الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE SAAD DAHLEB – BLIDA 1

FACULTE DES SCIENCES

DEPARTEMENT DE MATHEMATIQUES

MEMOIRE de fin d'études

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques

Spécialité : Recherche Opérationnelle

Thème

Problème à acheminement multiple dans un environnement dynamique

Présenté par : Melle Mammeri Sabrina

Soutenu publiquement le 15 Novembre 2020 devant le jury composé de :

Mme Kerdjoudj Samia	MCB à l'université de Blida 1	Présidente.
Mme Messaoudi Nadia Amel	MCB à l'université de Blida 1	Promotrice.
Mr El Mossaoui Hichem	MCB à l'université de Blida 1	Examinateur.

Promotion : 2019 / 2020

Dédicace

Je dédie ce modeste travail à :

- Mes parents : qui sont la cause de la lumière de mes jours, la source de mes forces, la flamme de mon cœur, ma vie et mon bonheur.

- A mon très cher père MAMMERI RACHID, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; merci pour les valeurs nobles, l'éducation et le soutient permanent venu de vous. Papa je t'aime.

- A ma très chère mère MAMMERI FOUZIYA, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude. Maman je t'aime.

A mes beaux-frères et ma sœur : Rafik, Ayoub, Hichem, Ikram dans tous mes moments d'examens par leur soutien moral et ses belles surprises sucrées. Je vous souhaite un avenir plein de joie, de bonheur, de réussite et de sérénité. Je vous exprime à travers ce travail mes sentiments de fraternité et d'amour.

Remerciements

Je remercie en particulier Mme Messaoudi Nadia Amel, pour l'honneur qu'elle m'a fait de bien vouloir m'encadrer, et pour les conseils donnés lors de la réalisation de ce travail.

Je remercie mes remerciements aux membres de jury :

Mme Kerdjoudj Samia et Mr El Mossaoui Hichem pour avoir accepté de me prêter 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.

Résumé

L'objectif de ce mémoire consiste à résoudre le problème d'ordonnancement job-shop à plusieurs machines dans un environnement dynamique, dans le but de minimiser le temps total d'exécution des tâches (le makespan) et de trouver une façon de comment insérer une nouvelle commande dans un plan prévisionnel. Nous avons utilisé l'heuristique de Dannenbring pour réduire le nombre de machine à deux, afin d'appliquer le principe de l'algorithme de Johnson pour calculer l'ordonnancement minimisant le makespan.

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

Abstract

The objective of this work is to solve the problem of job-shop scheduling with several machines in a dynamic environment, in order to minimize the total time of execution of the tasks (the makespan) and to find a way of how to insert a new order in a provisional plan. We used Dannenbring's heuristic to reduce the number of machines to two, in order to apply the principle of Johnson's algorithm to calculate the scheduling minimizing the makespan.

Keywords: Scheduling, job-shop in a dynamic environment, Johnson's algorithm, makespan.

ملخص

الهدف من هذه الرسالة هو حل مشكلة جدولة " جوب- شوب " باستخدام العديد من الأجهزة في بيئة ديناميكية من أجل تقليل الوقت الإجمالي لتنفيذ المهام (التكوين) وإيجاد طريقة لكيفية إدخال أمر جديد في خطة مؤقتة. استخدمنا استدلال دانبغينغ لتقليل عدد الآلات إلى اثنين ، من أجل تطبيق مبدأ خوارزمية جونسون لحساب الجدولة لتقليل حجم التكوين.

الكلمات الرئيسية: الجدولة ، "جوب- شوب" في بيئة ديناميكية ، خوارزمية جونسون ، مكسب.

Table des matières

Chapitre II : Modélisation d'un problème job-shop

Chapitre III : Méthode de résolution d'un problème job-shop

Chapitre IV : Conception et réalisation

Liste des Figures

Liste des Tableaux

Introduction générale

La recherche opérationnelle [14] est une discipline dont le but est de fournir des méthodes pour répondre à un type précis de problème. Sa vocation scientifique est donc de construire des modèles formels d'aide à la décision, en particulier des modèles liés à des problèmes d'optimisation, et de proposer des méthodes de résolution efficaces à ces modèles.

La théorie de l'ordonnancement [1] est une branche de la recherche opérationnelle .Elle s'intéresse par exemple au calcul des dates d'exécution optimales des tâches. Pour cela, il est très souvent nécessaire d'affecter en même temps les ressources nécessaires à l'exécution de ces tâches. Ces ressources sont par exemple des machines, des appareils, des hommes ou des processeurs.

Les différentes données d'un problème d'ordonnancement sont les tâches à exécuter, les ressources et les contraintes. Dans le système de production, un problème d'ordonnancements est défini comme étant la localisation dans le temps et l'espace la réalisation d'un ensemble de tâches, compte tenu des contraintes temporelles (une date de début avec une durée ou une date de fin) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches.

Le problème d'ordonnancement consiste à trouver une séquence optimale pour l'exécution de n tâches sur m machines afin d'optimiser une fonction objective et de déterminer également les dates de début et de fin d'exécution de chaque tâche. La résolution de ce problème de manière optimale s'avère dans la plupart des cas impossible à cause de son caractère fortement combinatoire.

On peut rencontrer les problèmes d'ordonnancement dans de très nombreux domaines [1] : les systèmes industriels de production (activités des ateliers en gestion de production et problèmes de logistique), les systèmes informatiques (les tâches sont les programmes et les ressources sont les processeurs, la mémoire...), les systèmes administratifs (gestion du personnel, emplois du temps,...), les systèmes de transport, la construction.

Les problèmes d'ordonnancement d'ateliers [10] constituent pour les entreprises une des difficultés importantes de leurs systèmes de gestion et de pilotage de la production. En effet, c'est à ce niveau que doivent être prises en compte les caractéristiques réelles multiples et complexes des ateliers. Dans ce type d'ordonnancement, les ressources sont généralement des

machines, et chaque travail à ordonnancer concerne un produit ou un lot de produits à fabriquer en respectant les gammes de fabrication.

Il y a plusieurs types d'ordonnancement dans les ateliers comme le problème de job-shop, flow-shop et open-shop.

Nous pouvons distinguer deux catégories de problèmes d'ordonnancement. En fonction du degré de connaissance que l'on a du problème à résoudre, on parlera de problème statique [9], lorsque les tâches à ordonner sur une période ainsi que l'état initial de l'atelier sont connus au début de la période, mais il est très difficile de gérer les tâches de haute priorité (événements extraordinaires) le cas ou, nous traiterons le deuxième problème, qui est le problème dynamique [9] : lorsque les décisions sont à prendre sur la période mais toutes les tâches à réaliser sur cette période ne sont pas connues au début de la période.

Dans notre travail, nous nous somme s'intéresse au problème de type job-shop a au moins deux machines dans le cas statique et dynamique.

L'algorithme de Johnson [12] est utilisé pour la résolution du problème à deux machines. Lorsque le nombre de machine dépasse deux, ici nous avons utilisé l'heuristique de Dannenbring [4]. Elle combine deux heuristiques celle de Palmer [4] et celle de Campbell Dudek et Smith [4]. Cette heuristique nous permet de réduire le nombre de machines à deux afin qu'on puisse appliquer la méthode de Johnson.

Ce mémoire est constitué de quatre chapitres :

Dans le premier chapitre, nous allons donner des notions générales sur le problème d'ordonnancement et sa formulation. Nous présentons également les problèmes d'ordonnancement dans les ateliers et leurs types, ainsi que les méthodes de représentations et la complexité du problème.

Dans le deuxième chapitre, nous allons présenter le problème d'ordonnancement de type job- shop, sa modélisation et sa complexité.

Le troisième chapitre est consacré aux méthodes de résolution d'un problème de job-shop. Nous décrivons l'algorithme de Johnson pour deux machines et l'heuristique de Dannenbring avec des exemples illustratifs.

Le chapitre quatre, portera sur l'implémentation de notre application ainsi que l'élaboration des tests expérimentaux.

Chapitre I :

Généralités sur l'ordonnancement

I.1 Ordonnancement

L'ordonnancement [1] fait partie de cette catégorie de problèmes que l'on regroupe sous le nom d'optimisation combinatoire. Il vise à organiser, à agencer « au mieux » des tâches (ou jobs) à exécuter au cours du temps sur un certain nombre de ressources. Par exemple, la réalisation de microcontrôleur nécessite un grand nombre d'opérations. Ces dernières ne peuvent s'effectuer dans n'importe quel ordre. De plus, certaines tâches demandent plus de temps que d'autre. Chercher à réaliser un microcontrôleur en un temps minimum apparaît donc comme un problème d'ordonnancement.

Dans le domaine industriel, l'ordonnancement consiste à organiser dans le temps la réalisation d'une suite de tâches, en prenant en compte les contraintes de production [15] :

- Temporelles : délais requis, retards, priorités
- Techniques : contraintes d'enchaînement, technologie machines
- Capacitaires : disponibilité des ressources

Dans la plupart des problèmes d'ordonnancement, les machines sont considérées disponibles. Dans un cas réel, cette hypothèse n'est pas toujours vraie. En effet, les machines peuvent connaître des périodes d'indisponibilité correspondant à des périodes de maintenance préventive, pannes, ou un changement d'outils.

L'ordonnancement d'atelier [17] consiste à organiser dans le temps le fonctionnement d'un atelier pour utiliser au mieux les ressources humaines et matérielles disponibles dans le but de produire les quantités désirées dans le temps imparti.

Aussi elle consiste à exploiter au mieux des moyens limités, les machines, pour réaliser un ensemble varié de produits, les lots.

La complexité ne réside pas dans le processus de fabrication prédéterminé mais plutôt dans la combinatoire qui naît de la prise en compte des limitations de ressources ,et elle est à prévoir l'enchaînement de toutes les opérations élémentaires nécessaires à la réalisation des ordres de fabrication sur les ressources de production tout en tenant compte des contraintes internes et externes.

I.2 Les Problèmes d'ordonnancement

Nous pouvons distinguer deux catégories de problèmes d'ordonnancement. En fonction du degré de connaissance que l'on a du problème à résoudre, on parlera de [9] :

- Problème statique : lorsque les tâches à ordonner sur une période ainsi que l'état initial de l'atelier sont connus au début de la période.
- Problème dynamique : lorsque les décisions sont à prendre sur la période mais toutes les tâches à réaliser sur cette période ne sont pas connues au début de la période. De plus, un ordonnancement se décompose en deux parties toujours présentes dans l'atelier, mais pouvant revêtir une importance variable :
- L'ordonnancement prédictif : consiste à prévoir "à priori" un certain nombre de décisions en fonction de données prévisionnelles et d'un modèle de l'atelier.
- L'ordonnancement réactif : consiste à adapter les décisions prévues en fonction de l'état courant du système et des déviations entre la réalité et le modèle (ce qui est prévu en théorie).

I.3 Formulation d'un problème d'ordonnancement

Dans un problème d'ordonnancement, cinq notions fondamentales interviennent : les tâches, les ressources, les contraintes, les critères et les objectifs. Dans ce qui suit, ordonnera une définition détaillée de chacun de ces notions. [11]

I.3.1 Les tâches :

Une tâche est une entité élémentaire localisée dans le temps par une date de début et/ou de fin, dont la réalisation nécessite une durée, et qui consomme un moyen selon une certaine intensité. Certains modèles intègrent la notion de date due, une date à laquelle la tâche doit être finie ; dans ces cas, le retard induit une pénalité.

Selon les problèmes, les tâches peuvent être exécutées par morceaux, ou doivent être exécutées sans interruption ; on parle alors respectivement de problèmes préemptifs et non préemptifs. Lorsque les tâches ne sont soumises à aucune contrainte de cohérence, elles sont dites indépendantes.

Plusieurs tâches peuvent constituer une activité et plusieurs activités peuvent définir un processus.

Une tâche « i » est localisée dans le temps par une date de début « t_i » et une date de fin « c_i », dont la réalisation est caractérisée par une durée positive « p_i » telle que :

$p_i = c_i - t_i$. Certaines caractéristiques relatives à l'exécution d'une tâche sont définies ainsi :

- Une date de disponibilité « r_i » qui correspond à la date de début au plutôt.
- Une date d'échéance « d_i » qui correspond à la date de fin au plus tard.
- Un poids « w_i » qui représente le facteur de priorité qui dénote l'importance de la tâche *i* relativement aux autres.

