Convert" est très importante dans l'algorithme. Nous l'avons utilisé à l'intérieur de cet algorithme, est détaillée :

```
Procédure Convert (nb Machine entier)
Début
        Si (nb machine >2)
         Début
            Trouver Minimum durée des taches de la machine 1
            Trouver les maximums durés des taches des autres machines sauf la dernier
                Machine n
            Si (Min Machine 1 > Max Machine i [1, n-1])
            Machine A1 = Somme (Machine [0, n-1])
            Machine A2 = Somme (Machine [1, n])

            Si (Min Machine n > Max Machine i [1, n-1])
            Machine A1 = Somme (Machine [0, n-1])
            Machine A2 = Somme (Machine [1, n])
            Sinon
            Ecrire (ne peut pas convertir ce problème a deux machines)
         Fin
Fin
```

La figure 5.2 : l'Algorithme de Procédureconvert.

1. La procédure "Convert" ramène les machines au cas 2 machines qui devient un problème à deux machines et fait un appel à la procédure findminmax :
2. Procédure findminmax :

   La procédure findminmax trouve l'ordre d'ordonnancement des machines. Cette procédure vérifie que l'algorithme respecte bien la règle de Johnson, car elle est vérifiée localement chaque fois.

```
Procédure Findminmax (nb Machine entier)

Début
        Si (nb machine =2)
         Début
            recherche la plus petite durée d d'opération sur les deux machines ;


           Si m = A, placer cette tâche à la première place disponible.

                    on placer cette tâche à la première place disponible ;

                fin si

                  Si    m = B, placer cette tâche à la dernière place disponible.
   on placer cette tâche à la

                fin si


      Supprimer la tâche i des tâches encore à programmer, retour en 1

Fin
```

La figure 5.3 : l'Algorithme de Procédure findminmax

3. La procédure getmax : c'est le critère que nous avons utilisé pour calculer le makespan (Cmax) de l'ordonnancement.
4. La procédure l'insertion : est appliquée après la procédure findminmax qui fait ajouter une nouvelle commande. Cette procédure consiste à changer l'ordre d'ordonnancement par un nouvel ordre.

Dans notre application, deux stratégies d'insertion sont utilisées :

- insertion à la fin.
- insertion aléatoire.

```
Procédure Insertion ()
Début
Lire (position NV Tache)
Lire (Les Durées NV Tache)
Ajouter à l'ordre (NV Tache) a L'indice (Position)
Ajouter Les durées de NV Tache et sa position à L'ensemble des Machines


Fin
```

La figure 5.4 : l'Algorithme de Procédure insertion

5. La procédure getmax2 : c'est le critère que nous avons utilisé pour calculer le makespan (Cmax) de l'ordonnancement après l'insertion d'une nouvelle commande.

Donc, on peut résumer ce travail par le schéma suivant :