La Figure suivante donne une représentation de la tâche en désignant ses principales caractéristiques :

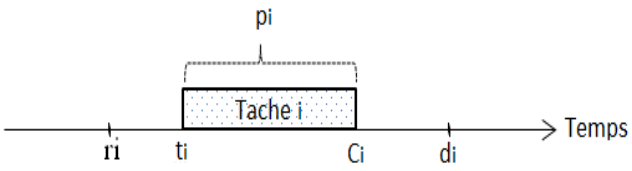


Figure I.1 : Représentation de la tâche

I.3.2 Les ressources :

Une ressource est un moyen matériel ou humain (machine, homme,..) nécessaire pour exécuter les tâches.

Plusieurs types de ressources sont à distinguer. Une ressource est renouvelable si après avoir été allouée à une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines, l'équipement en général) ; la quantité de ressource utilisable à chaque instant est limitée. Dans le cas contraire, elle est consommable (matières premières, budget) ; la consommation globale (ou cumul) au cours du temps est limitée. Une ressource est doublement contrainte lorsque son utilisation instantanée et sa consommation globale sont toutes deux limitées (l'argent en est un bon exemple).

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Sa courbe de disponibilité est en général connue a priori, sauf dans les cas où elle dépend du placement de certaines tâches génératrices.

On distingue par ailleurs principalement dans le cas de ressources renouvelables les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois (machine-outil, robot manipulateur) et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité (équipe d'ouvriers, poste de travail).

I.3.3 Les contraintes :

Les contraintes spécifient quels sont l'ordonnancement réalisable (en respectant les capacités physiques, les engagements contractuels, les objectifs commerciaux, …etc.).

On distingue les différents types de contraintes :

I.3.3.1 Contraintes potentielles :

Elles ont les formes générales suivantes :

Si i et j sont deux tâches, on distingue les contraintes : $t_j - t_i \geq a_{ij}$
Quand la tâche j commence à la fin de la tâche i, dans ce cas, on dit qu'il y a succession simple.
Où : t_j et t_i sont les époques de début des tâches i et j, $a_{ij} = pi$ (p_i étant la durée de la tâche i).

- Date de disponibilité : la tâche i ne peut débuter avant une certaine date r_i (livraison d'une matière première,…)
- Date butoir : la tâche j doit être terminée avant une certaine date d_i
- Contrainte d'antériorité (précédence) : la tâche i doit attendre qu'une tâche j soit achevée pour pouvoir débuter.

I.3.3.2 Contraintes disjonctives

Deux tâches i et j sont en disjonction, si elles ne peuvent être exécutées simultanément.

- Leurs intervalles d'exécution doivent être disjoints.
- Contrainte : $t_i - t_j \geq p_j$ ou $t_j - t_i \geq p_i$

Où :

p_j : la durée d'exécution de la tâche j

p_i : la durée d'exécution de la tâche i

I.3.3.3 Contrainte cumulatives :

- Généralisation des contraintes disjonctives à des ressources non unitaires.
- Généralement chaque ressource *k* a une capacité $a_k(t)$
- A chaque instant *t*, l'ensemble des activités ne doit pas requérir plus que la capacité $a_k(t)$ de la ressource *k*.

I.3.4 Les critères :

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi.

Les critères dépendant d'une application donnée sont très nombreux ; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la plus satisfaisante dépend du ou des critères préalablement définis, pouvant être classés suivant deux types, réguliers et irréguliers.

Les critères réguliers constituent des fonctions décroissantes des dates d'achèvement des opérations. Quelques exemples sont cités ci-dessous :

- la minimisation des dates d'achèvement des actions,
- la minimisation du maximum des dates d'achèvement des actions,
- la minimisation de la moyenne des dates d'achèvement des actions,
- la minimisation des retards sur les dates d'achèvement des actions,
- la minimisation du maximum des retards sur les dates d'achèvement des actions.

Les critères irréguliers sont des critères non réguliers, c'est-à-dire qui ne sont pas des fonctions monotones des dates de fin d'exécution des opérations, tels que :

- la minimisation des encours,
- la minimisation du coût de stockage des matières premières,
- l'équilibrage des charges des machines,
- l'optimisation des changements d'outils.

I.3.5 Les objectifs :

Dans la résolution d'un problème d'ordonnancement, on peut choisir entre deux grands types de stratégies, visant respectivement à l'optimalité des solutions, ou plus simplement à leur admissibilité. [11]

L'approche par optimisation suppose que les solutions candidates à un problème puissent être ordonnées de manière rationnelle selon un ou plusieurs critères d'évaluation numériques, construits sur la base d'indicateurs de performances. On cherchera donc à minimiser ou maximiser de tels critères. On note par exemple ceux :

Liés au temps :

- le temps total d'exécution ou le temps moyen d'achèvement d'un ensemble de tâches.
- le stock d'en-cours de traitement.
- différents retards (maximum, moyen, somme, nombre, etc.) ou avances par rapport aux dates limites fixées.

Liés aux ressources :

- la quantité totale ou pondérée de ressources nécessaires pour réaliser un ensemble de tâches.
- la charge de chaque ressource.
- liés à une énergie ou un débit.
- liés aux coûts de lancement, de production, de transport, etc., mais aussi aux revenus, aux retours d'investissements.

I.4 Domaine d'utilisation

L'ordonnancement peut apparaitre dans plusieurs domaines, on peut les illustrés par la figure suivante [7] :

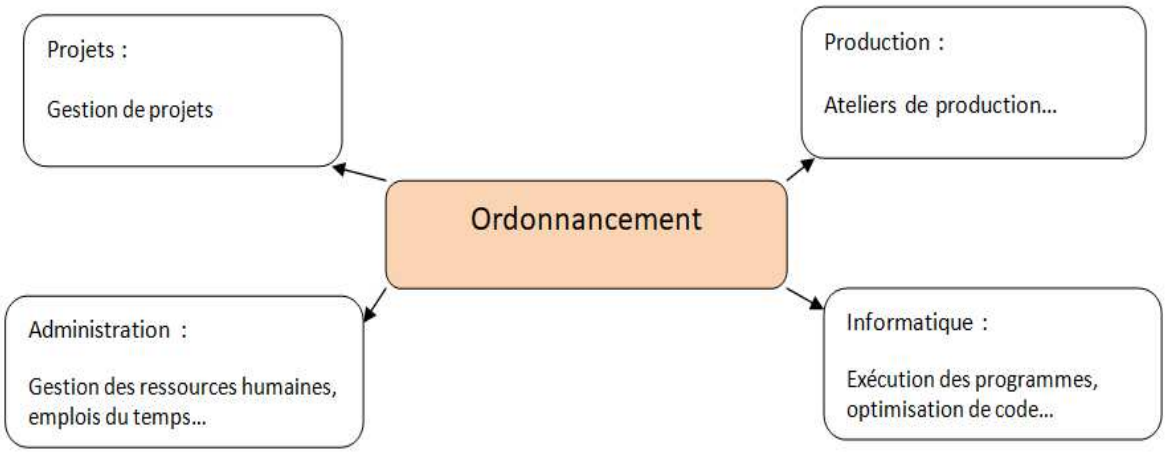


Figure I.2 : Domaine d'utilisation de l'ordonnancement

I.5 Les problèmes d'ordonnancement d'ateliers

Définition :

Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution.

Les problèmes d'ordonnancement, apparaissent dans tous les domaines de l'économie : l'informatique, la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emploi du temps). Les tâches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni immédiate, ni triviale.

Enfin, il faut programmer les tâches de façon à optimiser un certain objectif qui sera, suivant le cas, la minimisation de la durée totale (c'est le critère le plus fréquemment employé) ou le respect des dates de commande ou de lissage des courbes de main d'œuvre ou encore la minimisation d'un coût. D'une manière générale, trois types d'objectifs sont essentiels dans la résolution des problèmes d'ordonnancement : l'utilisation efficace des ressources, un délai d'exécution des tâches aussi faible que possible et le respect des dates d'achèvement prescrites à l'avance.

Dans un problème d'atelier, une pièce doit être usinée ou assemblée sur différentes machines.

Chaque machine est une ressource disjonctive, c'est-à-dire qu'elle ne peut exécuter qu'une tâche à la fois, et les tâches sont liées exclusivement par des contraintes d'enchaînement. Plus

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

Dans le cas des problèmes d'atelier, une tâche est une opération, une ressource est une machine et chaque opération nécessite pour sa réalisation une machine.

Dans le modèle de base de l'ordonnancement d'atelier :

- L'atelier est constitué de *m* machines, *n* travaux (jobs), disponibles à la date 0 et qui doivent être réalisés,
- Un travail *« i »* est constitué de *« n_i »* opérations, l'opération *j* du travail *i*

est notée (*i, j*).

$(i,1) < (i,2) <...< (i,n_i)$	si le travail *« i »* possède une gamme, (*A*<*B* signifie *A* précède *B*).

- Une opération (i, j) utilise la machine m_{ij} pendant toute sa durée p_{ij} et ne peut être interrompue.

Un atelier se détermine par le nombre de machines qu'il contient et par son type. Une classification peut exister selon le nombre des machines et l'ordre d'utilisation des machines, pour réaliser un travail (par exemple fabrication d'un produit qui dépend de la nature de l'atelier). [1]

I.6 Typologie des problèmes d'ordonnancements dans les ateliers

Une typologie [10] des problèmes d'ordonnancement dans un atelier peut s'opérer selon le nombre et la nature des machines ainsi que l'ordre d'enchaînement des opérations (gamme de fabrication). Deux grandes familles de problèmes d'ordonnancement se présentent. La première famille regroupe les problèmes pour les quels chaque tâche nécessite une seule opération. La deuxième regroupe ceux dont les tâches nécessitent plusieurs opérations.

I.6.1 Problèmes à une opération :

- **Problèmes à une machine** : les problèmes d'atelier à une machine en anglais appelé : (single machine problem) consistent à ordonnancer, sur une seule machine, des jobs constitués d'une seule opération.

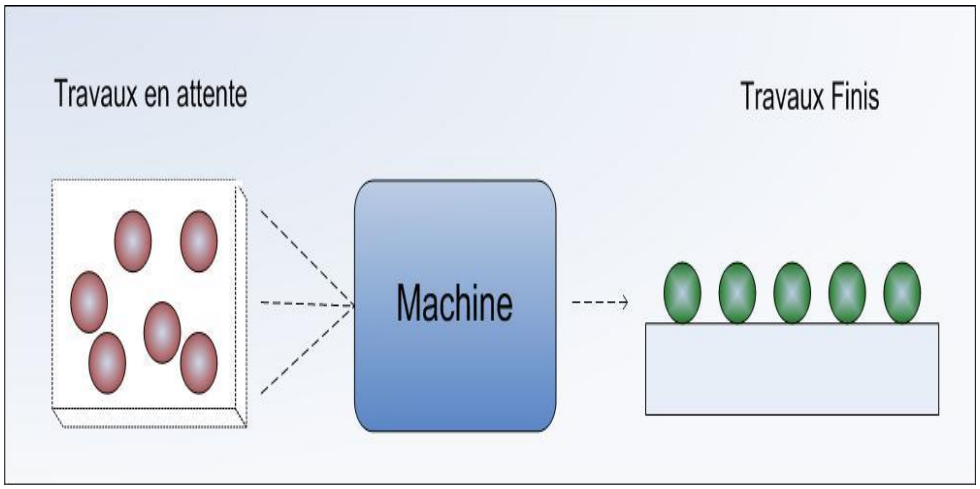


Figure I.3 : Représentation des machines uniques

- **Problèmes à machines parallèles :** les problèmes d'atelier à machines parallèles (parallel machine problem) sont une généralisation des problèmes à une machine. Ce type d'atelier se caractérise par le fait que chaque opération peut être réalisée par n'importe quelle machine, disposée en parallèle, mais n'en nécessite qu'une seul. Le problème d'ordonnancement consiste donc à déterminer l'affectation des opérations aux machines puis le séquencèrent de ces opérations sur chaque machine.