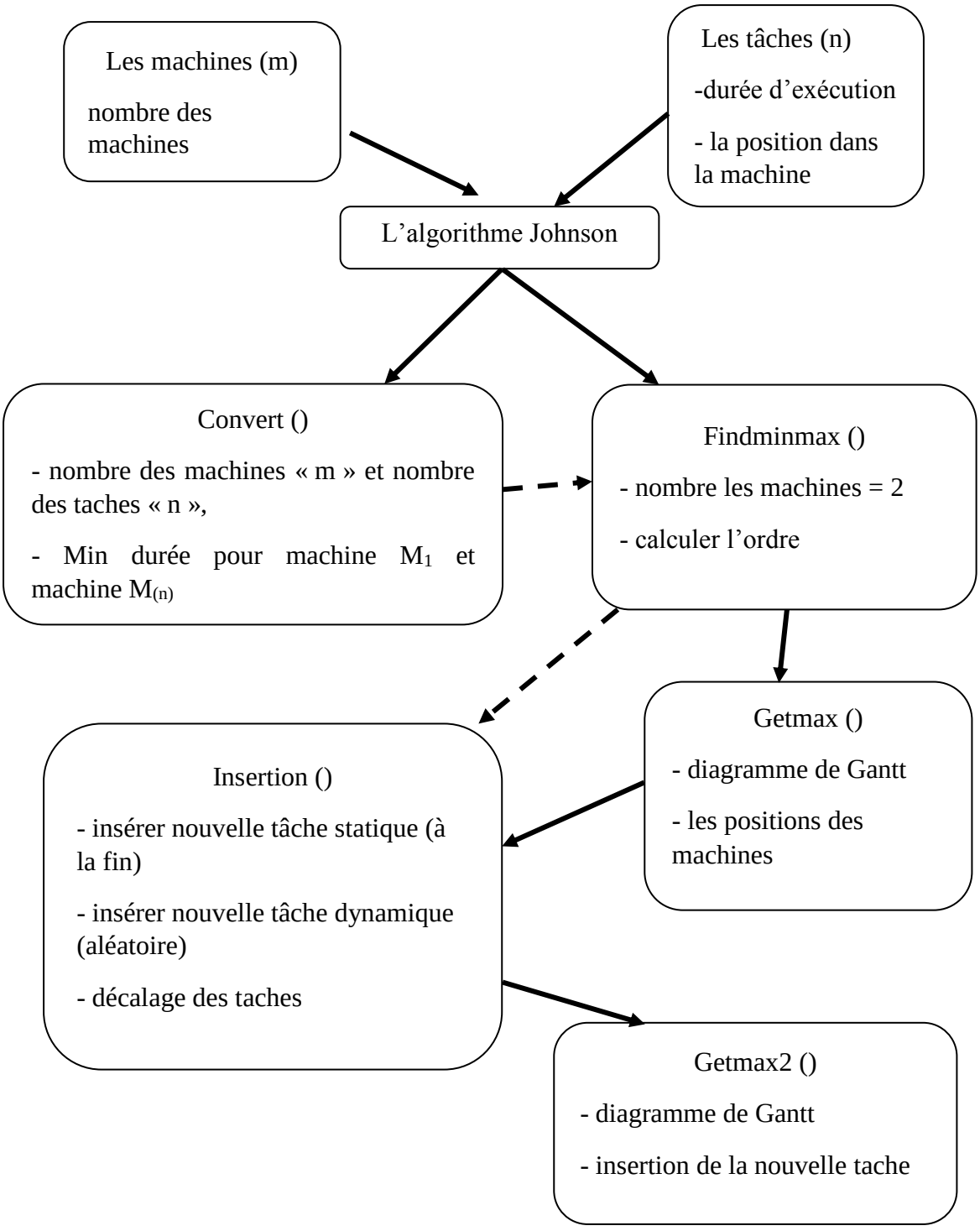


Figure 5.5 : Le schéma de réalisation du problème

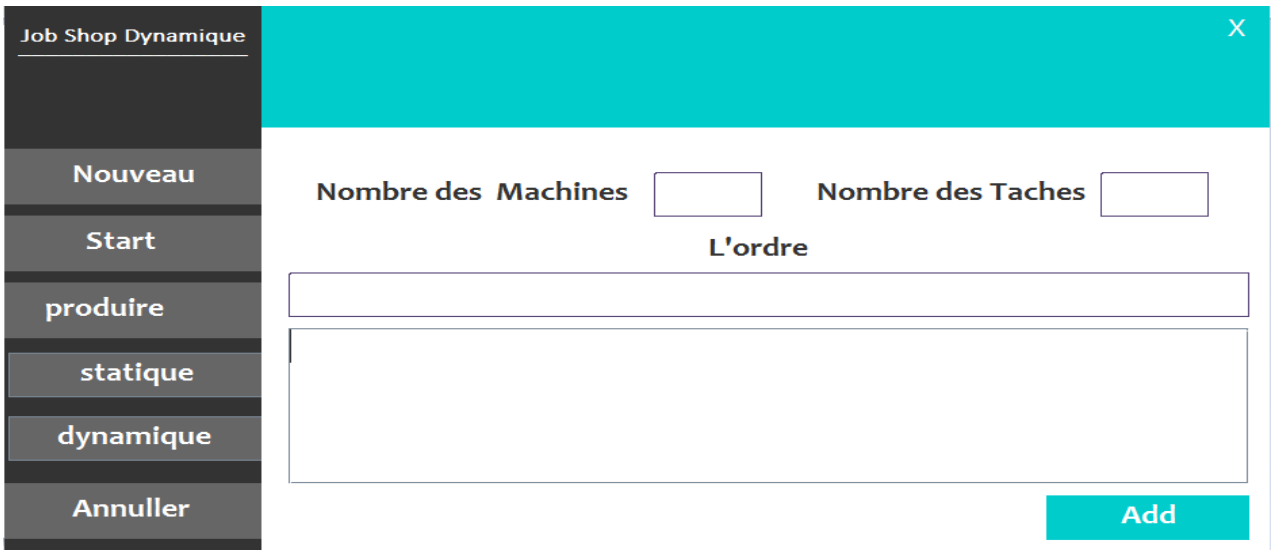
5 .Interface graphique de l'application

5.1 Modèle statique

Pour faire résoudre le problème d'ordonnancement d'atelier en temps réel de type job shop dynamique, nous sommes conduit à développer un logiciel simplifié. L'interface du logiciel conçue contient une seule fenêtre. Cette fenêtre permet à l'utilisateur de saisir les paramètres de nombre des taches, le de nombre des machines et la durée des taches de chaque machine pour créer les listes. En voici le menu principal :

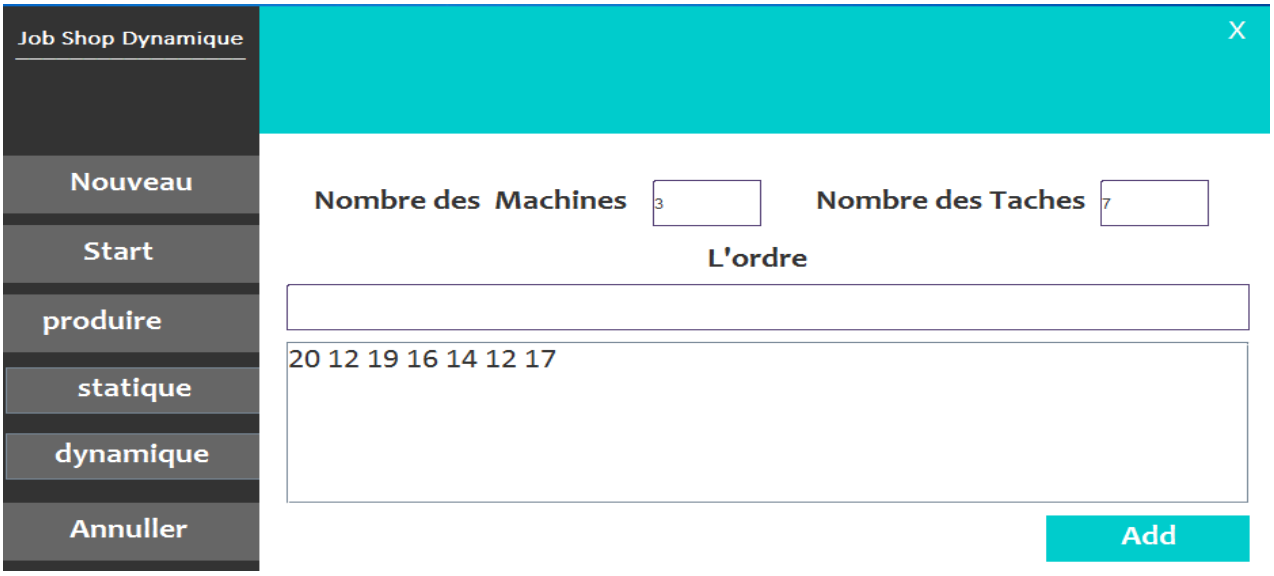
La figure 5.6 : la page d'entrée de l'application.

Le menu suivant :



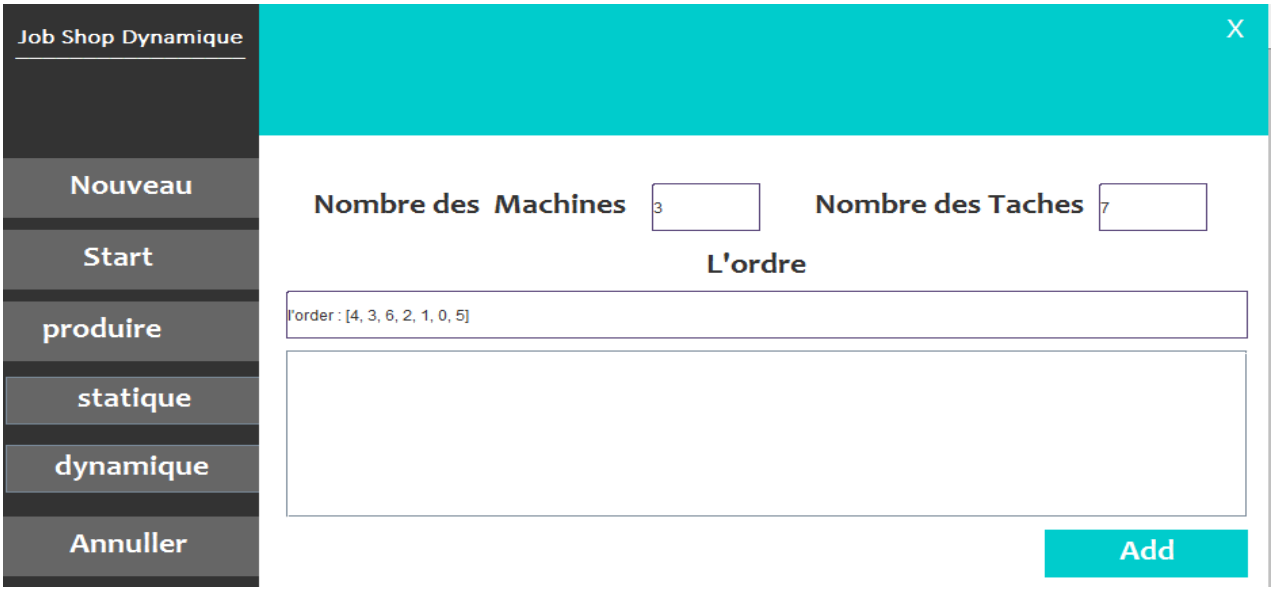
La figure 5.7: la page de l'interface.

La première étape d'utilisation est la détermination de l'instance de job shop dynamique qui constitue le problème à résoudre. Les instances doivent être de type Job Shop. Le modèle statique est constitué de m machines, J jobs et des durées de chaque machine.