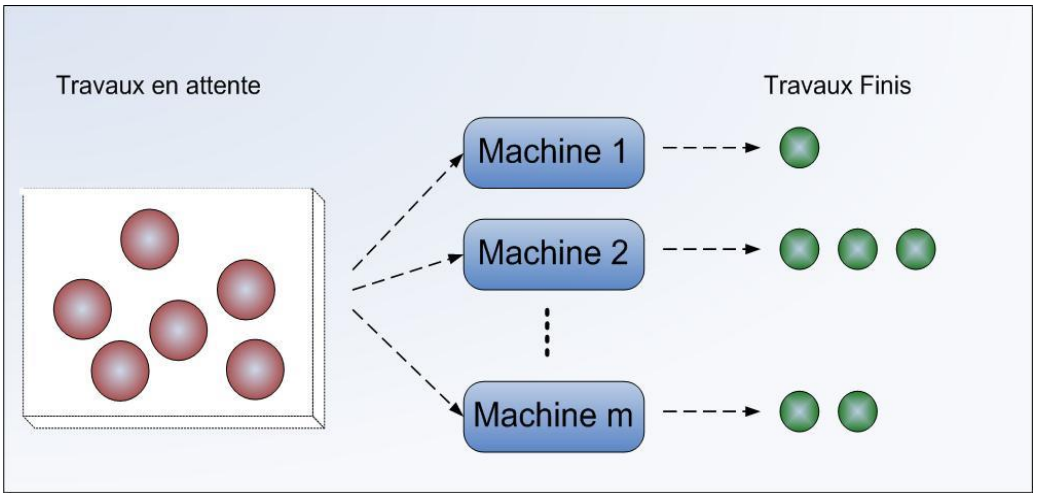


Figure I.4 : Représentation des machines parallèles

Dans le dernier cas, il est possible de distinguer trois classes de machines :

- **Machines parallèles identiques** (identical parallel machines) : la durée d'exécution des opérations est la même sur toutes les machines.
- **Machines parallèles uniformes** (uniform parallel machines) : la durée d'exécution des opérations varie uniformément en fonction de la performance de la machine choisie.
- **Machines parallèles indépendantes** ou non liées (unrelated parallel machines) : les durées opératoires dépendent complètement des machines utilisées.

I.6.2 Problèmes à plus d'une opération :

Les problèmes de la deuxième catégorie sont dits problèmes d'atelier du fait de la nécessité du passage de chaque tâche sur deux ou plusieurs machines dédiées suivant le mode de passage des opérations sur les différentes machines, trois types d'ateliers sont distingués à savoir :

I.6.2.1 Problèmes Flow-Shop :

Les ateliers de type Flow-Shop appelés également ateliers à cheminement unique, il s'agit d'un ensemble de m machines disposées en séries. Toutes les opérations de tous les jobs passent par les machines dans le même ordre (flot unidirectionnel).

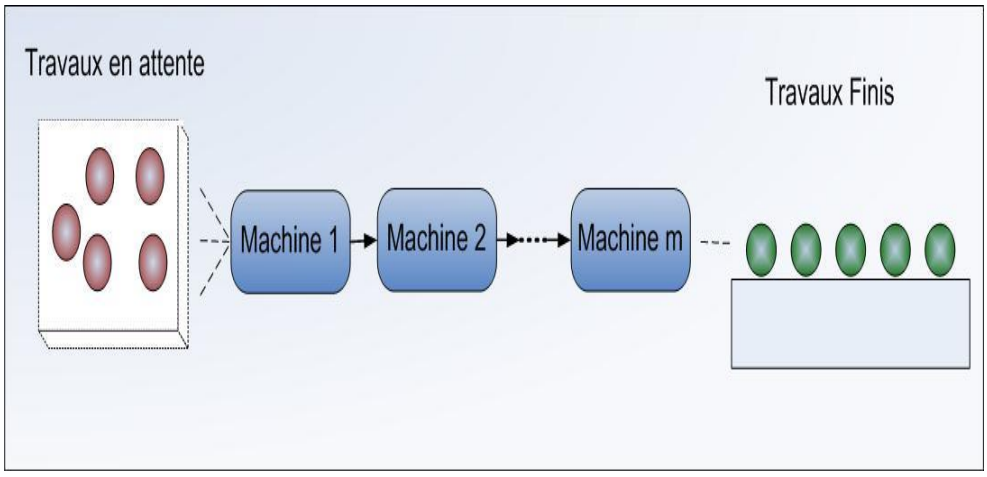


Figure I.5 : Représentation d'ateliers à cheminement unique (flow-shop)

I.6.2.2 Problèmes job-Shop :

Dans cette classe d'ateliers, appelés aussi ateliers à cheminements multiples, chaque opération passe sur les machines dans un ordre fixé, mais à la différence du Flow-Shop, cet ordre peut être différent pour chaque tâche (flot multidirectionnel).

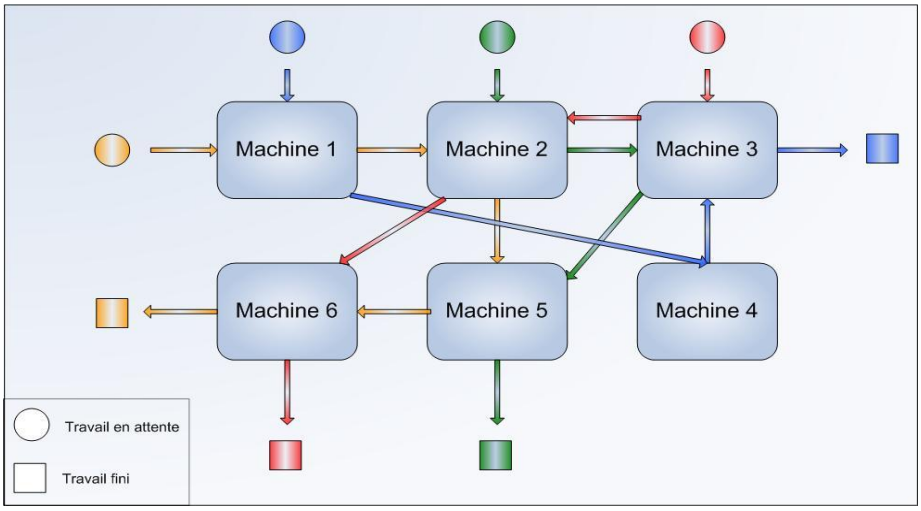


Figure I.6 : représentation d'ateliers à cheminement multiple (job-shop)

I.6.2.3 Problèmes Open-Shop :

Dans cette classe d'ateliers, appelés aussi ateliers à cheminement libre, les gammes opératoires des différents jobs ne sont pas fixées a priori contrairement au problème d'atelier Job-Shop. Les opérations d'un même job peuvent donc être exécutées dans un ordre quelconque. Le problème consiste d'une part à déterminer le cheminement de chaque job et d'autre part à ordonnancer les jobs en tenant compte des gammes trouvées.

I.7 Méthodes de représentation d'un problème job-shop

Les méthodes d'ordonnancement des tâches permettent d'avoir une représentation graphique (immuable ou non) d'une réalisation en représentant chaque opération (ou tâche) par un arc, une liaison, ou un rectangle qui peut être proportionnel ou non à la durée. Ce graphique dans tous les cas permet le positionnement relatif des opérations dans le temps. Il existe trois sortes de présentations possibles d'un problème d'ordonnancement : le diagramme de GANTT, le graphe potentiel-tâches (MPM), et la méthode (PERT).[5]

I.7.1 Le diagramme de GANTT :

Ce type de diagramme [5] a été mis au point par un Américain Henry GANTT en 1910, celui-ci indique, selon une échelle temporelle l'occupation des machines par les différentes taches, les temps morts, et les éventuelles indisponibilités des machines dues au changement entre tâches, par exemple :

Les tâches T1, T2 et T3 sont traitées par la machine M1, M2 et M3 comme le montre la figure I.7. Donc la durée que va prendre l'aboutissement des trois tâches sur les trois machines est 12h.

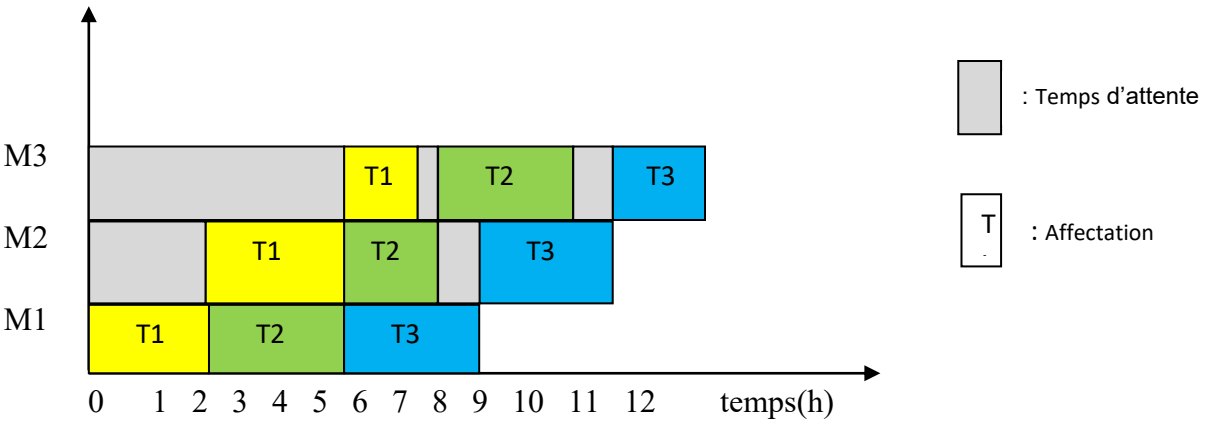


Figure I.7 : Exemple illustrative du diagramme de GANTT

I.7.2 La méthode PERT :

PERT (en anglais : *Program Evaluation and Review Technique*) [5] est une méthode conventionnelle utilisable en gestion de projet, ordonnancement et planification développée aux États-Unis par la marine de guerre américaine dans les années cinquante.

Elle fournit une méthode et des moyens pratiques pour décrire, représenter, analyser et suivre de manière logique les tâches (en) et le réseau des tâches à réaliser dans le cadre d'une action à entreprendre ou à suivre.

Le diagramme PERT représente le planning des travaux par un graphe de dépendances. Son formalisme en réseau se focalise sur l'interconnexion des tâches à effectuer et sur le calcul des chemins critiques. Une différence importante avec le diagramme de Gantt est l'échelle de temps conventionnelle du diagramme PERT qui représente un enchaînement de tâches et non des durées ou un calendrier.

Elle consiste à associer à un problème d'ordonnancement un graphe constitué d'un ensemble de nœuds reliés entre eux par des arcs où :

- Chaque sommet correspond à un instant de déclenchement d'exécution d'une tâche représentée par l'arc sortant du nœud et également à un instant d'achèvement d'une tâche représentée par l'arc entrant un nœud.
- Sur les arcs, nous trouvons généralement les durées d'exécution associées aux tâches.

Dans le cas où la valeur d'un arc est égale à 0 cela présente une contrainte de précédence.

I.7.3 La méthode des potentiels :

Elle a été développée vers la fin des années 50 parallèlement à la méthode PERT. Elle est appelée également la méthode MPM (méthode des potentiels Metra) ou encore méthode des potentiels tâches. Elle est basée aussi sur une modélisation par graphe constitué d'un ensemble de nœuds et d'arcs. Les sommets du graphe représentent les tâches composant les travaux à réaliser, auxquels s'ajoutent deux sommets fictifs appelés « source » et « puits » correspondant respectivement au début et à la fin des travaux. Ainsi les arcs peuvent être de deux types :

- Arcs conjonctifs connectent deux tâches consécutives d'un même travail (contraintes de précédence) .Ces arcs sont pondérés par la durée opératoire de la tâche, exceptés les arcs de la source qui sont pondérés par 0.
- Arcs disjonctifs connectent deux tâches appartenant à des travaux différents qui utilisent la même machine (contraintes de ressources).

I.8 Classification des problèmes d'ordonnancement

Étant donné la diversité des problèmes d'ordonnancement, un formalisme permettant de distinguer ces différents problèmes entre eux et les classifier est utilisé, ce formalisme comporte trois champs α / β / γ. Cette notion introduite par Graham et al. Est couramment utilisée pour distinguer les différentes entités d'un problème d'ordonnancement, ces champs se caractérisent comme suit [5] :

I.8.1 Le champ α (ressources) :

Le champ α décrit les machines utilisées, le type de machines, leur nombre, et le type d'atelier, il est composé de plusieurs sous champs et est distingué par :

$\alpha = \alpha_1, \alpha_2$ tel que :

- $\alpha_1 \in \{1, P, Q, R, O, F, J\}$

$\alpha_1 = 1$: une seule machine.