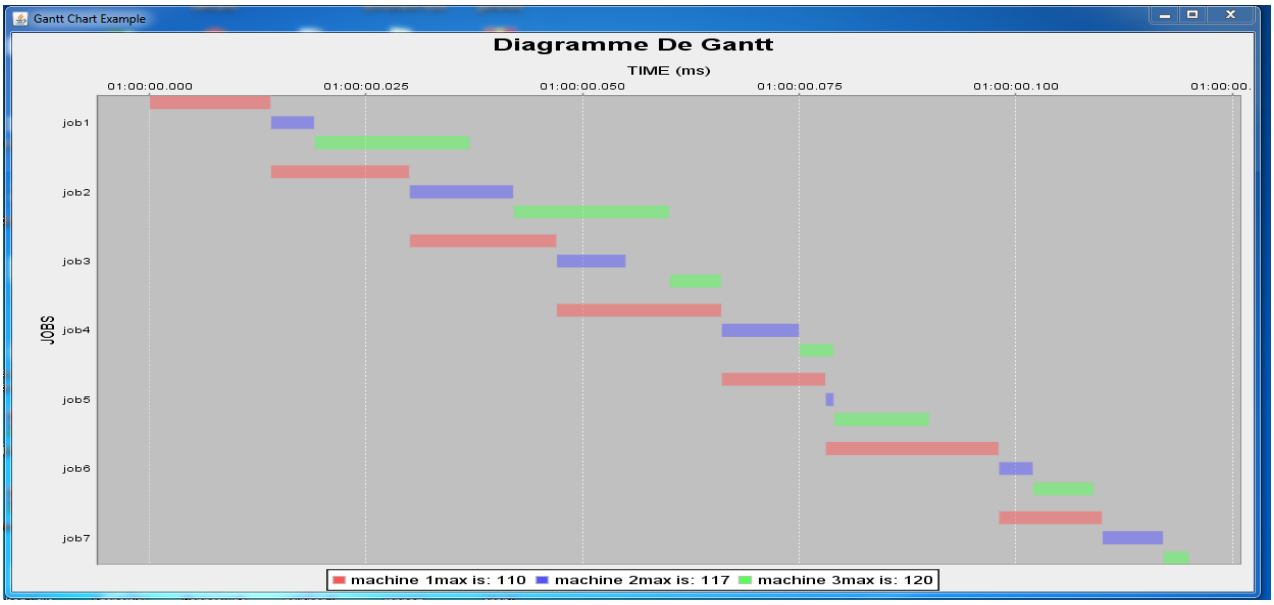
La figure 5.8 : Un exemple applique sur l'interface.

La figure 5.4 illustre l'exemple précédant de type job shop dans le chapitre 04 (Tableau 4.4) composé de 7 jobs et 3 machines. Un nombre de tâches et de machines sont en entrée, puis nous exécutons le bouton produire pour générer les listes de machines. Il viendra après, la saisie des durées des taches passants par chaque machine et exécuter sur le bouton Start qui affiche l'ordre d'ordonnancement des machines avec le diagramme de gant et le makespan (Cmax) de chaque machine.



La figure 5.9: Interface après exécution.

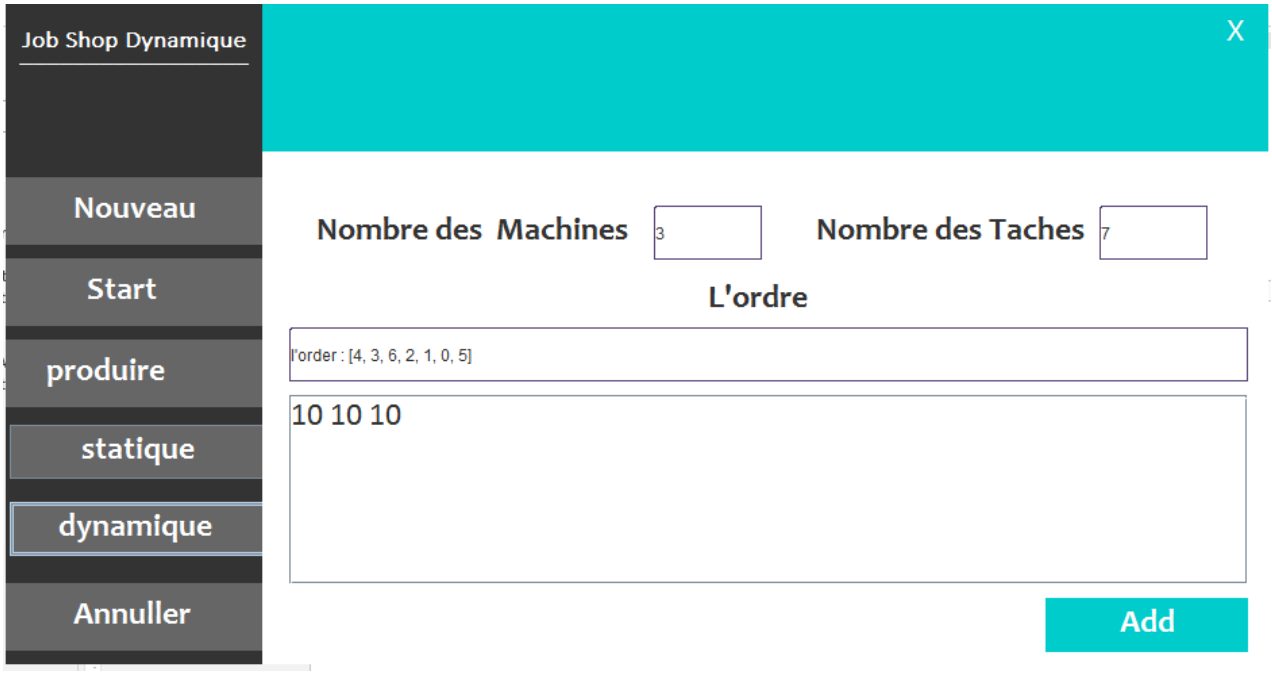
Après l'exécution le résultat présenté le diagramme de Gantt, et affiche aussi le Makespan (Cmax).



La figure 5.10: le diagramme de Gantt représentant la solution optimale de chaque machine.

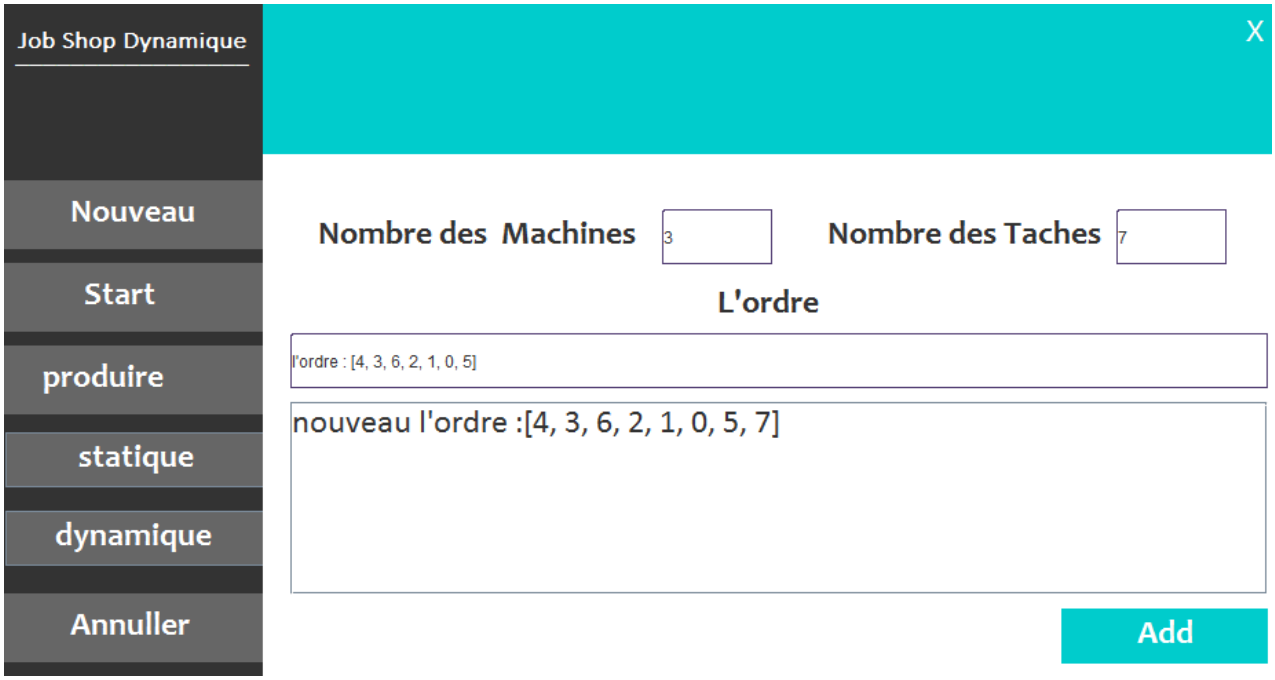
5.2 Modèle dynamique

Après le modèle statique nous présentons le modèle dynamique qui contient deux contextes d'insertion : insertion statique et insertion dynamique en temps réel.



La figure 5.11 : les durées de la nouvelle tâche.