$\alpha_1 = P$: machines parallèles identique.

$\alpha_1 = Q$: machines parallèles uniformes.

α_1 = R : machines parallèles différentes.

α_1 = O : machines spécialisées (Open shop).

α_1 = F : machines spécialisées (Flow shop).

α_1 = J : machines spécialisées (job shop).

- $\alpha_2 \in \{\emptyset, K\}$: où $\alpha_2 = \emptyset$: Le nombre de machines est variable.

α_2 = K : Le nombre de machines est égal à K avec $K \in \mathbb{N}$.

I.8.2 Le champ β (contraintes) :

Le champ β décrit les taches, les contraintes liées aux tâches, il est composé de cinq sous champs, $\beta = \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ tel que :

- $\beta_1 \in \{\emptyset, pmtn\}$: indique la possibilité d'interruption des tâches.

$\beta_1 = \emptyset$: La préemption des tâches n'est pas autorisée.

β_1 = pmtn : La préemption des tâches autorisée.

- $\beta_2 \in \{\emptyset,prec, tree, chain,...\}$: indique le type de contraintes de précédence.

$\beta_2 = \emptyset$: pas de contraintes de précédence (les tâches sont indépendantes) ;

β_2 = prec : les contraintes de précédence sont quelconques ;

β_2 = tree : les contrainte de précédence sont données sous forme d'un arbre.

- $\beta_3 \in \{\emptyset, ri\}$: décrit les dates de disponibilité.

$\beta_3 = \emptyset$: toutes les dates de disponibilité sont nulles ;

$\beta_3 = r_i$: ils existent des dates de disponibilité différentes.

- $\beta_4 \in \{\emptyset, Pi = P ; P' \leq P_i \leq P''\}$: indique les temps de traitement.

$\beta_4 = \emptyset$: les temps de traitement des tâches sont différents.

$\beta_4 = C_i = C$: les temps de traitement des tâches sont tous égaux à C.

$\beta_4 = C' \leq C_i \leq C''\}$: les temps de traitement des tâches sont compris entre C' et C''.

- $\beta_5 \in \{\emptyset, d_i\}$

$\beta_5 = \emptyset$: soit on n'a pas de dates échues, soit on a des tâches échues dans le cas où le critère d'optimisation est en fonction de ces dernières.

$\beta_5 = d_i$: il existe des dates échues.

I.8.3 Le champ γ (ressource) :

Le champ γ décrit le critère qu'on veut optimiser.

- $\gamma \in \{C_{max}, \sum C_i\}$.

Où :

C_{max} : La durée totale d'exécution des tâches.

$\sum C_i$: La date de fin moyenne des tâches.

I.9 Complexité des problèmes d'ordonnancement

La théorie de la complexité [6] s'intéresse à l'étude formelle de la difficulté théorique intrinsèque des problèmes en informatique ou d'un problème par rapport à un autre et de l'analyse de la complexité des programmes et des algorithmes. Concrètement, on cherche à savoir si le problème étudié est plutôt « facile » ou plutôt « difficile » à résoudre en se basant sur une estimation (théorique) des temps de calcul et des besoins en mémoire informatique.

Les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoire [6]. Pour résoudre un problème d'ordonnancement, nous devons toujours chercher à établir sa complexité, car cela détermine la nature de l'algorithme à mettre en œuvre. Si le problème étudié appartient à la classe P, nous savons d'avance qu'un algorithme polynomial existe pour le résoudre. Dans le cas contraire, si le problème est NP-difficile, deux approches sont possibles. La première est de proposer un algorithme approché, donc une heuristique, qui calcule en temps polynomial une solution s'approchant au mieux de la solution optimale. Une alternative est de proposer un algorithme qui calcule la solution optimale du problème, mais pour lequel la complexité dans le pire des cas est exponentielle. Dans ce cas, le défi est de concevoir un algorithme qui peut résoudre en temps acceptable les problèmes de la plus grande taille possible.

Chapitre II :

Modélisation de problème job-shop

II.1 Présentation d'un problème de type job-shop

Un job-shop [12] est constitué d'un ensemble fini, noté M, de m machines différentes qui doit exécuter un ensemble J, lui aussi fini, de n tâches. Chaque tâche, notée j_i, avec

$i \in \{1,..., n\}$, est constituée d'une gamme opératoire, aussi appelée suite linéaire, de n_i opérations. Cette séquence ne dépend que de la tâche et peut donc varier d'une tâche à l'autre. Elle correspond à l'ordre de passage prédéterminé sur chaque machine. Ainsi, une opération O_{ij} (i machine, j tâche) de temps d'exécution p_{ij} sera nommée première opération si celle-ci est la première de la séquence formant la tâche. Au total, il y a donc$\sum_1^n ni$, opérations à exécuter dans l'atelier. Chaque machine, notée M_k avec $k \in \{1,..., m\}$, est spécialisée et ne peut effectuer qu'un seul type d'opération.

Les contraintes intrinsèques au job-shop sont les suivantes :

- les machines sont indépendantes les unes des autres, elles n'utilisent pas d'outils en commun par exemple.
- une machine est disponible pendant tout l'ordonnancement, même les pannes éventuelles sont prises en compte.
- une machine ne peut effectuer qu'une seule opération à la fois.
- la préemption n'est pas permise.
- deux opérations d'une même tâche ne peuvent être exécutées simultanément.
- les tâches sont indépendantes les unes des autres.

Rappelons qu'il est possible qu'une tâche puisse visiter plusieurs fois la même machine [12]. Ce phénomène est appelé « recirculation » ou gamme bouclante. Toutefois, il est fréquent de trouver des formulations plus restrictives [12]. Par exemple, si les gammes bouclantes ne sont pas permises, chaque tâche est alors composée de m opérations et chaque machine doit effectuer n tâches. Le nombre total d'opérations est alors $m*n$. De plus, si $m=n$ alors le problème est dit *c*arré.

Une autre restriction possible est que la $j^{ème}$ opération doit être exécutée par la $j^{ème}$ machine, ce qui revient à traiter un problème de flow-shop.

L'ordre de passage sur les machines étant fixé, le but est de trouver l'ordonnancement de toutes les opérations sur chaque machine. La plupart des recherches effectuées sur ce type d'atelier ont comme fonction objectif le makespan [12].

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 les 3 pièces (3 jobs).

Le tableau (2.1) suivant représente les gammes opératoires des jobs et les dures des opérations :

Job	Opération 1	Opération 2	Opération 3
Job 1	(M3, 1)	(M1, 3)	(M2, 7)
Job 2	(M3, 5)	(M4, 4)	(M1, 1)
Job 3	(M2, 5)	(M1, 5)	(M3, 3)

Tableau **II**.1 : Représentation des gammes opératoires des jobs [12]

Une solution de ce problème consiste à définir un ordre des opérations sur les machines. Le tableau 2.2 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			Op.2 du job 2

Tableau **II**.2 : Représentation de l'ordre de passage des différents jobs [12]

La figure suivante représente l'ordre des différents jobs sur chaque machine

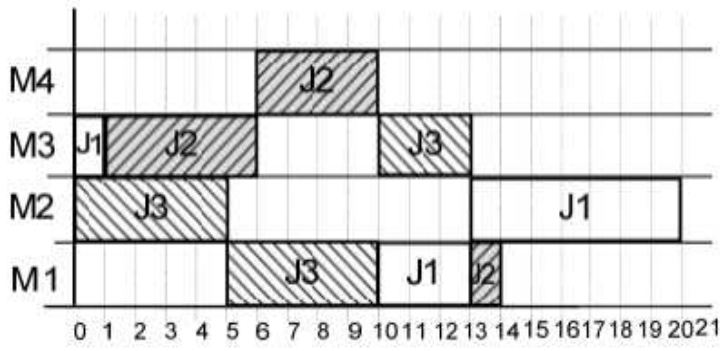


Figure II.1 : Une représentation sous forme de diagramme de Gantt d'un problème du job-shop [12].

II.2 Formulation du problème

II.2.1 Formulation linéaire :

Plusieurs formulations linéaires existent pour le job-shop et certaines d'entre elle se basent sur la formulation de Manne [13].

Dans Pham [13], une évaluation des formulations linéaires du job-shop est proposée. Pour un problème de Job-shop de n jobs et m machines, on considère les notations suivantes :

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

I : L'ensemble de toutes les opérations ;

M : L'ensemble de machines $M = \{1, 2,...., m\}$;

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 $\tau \in I$;

II.2.2 Formulation mathématiques :

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 [13] :

$$x_j - x_i \geq P_i \qquad \forall (i,j) \in A_k,\ \forall k \in \tilde{J} \qquad (2.1)$$

$$x_j + (1 - y_{ij}) - x_i \geq P_i \qquad \forall (i,j) \in B \qquad (2.2)$$

$$x_i + H y_{ij} - x_j \geq P_j \qquad \forall (i,j) \in B \qquad (2.3)$$

$$x_\tau - x_i - P_i \geq 0 \qquad \forall i \in I \qquad (2.4)$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in B \qquad (2.5)$$

$$x_i \geq 0 \qquad \forall i \in I \qquad (2.6)$$

$$x_\tau \geq 0 \qquad (2.7)$$

- Les contraintes (2.1) assurent qu'aucune opération ne peut commencer avant la fin d'exécution de l'opération qui la précède.
- Les contraintes (2.2) et (2.3) 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 (2.5).
- Les contraintes (2.4) fixent la date de début de l'opération τ, ceci est assuré par le fait que x_τ est supérieur à toutes les dates de début plus le temps de traitement $x_i + p_i$ de chaque opération par $i \in I$.
- Les contraintes (2.6) et (2.7) imposent que toutes les dates de débuts des opérations sont positives ou nulles.

II.3 Modélisation d'un problème de job-shop

Le job-shop peut être modélisé de différentes façons, mais les deux plus communes sont la modélisation programmation linéaire mixte et celle par graphe disjonctif. Notons que d'autres modélisations du job shop ont été proposées. Pour plus de détails, le lecteur peut se référer à [13].

II.3.1 Modélisation linéaire mixte :

La modélisation linéaire mixte a été l'une des premières méthodes utilisées dans la littérature [13]. Elle consiste à :

Minimiser (t_n)

Sous les contraintes suivantes :

$$t_i - t_j \geq p_i \qquad (i,j) \in A \qquad (2.8)$$

$$t_i \geq 0 \qquad i \in N \qquad (2.9)$$

$$(t_j - t_i \geq p_j) \ ou \ (t_i - t_j \geq p_i) \qquad (i,j) \in E_k, k \in M \qquad (2.10)$$

Rappelons que :

- M est l'ensemble des machines.
- N est l'ensemble des opérations (incluant deux opérations supplémentaires fictives, les opérations de départ et de fin de l'atelier)

- A l'ensemble de paires (i,j) d'opérations marquant les relations de précédence de la séquence des machines pour une tâche.
- E_k est l'ensemble des paires d'opérations devant être exécuté sur la machine k.
- t_i est la date de départ de l'opération.
- La contrainte (2.8) assure que la séquence des opérations de chaque tâche correspond à l'ordre prédéterminé.
- La contrainte (2.9) assure la terminaison de toutes les tâches. Déterminer un ordonnancement respectant ces contraintes est alors une solution réalisable du problème de job-shop.
- La contrainte (2.10) garantit que les machines n'exécutent qu'une tâche à la fois.

II.3.2 Modélisation par graphe disjonctif :

Bien que, le diagramme de Gantt soit la méthode traditionnelle pour visualiser un ordonnancement. Blazewicz et al. [13] ont montré que la modélisation par graphe disjonctif est maintenant prévalente. Cet outil de modélisation a été présenté en 1964 pour la première fois par Roy et Sussmann [13].

L'atelier est représenté sous la forme d'un graphe G = (S, C, D) où :

S : est l'ensemble des nœuds du graphe. Cet ensemble de nœuds représente l'ensemble des opérations de toutes les tâches, aux quelles sont ajoutées deux nœuds fictifs s et p. Ces deux nœuds représentant respectivement le début et la fin de l'ordonnancement.