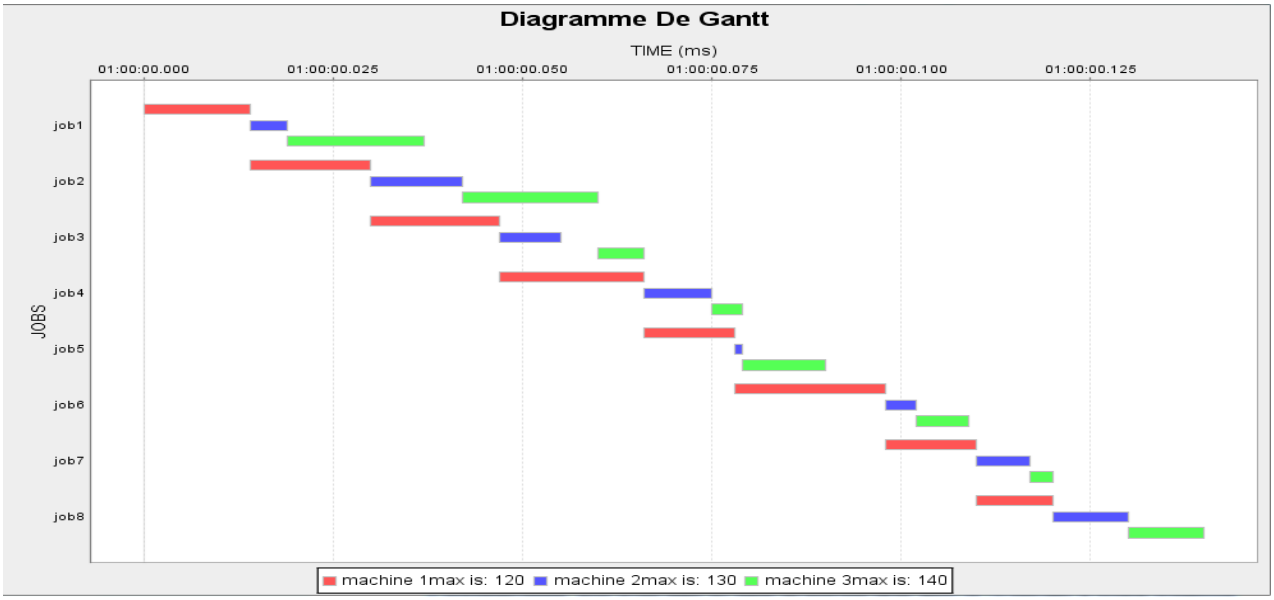
Dans l'insertion statique on va ajouter une nouvelle commande qui s'inscrit à la fin de l'ordre d'ordonnancement. On saisi les durées de la nouvelle tâche après on va exécuter le bouton statique et puis le bouton Start. Qui affiche le nouvel ordre d'ordonnancement des machines avec le diagramme de Gantt et le makespan (Cmax) de chaque machine.



La figure 5.12 : interface après insertion statique.

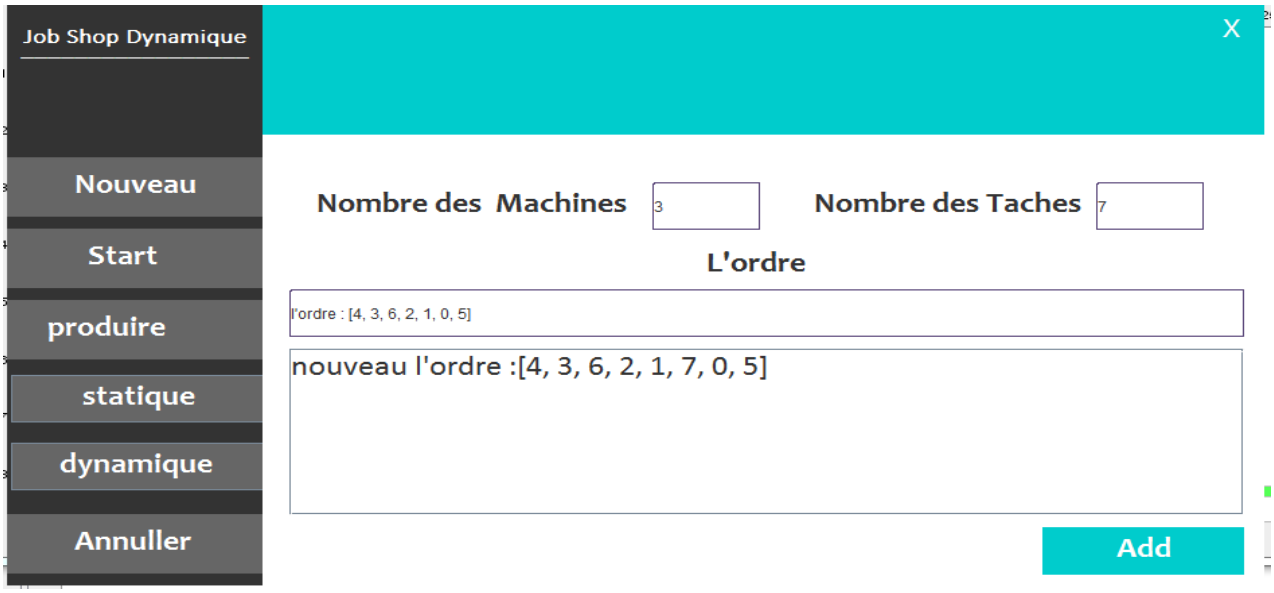
Le résultat présente le nouvel ordre et le diagramme de Gantt de l'insertion statique, et

affiche aussi le Makespan (Cmax).