C : est l'ensemble des arcs appelés arcs conjonctifs qui symbolisent les chemins de toutes les tâches. Ces arcs indiquent donc les contraintes de précédence entre les opérations. Ainsi s'il existe un arc allant de l'opération O_{11} à l'opération O_{12} alors l'opération O_{11} on est le prédécesseur immédiat de O_{12}.

D : est l'ensemble des arcs disjonctifs marquant les contraintes de ressource. Ces arcs sont, au départ, non dirigés, ils relient les opérations devant passer sur la même machine. Le choix d'une direction assure que deux opérations partageant la même ressource ne sont pas exécutées simultanément. Il est à noter que les arcs de D reliant toutes les opérations partageant une même machine forment une clique de disjonctions (ou sous graphe complet) de doubles arcs. Puisqu'il y a m machines, alors l'ensemble d'arcs disjonctifs forment m cliques. Chaque arc du graphe est pondéré avec le temps d'exécution de l'opération origine de l'arc. Les arcs partant du nœud source s sont étiquetés par un temps nul.

Un ordonnancement réalisable correspond à la sélection d'un arc disjonctif dans chaque paire formant les cliques pour aboutir à un graphe acyclique. La sélection des arcs dans une clique doit donc être acyclique. C'est l'arc sélectionné qui donne l'ordre de passage sur la machine. La durée totale d'un ordonnancement est obtenue, après la création du graphe acyclique, en recherchant le plus long chemin entre le nœud initial et le nœud final. Ce chemin est alors appelé chemin critique (Le chemin critique de votre projet est la plus longue séquence de tâches qui doit être accomplie pour que le projet soit terminé à la date d'échéance).

II.4 Complexité des problèmes job-shop

Ils sont en général NP-complets [13], même si l'atelier est simple. En effet, Lenstra et Rinnooy Kan [13] ont montré que les ateliers possédant plus de trois machines, ou un nombre de tâches supérieur ou égal à trois, sont NP-difficiles même si la préemption est permise. De même, pour les problèmes à deux machines, dès qu'il y a recirculation, ils deviennent fortement NP-difficiles. En revanche, si la fonction objective est la minimisation du makespan, alors il existe quelques cas particuliers résolubles en temps polynomiaux :

- un atelier composé de deux machines dont chaque tâche comprend au plus deux opérations, Jackson [13],

- un atelier comprenant deux tâches sur m machines.

- un atelier à deux machines dont les tâches ont des opérations unitaires.

- un atelier à deux machines dont les tâches ont des opérations unitaires avec des dates de disponibilité.

- un atelier à deux machines dont le nombre de tâches est fixe ?. Malheureusement, même de petites modifications à ces ateliers les font basculer dans la classe des problèmes NP-difficiles.

Par exemple, l'ajout d'une troisième machine même pour des ateliers ayant des tâches unitaires devient NP difficile.

Le tableau (2.3) résume les principales complexités en fonction des caractéristiques de l'atelier.

Type d'atelier	Complexité
$J2 \mid ni \leq 2 \mid c_{max}$	*O(nlogn)*
$J2 \mid pij = 1 \mid c_{max}$	*O(nlog(nr))*
$J2 \mid pij = 1; ri \mid c_{max}$	$O(n^2)$
$J2 \mid n = k \mid c_{max}$	$O(r^{2k})$
$J2 \mid pij \in \{1,2\} \mid c_{max}$	*NP-difficile*
$J2 \mid chains ; pij = 1 \mid c_{max}$	*NP-difficile*
$J2 \mid prmtn \mid c_{max}$	*NP-difficile*
$J2 \mid n = 3; prmtn \mid c_{max}$	*NP-difficile*
$J3 \mid pij = 1 \mid c_{max}$	*NP-difficile*
$J2 \mid n = 3 \mid c_{max}$	*NP-difficile*

Tableau II .3 : Récapitulatif des principales complexités d'un job-shop

Chapitre III :

Méthodes de résolution de problème job-shop

III.1 Les méthodes de résolution d'un problème d'ordonnancement

Dans la littérature, on distingue deux types de méthodes de résolution d'un problème d'ordonnancement [10] : les méthodes exactes et les méthodes approchées. Voici un schéma général qui résume les différentes méthodes de résolution du problème :

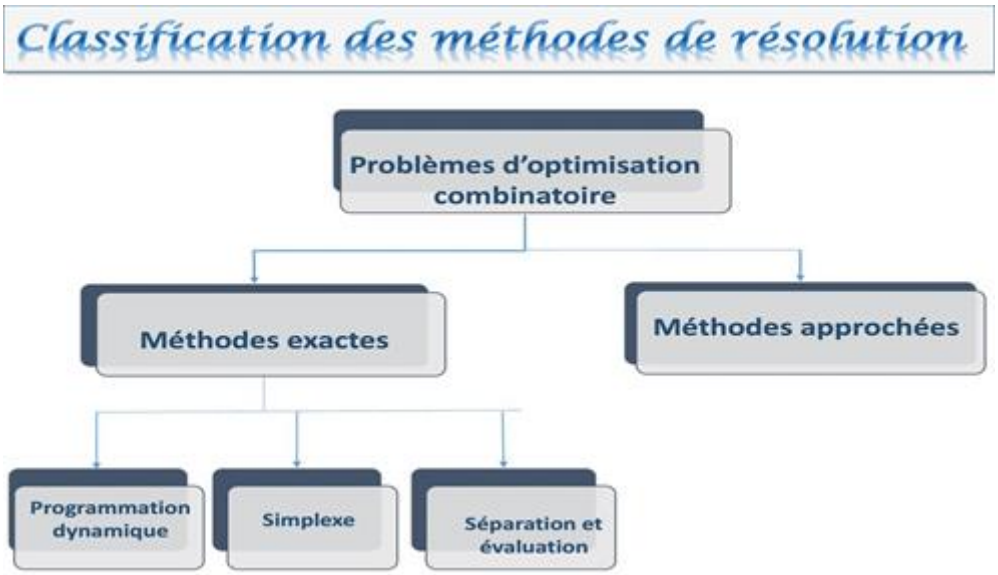


Figure III.1 : Classification des méthodes de résolution

III.1.1 Les algorithmes exacts :

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

Dans ce qui suit, on cite quelques méthodes exactes [10] :

- Procédure par Séparation et Évaluation (en anglais Branch and Bound).
- Programmation dynamique.
- Programmation par contraintes.

III.1.1.1 La méthode de "Branch and Bound" :

La méthode de branch and bound (procédure par évaluation et séparation progressive) consiste à énumérer ces solutions d'une manière intelligente en ce sens que, en utilisant

certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème. Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu, la performance d'une méthode de branch and bound dépend, entre autres, de la qualité de cette fonction (de sa capacité d'exclure des solutions partielles tôt).

Algorithme général de la méthode :

Par convenance, on représente l'exécution de la méthode de branch-and-bound à travers une arborescence. La racine de cette arborescence représente l'ensemble de toutes les solutions du problème considéré. Dans ce qui suit, nous résumons la méthode de branch-and-bound sur des problèmes de minimisation.

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

- Diviser l'espace de recherche en sous espaces,
- Chercher une borne minimale en terme de fonction objectif associée à chaque sous espace de recherche,
- Eliminer les mauvais sous-espaces,

Reproduire les étapes précédentes jusqu'à l'obtention de l'optimum global.

III.1.1.2 Programmation dynamique :

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

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

Le système est constitué de n étapes que l'on résout séquentiellement, le passage d'une étape à une autre se faisant à partir des lois d'évolution du système et d'une décision. Le principe d'optimalité de Bellman est basé sur l'existence d'une équation récursive permettant de décrire

la valeur optimale du critère à une étape en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille k -1 et porte le nombre de sous-ensembles considérés à 2n (où n est le nombre d'éléments considérés dans le problème).

III.1.1.3 La programmation par contraintes :

La programmation par contraintes [10] est un paradigme de programmation apparu dans les années 1970 et 1980 permettant de résoudre des problèmes combinatoires de grande taille 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, 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).

III.1.2 Algorithmes approchés :

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.

Elles sont plus simples et plus rapides à mettre en œuvre quand la qualité de la solution n'est pas trop importante.

Elles sont plus souples dans la résolution des problèmes réels.

En pratique, il arrive souvent que l'on doive prendre en considération de nouvelles contraintes qu'on ne pouvait pas formuler dès le départ. Ceci peut être fatal pour une méthode exacte si les nouvelles contraintes changent les caractéristiques sur lesquelles s'autorisait la méthode.

Elles fournissent des solutions et des bornes qui peuvent être utiles dans la conception de méthodes exactes. Pour les problèmes d'ordonnancement, si on cherche une permutation optimale des tâches, l'espace des solutions admissibles comporte n! Permutations. Un exemple de voisinage d'une solution est défini par toutes les permutations obtenues en échangeant deux tâches consécutives de la permutation.

Les méthodes approchées diffèrent aussi, les unes des autres, par le type de compromis qualité / complexité qu'elles offrent. Nous en distinguons deux classes [10] : les heuristiques et les métaheuristiques.

III.1.2.1 Les heuristiques :

Un raisonnement ou une méthode heuristique [10] est une méthode de résolution de problème qui ne s'appuie pas sur une analyse détaillée ou exhaustive du problème. Elle consiste à fonctionner par approches successives en s'appuyant, par exemple, sur des similitudes avec des problèmes déjà traités afin d'éliminer progressivement les alternatives et ne conserver qu'une série limitée de solutions pour tendre vers celle qui est optimale.

Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable. Parmi ces stratégies, nous distinguons [1] :

- FIFO (First In First Out), où la première tâche arrivée est la première à être ordonnancée,
- SPT (Shortest Processing Time), où la tâche ayant le temps opératoire le plus court est traitée en premier,
- LPT (Longest Processing Time), où la tâche ayant le temps opératoire le plus important est traitée en premier,
- EDD (Earliest Due Date), où la tâche ayant la date due la plus petite est la plus prioritaire.

III.1.2.2 La métaheuristique :

Les métaheuristiques [1] forment un ensemble de méthodes utilisées en recherche opérationnelle et en intelligence artificielle pour résoudre des problèmes d'optimisation réputés difficiles.

Une métaheurisique est un ensemble de concepts qui permettent de construire des méthodes approchées pour plusieurs problèmes différents. En d'autres termes, une métaheuristique peut être vue comme un cadre algorithmique général qui peut être appliqué à différents problèmes avec peu de modifications pour les adapter à un problème spécifique.La figure (III.2) suivante présente quelques méthaheuristiques [1] :

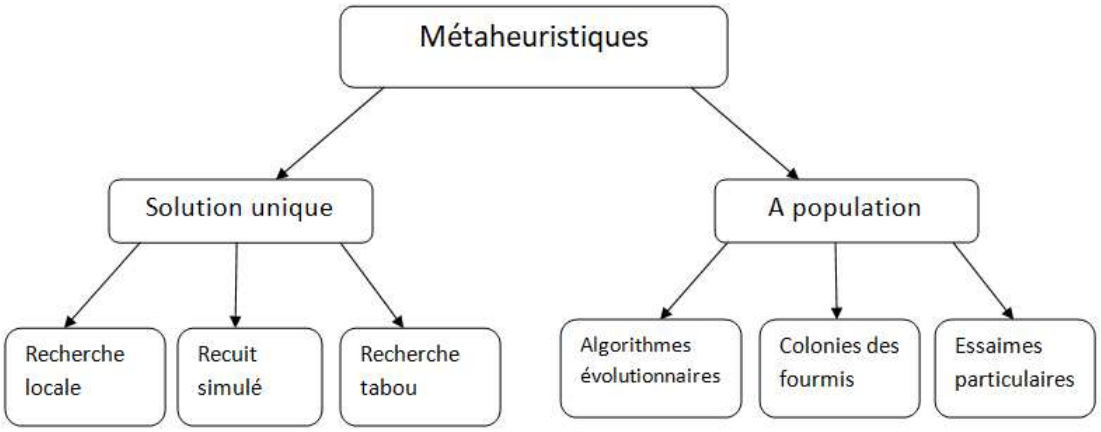


Figure III.2 : Les métaheuristiques

III.2 Algorithmes de résolution d'un problème à une seule machine

Les modèles conçus pour résoudre le problème d'une seule machine sont imposants pour plusieurs raisons. Chaque modèle a été développé dans un objectif bien défini. Il y a ceux qui visent l'optimisation du temps d'achèvement des tâches, ou la minimisation du makespan, d'autres qui visent la minimisation des retards ou du nombre de tâches en retard.

L'environnement manufacturier composé d'une seule machine est un cas spécial de tous les autres environnements. Dans la pratique, les problèmes d'ordonnancement dans des environnements complexes sont souvent décomposés en sous problèmes qui seront traités avec des modèles d'optimisation utilisés pour une seule machine.

Généralement, lorsque les tâches n'arrivent pas en même moment dans le système, une règle de priorité est utilisée pour résoudre un tel problème. [4]

III.2.1 Premier arrivée, premier servi ("PAPS") :

Les tâches (commandes) sont traitées sur la machine suivant leur ordre d'arrivée. L'algorithme stipule que : la séquence optimale est obtenue en classant ces dernières dans l'ordre croissant des dates de disponibilité r_i.

Exemple :

Tâche (i)	1	2	3	4	5
r_i	2	0	1	5	4

Tableau **III**.1 : Temps de disponibilité des tâches

- La séquence des tâches à traiter sur la machine est : 2 3 1 5 4

Lorsque, toutes les tâches sont disponibles au même moment dans le système, alors les règles de priorité suivantes sont utilisées pour résoudre le problème d'ordonnancement.

III.2.2 Temps de traitement le plus court en premier ("TTPC")

Nous traitons en premier la tâche qui a le temps de traitement le plus court. Alors, nous classons les tâches dans l'ordre croissant de leurs temps de traitement p_j : min (p_j).

Exemple :

Tache (j)	1	2	3	4	5
p_j	4	8	1	2	7

Tableau **III**.2 : Le temps de traitement des tâches

- L'ordre croissant des p_j est *:* p_3, p_4, p_1, p_5, p_2.
- La séquence optimale est alors : 3 4 1 5 2.

III.2.3 Temps de traitement le plus long en premier ("ITPL")

La séquence optimale est obtenue en classant les tâches dans l'ordre décroissant des temps de traitement p_j Max (p_j).

Exemple :

Nous considérons les données du tableau précédent et nous appliquons la règle TTPL.

- L'ordre décroissant des p_j est : p_2, p_5, p_1, p_4, p_3.
- La séquence optimale est : 2 5 1 4 3.

III.3 Job-shop à au moins deux machines et l'algorithme de Jonhson

Dans ce qui suit, on décrit l'algorithme de Johnson pour la résolution de problème d'ordonnancement de type job-shop dans deux cas [12] : statique et dynamique.

III.3.1 Le modèle statique :

III.3.1.1 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.

III.3.1.2 Algorithme de Johnson :

L'algorithme de Johnson proposé par S.M Johnson [12] 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 : $O\ (n\ log(n))$.

III.3.1.3 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 de l'exécution 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. [12]

Algorithme à deux machines construit à partir de la règle de Johnson

Présentation d'algorithme de deux machines (m=2) : la règle de Johnson
❖ A_i : Durée du travail i sur la première machine. ❖ B_i : Durée du travail i sur la seconde machine. • Classer les travaux en deux groupes. • Dans le premier groupe G1, mettre tous les travaux tels que $A_i<B_i$ • Dans le second groupe G2, mettre tous les travaux tels que $B_i \leq A_i$ • Classer G1 par A_i croissants • Classer *G2* par B_i décroissants • La solution optimale est constituée de la séquence *G1* suivi de *G2*. ➢ Cette séquence est optimale.

Figure III.3 : Présentation de premier algorithme de deux machines, la règle de Johnson

Exemple :

Soit un problème de job-shop consistant à usiner sur 5 taches soient à exécuter sur 2 machines.

Le tableau (3.3) suivant représenté les durées des jobs :

Tâche (i)	1	2	3	4	5
Tâche t_{iA}	50	150	80	200	30
Tâche t_{iB}	60	50	150	70	200

Tableau **III**.3 : Présentation d'un exemple l'algorithme de deux machines

- Dans le premier groupe *G1*, $A_i < B_i$

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

1	2	3	4	5
5				

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

1	2	3	4	5
5	1			

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

1	2	3	4	5
5	1	3		

- Dans le deuxième groupe *G2*, $B_i \leq A_i$:
- la tache 2 (50< 150) donc mise en première position.

1	2	3	4	5
2				

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

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

- Classer G1 par A_i croissants :

1	2	3	4	5
5	3	1		

- Classer G2 par B_i décroissants :

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

- La solution optimale est constituée de la séquence *G1* suivi de *G2*.

1	2	3	4	5
5	1	3	4	2

  - 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.

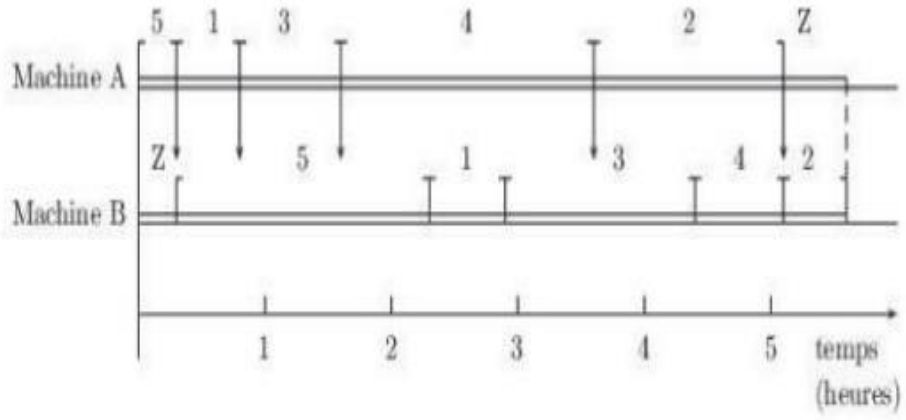


Figure III.4 : Représentation de diagramme de Gantt de 2 machines [12].

III.3.1.4 Ordonnancement d'atelier à m>2

Les environnements "job-shop" complexes sont caractérisés par une multitude de machines (m) qui traitent une variété de différentes tâches (n) qui arrivent d'une façon intermittente devant les machines. Pour résoudre ce type de problème, nous utilisons des approches basées sur des heuristiques qui donnent des résultats satisfaisants. Dans de tels genres de problème, il n'existe pas des algorithmes qui fournissent des cédules de production optimales. [4]

Parmi les heuristiques, on peut citer :

- Palmer : Est souvent utilisée pour la résolution du problème de 3 machines. Elle est basée sur le calcul de l'indice de la pente.
- Heuristique de Campbell, Dudek et Smith : La procédure de cette heuristique est basée sur deux étapes essentiales :

- Étape 1 · La réduction du problème à deux machines en construisant les temps de traitement pour les machines 1 et 2.
- Étape 2 : Application de l'algorithme de Johnson sur ces deux machines.
  - Heuristique de Dannenbring : Cette heuristique combine les avantages de celles de Palmer et de Campbell, Dudek, Smith.

Dans notre travail, on va présenter uniquement l'heuristique de Dannenbring.

III.3.1.5 Heuristique de Dannenbring

L'idée de cette heuristique est basée sur la construction des temps de traitement pour arriver à un problème à deux machines et à partir de là, on utilise l'algorithme de Johnson.[4]

Le temps de traitement sur la machine 1 est :

$$a_j = \sum_{i=1}^{m}(m - i + 1)\ p_{ij} \qquad (3.1)$$

Et celui sur la machine 2 :

$$b_j = \sum_{i=1}^{m} i * p_{ij} \qquad (3.2)$$

Exemple :

Tâche (j)	Temps de traitement		
	Machine 1 P_{1j}	machine 2 p_{2j}	machine 3 p_{3j}
1	1	2	2
2	2	6	6
3	6	8	8
4	3	4	4

Tableau **III**.3 : Temps de traitement de la tâche (j) sur la machine (i)

Nous considérons les mêmes données du tableau (3.4) et nous construisons les temps de traitement pour les machines 1 et 2.

En appliquant les équations (3.1) et (3 .2), nous obtenons les temps de traitement présentés dans le tableau suivant :

Pour m= 3 et n= 4 tâches, les équations (3.1) et (3. 2) s'écrivent comme suit :

$$a_j = 3p_{1j}+2p_{2j}+p_{3j}$$

$$b_j = p_{1j}+2p_{2j}+3p_3$$

Tâche (j)	Problème de m=3 et n=4	
	Construction du temps de traitement	
	a_j =	b_j=
1	9	11
2	24	32
3	42	46
4	21	23

Tableau **III**.4 : Construction des temps de traitement (Dannenbring).

La seconde étape consiste à appliquer l'algorithme de Johnson, pour obtenir la séquence optimale pour un problème de trois machines réduit à un problème de deux machines. Nous obtenons ainsi, la séquence optimale suivante : 1 4 2 3.

Calcul du Makespan (Cmax)

Le makespan [7] (noté C_{max}) représente le temps de fin d'exécution du dernier job dans une séquence. Il est l'un des critères les plus utilisés pour évaluer le coût d'un ordonnancement. En minimisant ce critère, on peut améliorer le rendement et réduire le temps moyen d'inactivité des machines.

Notre objectif est de déterminer le C_{max} minimum. Pour le calculé nous avons utilisé la méthode de diagramme de Gantt qui représente l'ordonnancement du différent job à exécuter dans les machines dont les jobs ont des routages spécifiques, des temps d'exécution et temps de transport vers chaque machine. En respectant l'ordre d'exécution des produits le C_{max} représente le temps de la sortie du dernier produit exécuté dans le système.

Nous avons calculé le C_{max} sur le diagramme de Gantt de l'exemple (Tableau 3.4) précédent, on obtient les résultats suivants :

	Tâche 1		Tâche 4		Tâche 2		Tâche 3	
	t_i	c_i	t_i	c_i	t_i	c_i	t_i	c_i
Machine 1	0	1	1	4	4	8	8	12
Machine 2	1	3	4	8	8	14	14	22
Machine 3	3	5	8	12	12	18	22	30

Tableau III.6 : Résultats de calcul de C_{max}

Les valeurs de C_{max} obtenues pour les trois machines sont données comme suit :

$$C_{max1}=12 \quad , \quad C_{max2}=22 \quad , \quad C_{max3}=3$$

III.3.2 Modèle dynamique :

III.3.2.1 Ordonnancement d'atelier en temps réel :

Le problème d'ordonnancement en temps réel (ou modèle dynamique) [8] 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.

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.

III.3.2.1.1 Approches pour l'ordonnancement en temps réel d'atelier :

Pour la résolution des problèmes d'ordonnancement en temps réel deux types d'approches [8] sont utilisées. La première approche est basée sur un contexte statique et la seconde est basée sur un contexte dynamique.

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 [8], 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.

I.3.2.1.2 Méthode d'insertion de tâches :

Une approche basée sur le contexte statique et dynamique, c'est la méthode d'insertion des tâches [12] 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.

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.

III.3.2.1.3 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.[12]

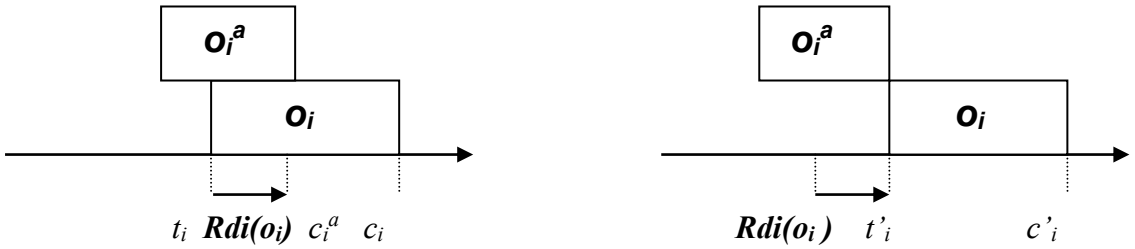
III.3.2.1.4 Procédure de décalage :

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 III.5).