La figure 5.13 : le diagramme de Gantt après insertion statique

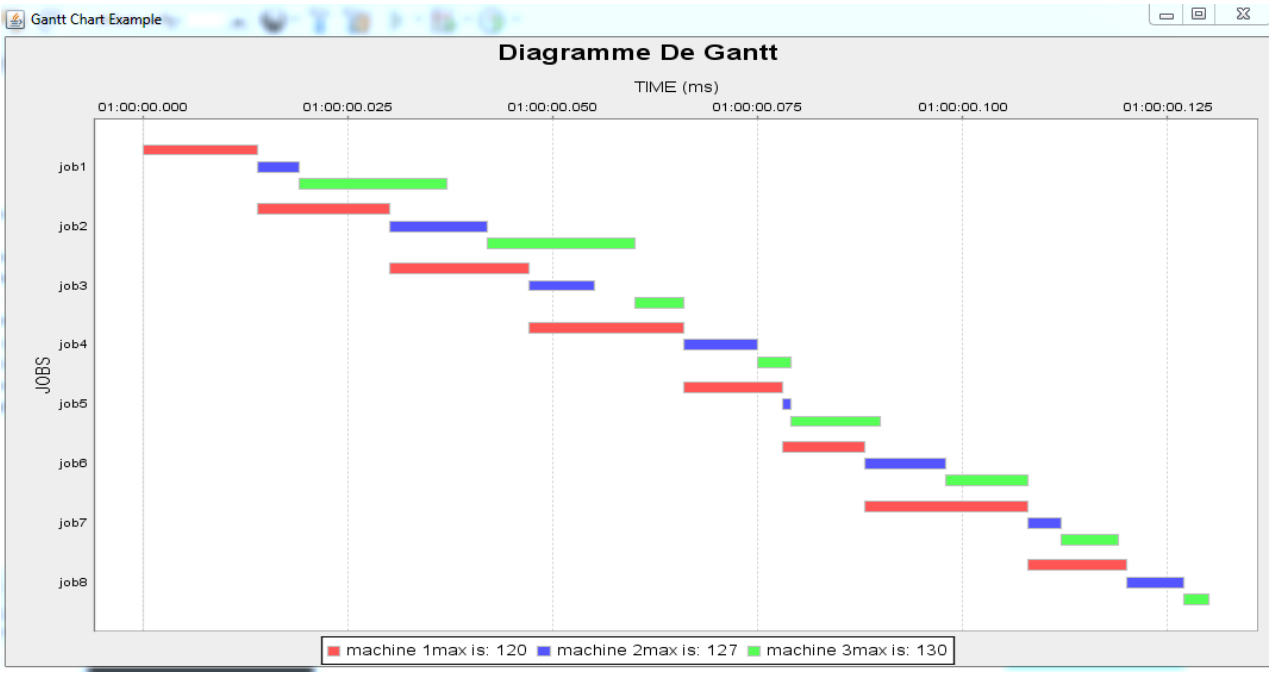
Dans l'insertion dynamique on va ajouter une nouvelle commande aléatoire dans d'ordre d'ordonnancement, nous avons saisi les durées de la nouvelle tâche, après, on va exécuter en cliquant sur le bouton dynamique puis le bouton Start :



La figure 5.14 :Le diagramme de Gantt après insertion statique

Après exécution, le résultat présente le nouvel ordre et le diagramme de Gantt d'insertion

dynamique (plan prévisionnel), et affiche aussi le Makespan (Cmax):



La figure 5.15 : le diagramme de Gantt après insertion dynamique

Conclusion générale

Nous avons commencé notre projet par l'étude de la fonction ordonnancement et sa résolution qui repose essentiellement sur la résolution des problèmes combinatoires. Ces problèmes consistent à la rechercher dans un ensemble de solutions possibles.

Nous avons vu les problèmes d'ordonnancement de type job shop, suivi d'une description du problème à étudier. Nous nous sommes focalisés sur notre méthode de résolution, à savoir le principe d'algorithme de Johnson et son application sur le job shop dans les deux cas : statique et dynamique.

Nous avons implémenté notre solution en utilisant le langage Java et où nous avons élaboré des tests expérimentaux. Les résultats obtenus sont discutés et analysés afin d'appartenir certaine conclusion.

La principale contribution de notre travail consigné dans ce mémoire est la résolution de l'un des problèmes d'ordonnancement combinatoire, et de manière plus particulière, le problème d'ordonnancement de type job shop dans un environnement dynamique. Ce problème est présenté par l'occurrence des nouvelles commandes qui doivent être insérées dans un système de production.

Notre travail a permis donc de développer un algorithme permettant de calculer l'ordre d'ordonnancement et d'insérer les nouvelles commandes dans un ordonnancement existant, sous la contrainte temps-réel. Cet algorithme est considéré comme efficace et donne de bons résultats dans le cadre des hypothèses définies auparavant et faciles à implémenter.

On peut donc résumer l'utilité de ce travail par les points suivants :

- L'algorithme de Johnson calcule l'ordonnancement minimisant le temps total d'exécution des tâches.

- La création de l'algorithme qui permet de générer des positions d'insertion possibles sous la contrainte succession sans interruption.

- Notre objectif est de trouver le Cmax minimum

- Notre travail a permis d'insérer les nouvelles commandes dans un ordonnancement existant, sous la contrainte temps-réel.

Il serait très intéressant de pouvoir améliorer certains autres aspects dans l'approche proposée qui sont envisagés en perspectives comme l'utilisation de méthodes performantes comme les algorithmes génétiques et l'algorithme d'abeille dans la phase de définition des positions d'insertion ainsi que la prise en compte d'autres critères pour la sélection des solutions tels que la fusion entre les deux critères (Makespan et coût de décalage).

BIBLIOGRAPHIES

[1] A. Hassam Ahmed, Développement et analyse de méthodes d'ordonnancement temps réel pour les systèmes flexibles de production, Tlemcen, Thèse de Doctorat, 2012.