III.3.2.1.5 Création d'une position admissible et détermination du nouvel ordre :

On définit par Rdi(O_i) le temps de décalage (retard) que doit subir la tâche (O_i) pour permettre l'insertion de la tâche (O^a_i). 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). [8]

$$Rdi\ (O_i) = C_i^a - t_i \qquad (3.3)$$



a. Position avant le décalage b. Position après le décalage

Figure III.5 : Exemple d'une opération de décalage effectué pour insérer la tâche

Les nouvelles valeurs pour les durées sont les suivantes :

$$t'_i = t_i + Rdi(o_i) \qquad (3.4)$$

$$c'_i = c_i + Rdi(o_i) \qquad (3.5)$$

Chapitre IV :

Conception

Et

Réalisation

IV.1 Le langage de programmation Matlab

MATLAB (« *matrix laboratory* ») est un langage de script émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. Les utilisateurs de MATLAB (environ 4 millions en 2019) sont de milieux très différents comme l'ingénierie, les sciences et l'économie dans un contexte aussi bien industriel que pour la recherche. [16]

IV.2 Application de l'algorithme Johnson d'un problème job-shop dans un environnement dynamique

Dans notre application, deux stratégies d'insertion sont utilisées [8] :

- insertion à la fin.
- insertion aléatoire.

Procédure Insertion ()

Début

Lire (position NV Tâche)

Lire (Les Durées NV Tâche)

Ajouter à l'ordre (NV Tâche) a L'indice (Position)

Ajouter Les durées de NV Tâche et sa position à L'ensemble des Machines

Fin

Figure IV.1 : L'algorithme de Procédure insertion

L'algorithme de Johnson d'un problème job shop dans environnement statique et dynamique est présenté par l'organigramme suivant :

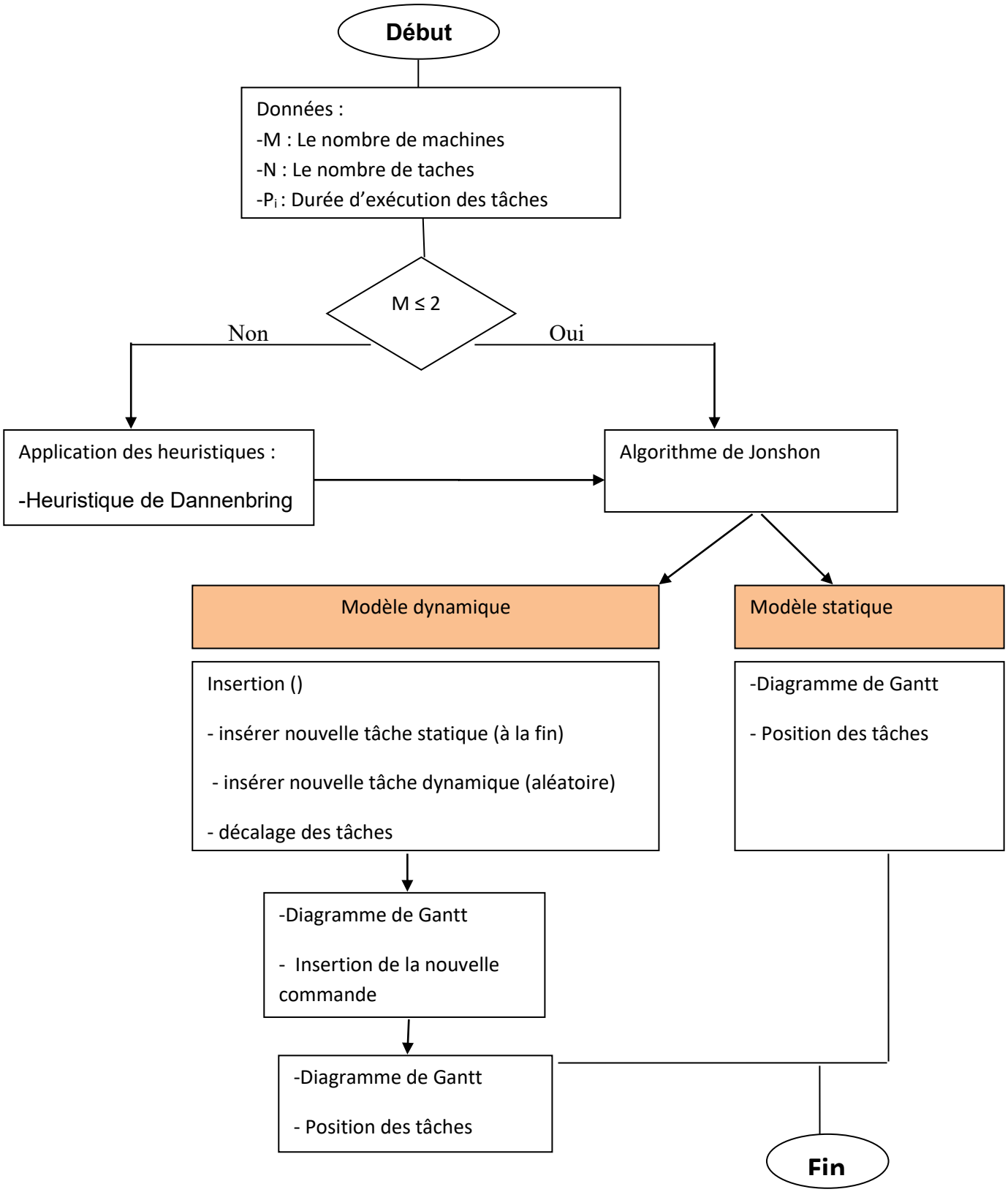


Figure IV.2 : Organigramme de résolution de problème job- shop pour au mois de machines

IV.3 Interface graphique de l'application

IV.3.1 Modèle statique :

Nous avons développé une application permettant de résoudre le problème d'ordonnancement d'atelier en temps réel de type job-shop dynamique.

L'interface de l'application conçue contient une seule fenêtre. Afin d'appliquer les méthodes de résolution décrites dans le chapitre quatre, l'utilisateur doit saisir le nombre des taches, le nombre des machines et la durée des taches de chaque machine pour créer les listes. Le menu principal de l'application est donné par la figure suivante :

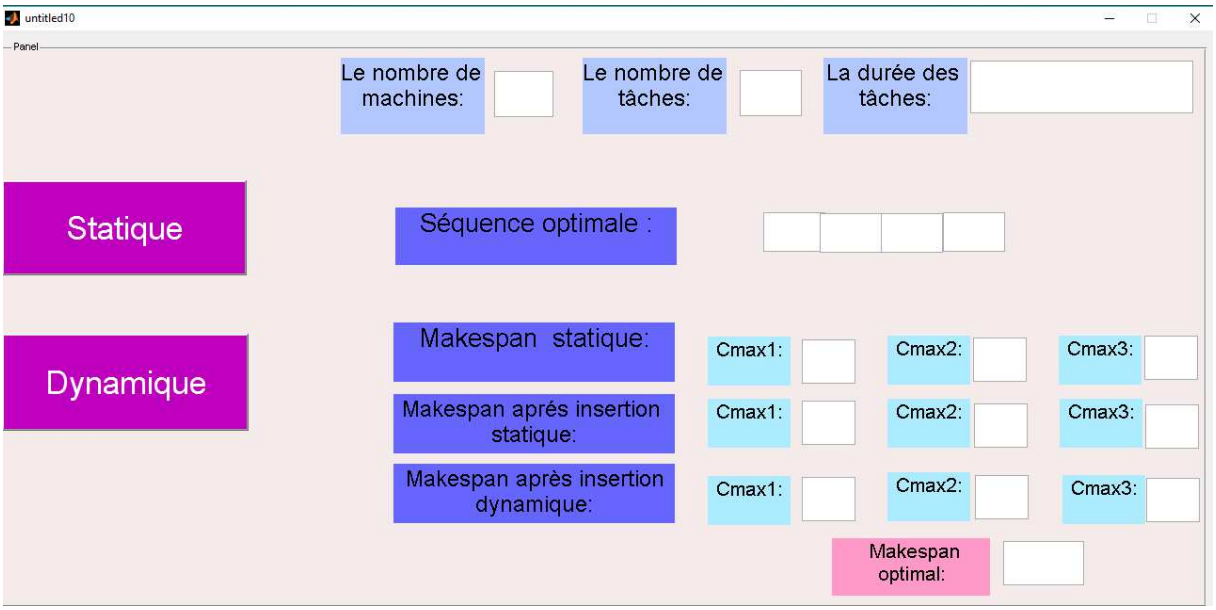


Figure IV.3 : Page de l'interface.

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, n tâches et des durées de chaque machine.

La figure (IV.4) présente les données de l'exemple illustratif de type job shop donné dans le chapitre 3 (Tableau 3.4) composé de 4 tâches et 3 machines.

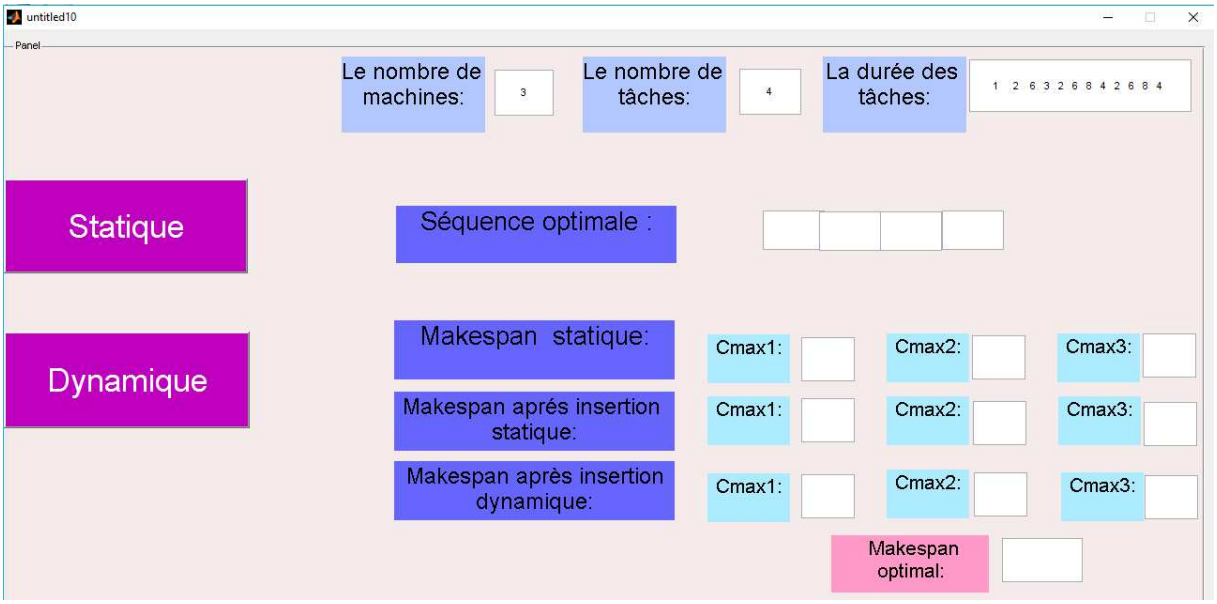


Figure IV.4 : Les données de l'exemple illustratif

Un nombre de tâches et de machines sont en entrée , après, la saisie des durées des tâches passants par chaque machine et exécuter sur le bouton Statique qui affiche l'ordre d'ordonnancement des tâches avec le diagramme de gant et le makespan (C_{max}) de chaque machine.