[2] A. Karray, Contribution à l'ordonnancement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides, Tunis, Thèse de Doctorat, 2011.

[3] A. Liefooghe, L. Jourdan, M.Basseur, E.Talbi. Métaheuristique pour le flow-shop de permutation bi-objective stochastique. Revue des Sciences et Technologies de l'Information - Série RIA : Revue d'Intelligence Artificielle, Lavoisier, 2008,

[4] A. Hadri, L'ordonnancement par insertion en temps réel de la production dans un atelier flexible, Batna, Mémoire de Magister, 2012.

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

[6] H. Hentous, Flow Shop hybride sous contraintes, Rapport de DUA de recherche opérationnelle, Université de Joseph Fourier, 1995

[7] H. Boukef Ben Othman, Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques optimisation par algorithmes génétiques et essaims particulaires, Tunis, Thèse de Doctorat, 2009.

[8] H. Kellal, Les méthodes PERT et MPM pour l'ordonnancement des taches d'un projet, mémoire de master en Recherche, Université M'hamed Bougara Boumerdes, Faculté des Sciences, Département de Mathématiques 2015-2016

[9] H. Abdelkader, l'ordonnancement temps réel de la production dans un atelier flexible, Mémoire de Magister en Technologie Génie Industriel et Productique, universitéHadj Lakhdar Batna, 2011/2012

[10] I. Driss, Analyse d'un système job shop aspect ordonnancement, Batna, Thèse de Doctorat, 2016.

[11] J. Liouville Université du Littoral, zone universitaire de la Mi-Voix, bâtiment H. Poincaré 50, rue F. Buisson, BP 699, F-62228 Calais cedex.

[12] M. kebabla, Utilisation des stratégies Métaheuristique pour l'ordonnancement des ateliers de type Job Shop, Batna, Mémoire de Magister, 2008.

[13] M. Sakarovitch,« Graphes et Programmation Linéaire, Edition Hermann, Paris, 1984.

[14] W. Marino, Les métaheuristique : des outils performants pour les problèmes industriels, Université de Fribourg France Département d'informatique, 2001.

[15] M. Bombrun, L'optimisation par essaim particulaire pour des problèmes d'ordonnancement GOURGAND 2011.

[16] M. Larabi. Le problème de job-shop avec transport : modélisation et optimisation. Université Blaise Pascal - Clermont-Ferrand II, 2010.

[17] M. Meziane, 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.

[18] N. Mouhoub, Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet, Sétif, Thèse de Doctorat, 2011.

[19] V. Giard, Gestion de production, Paris Economica, 1988.

[20] Y. Bahmani, Optimisation multicritère de l'ordonnancement des activités de la production et de la maintenance intégrées dans un atelier Job Shop, Batna, Thèse de Doctorat, 2017.

Les sites web

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

[b] www.tleroux_genetic_algorithm/fonctionnement.

[c] www.langage-java-histoire-caracteristiques-popularite.

[d] www.unit.eu/cours/EnsROtice/module_avance_thg_voo6/co/johnson.html#menu

Résumé

Les problèmes d'ordonnancement d'atelier sont souvent classés NP-Difficiles. Leur résolution nécessite des méthodes dédiées à leur degré de complexité ; pour cette raison plusieurs heuristiques et méta-heuristiques ont été conçues.

L'objectif de ce mémoire est de proposer des méthodes pour la résolution de ce problème d'ordonnancement job shop dans un environnement dynamique, en vue de minimiser le makespan et comment insérer une nouvelle commande dans plan prévisionnel.

Dans ce travail, nous avons insisté sur le principe d'algorithme de Johnson qui calcule l'ordonnancement minimisant le temps total d'exécution des taches.

Mots clés : Ordonnancement, job shop dans un environnement dynamique, d'algorithme de Johnson, makespan

ملخص

إن مشاكل الجدولة في الغالب تعتبر من مشاكل التحسين الصعبة ن ب، حلها يتطلب أساليب مختلفة نظرا لتعقيداتها. لهذا السبب تم تصميم العديد من الاستدلال وفوقية الاستدلال.

الهدف من هذه الأطروحة هو اقتراح طرق لحل مشكلات الجدولة الديناميكية لمحل العمل، وذلك من أجل التقليل إلى أدنى حد من الكيفية. وكيفية إدراج أمر جديد في الخطة المؤقتة.

في هذا العمل: اعتمدنا على مبدأ خوارزمية جونسون الذي يحسب الجدولة ويقلل الوقت الإجمالي لتنفيذ المهمة.

الكلمات المفتاحية:الجدولة، الجوب شوب الديناميكي، خوارزمية جونسون،وقت الاتمام الأكبر

Abstract

Job Shop scheduling problems are considered as one of the NP-Hard classes. Their resolution requires methods dedicated to their degree of complexity for this reason several heuristics and meta-heuristics have been conceived.

The objective of this thesis is to propose methods for solving dynamic job shop scheduling problems, in order to minimize the makespan. And how to insert a new task in the provisional plan.

In this work: we have based on Johnson algorithm, which calculates the scheduling and minimize the total time of task execution.

Keywords: Scheduling, Dynamic Job Shop, Johnson Algorithm, the makespan.