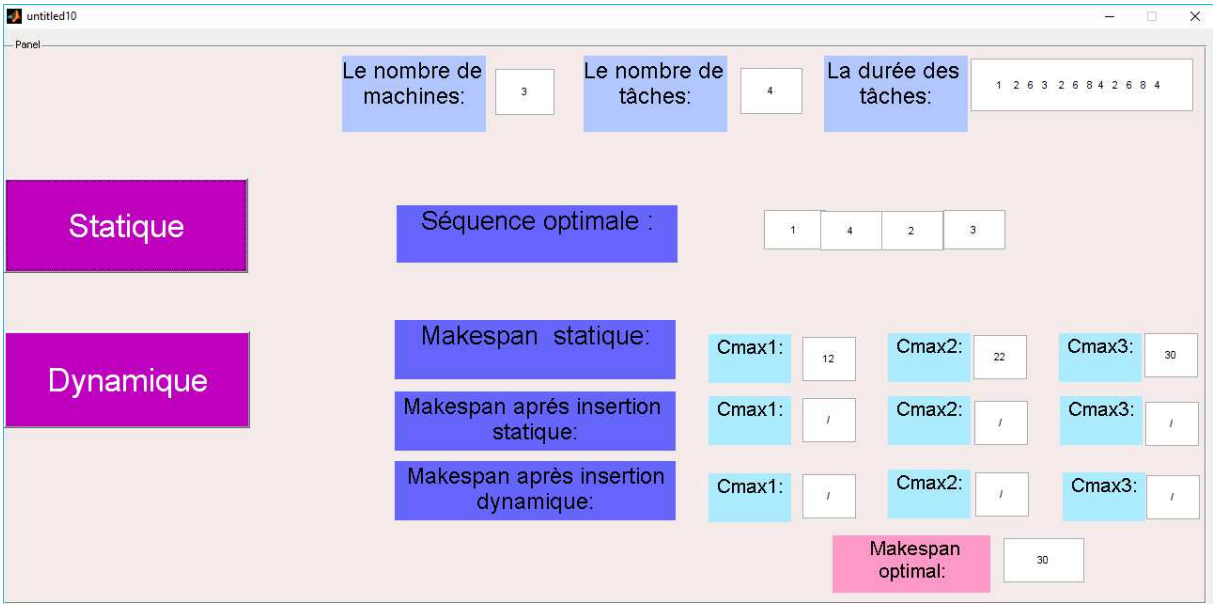


Figure IV.5 : Interface après exécution.

Après l'exécution le résultat présenté le diagramme de Gantt, et donne aussi le Makespan (C_{max}).

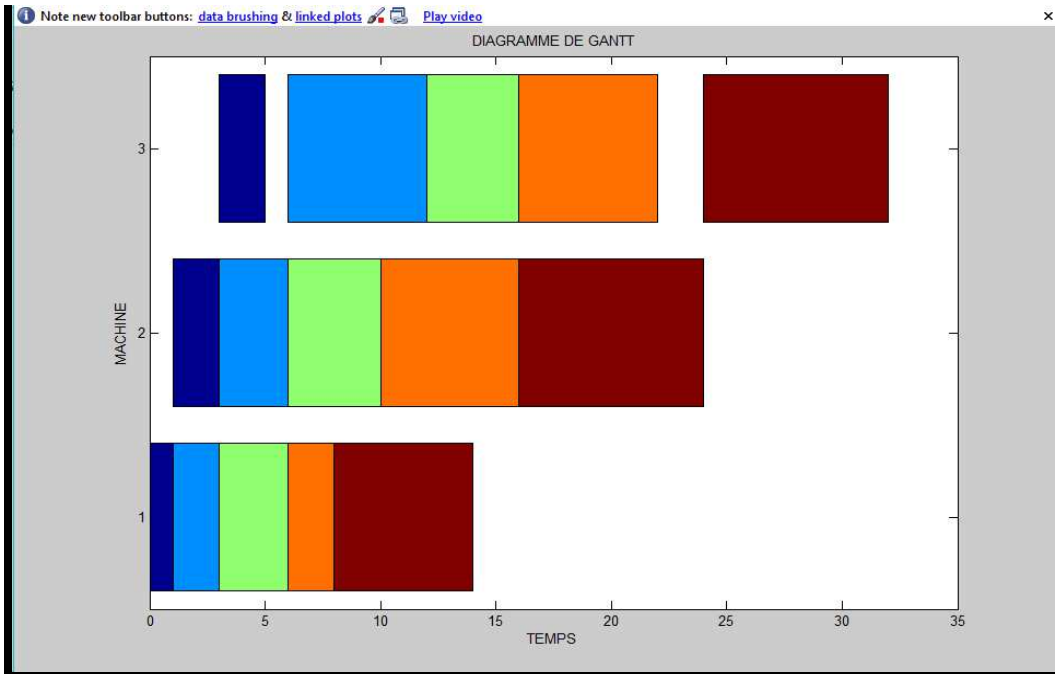


Figure IV.6 : Le diagramme de Gantt représentant la solution optimale de chaque machine.

IV.3.2 Modèle dynamique :

Nous appliquons dans ce qui suit l'ordonnancement dynamique qui contient deux contextes d'insertion : insertion statique et insertion dynamique en temps réel.

Premièrement, on introduit les durées de la nouvelle tache sur l'interface, comme montre la figure suivante :

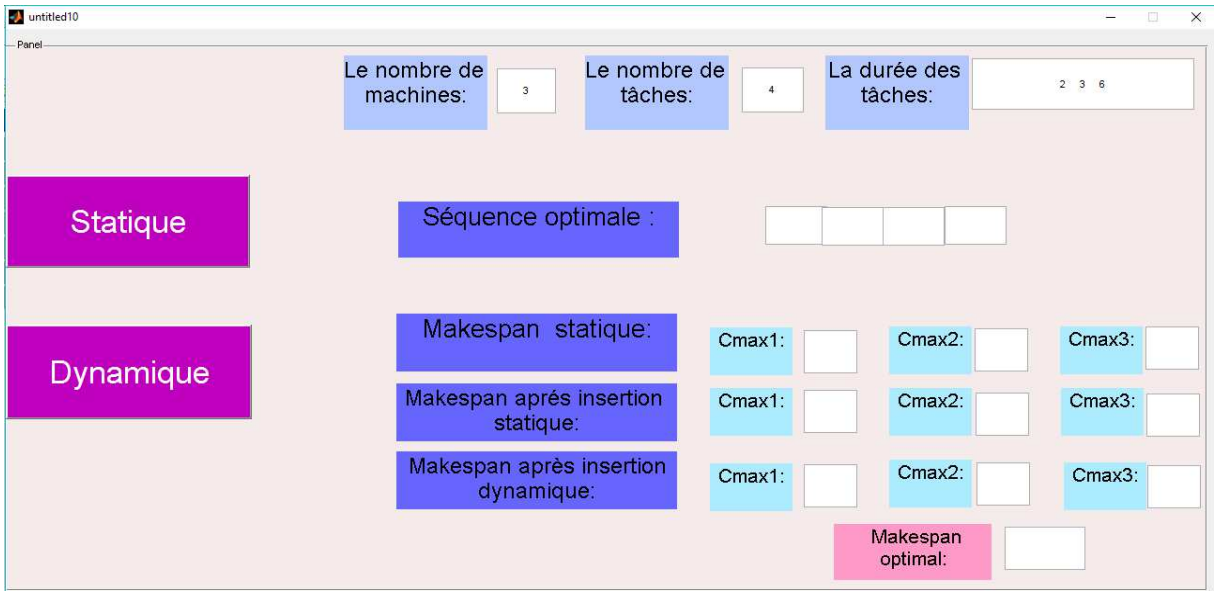


Figure IV.7 : Les durées de la nouvelle tâche

En suit, on choisit l'une des deux types d'insertions :

- Dans l'insertion statique on va ajouter une nouvelle commande qui s'inscrit à la fin de l'ordre d'ordonnancement.
- Dans l'insertion dynamique on va ajouter une nouvelle commande aléatoire dans d'ordre d'ordonnancement.

Enfin, pour exécuter on clique sur le bouton dynamique. Les résultats qui s'affichent sont : le mekespan après insertion statique et après insertion dynamique avec leur diagramme de Gantt ainsi que le makespan (C_{max}) de chaque machine.

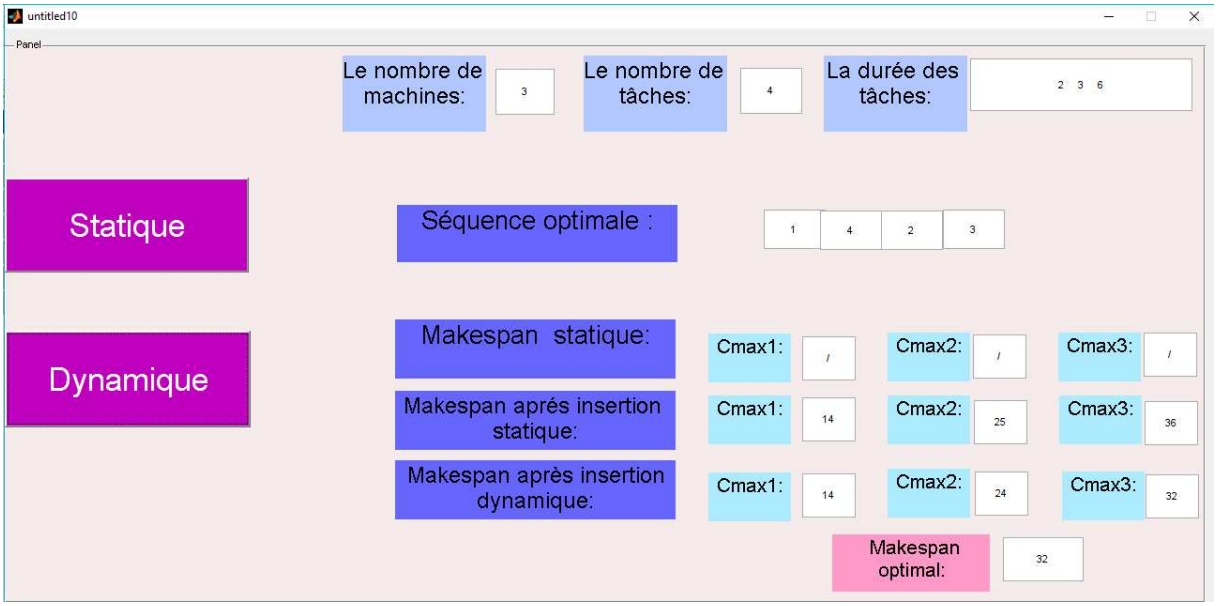


Figure IV.8 : Interface après insertion statique et dynamique.

Les figures suivantes présentent le nouvel ordre et le diagramme de Gantt de l'insertion statique, et dynamique ainsi que le makespan (C_{max}).

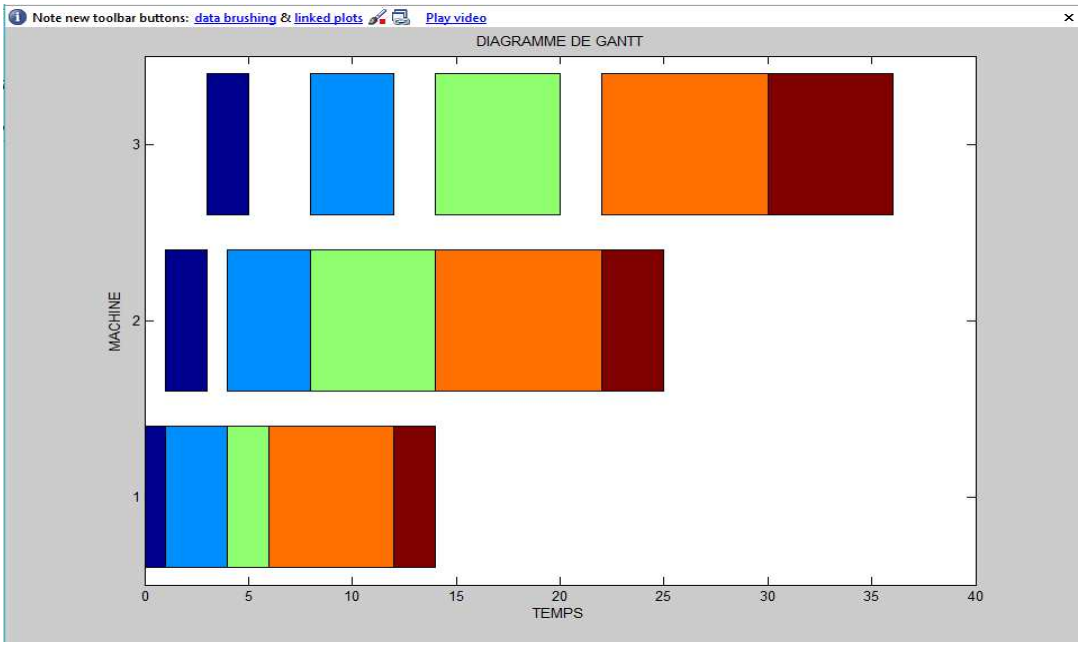


Figure IV.9 : Le diagramme de Gantt après insertion statique

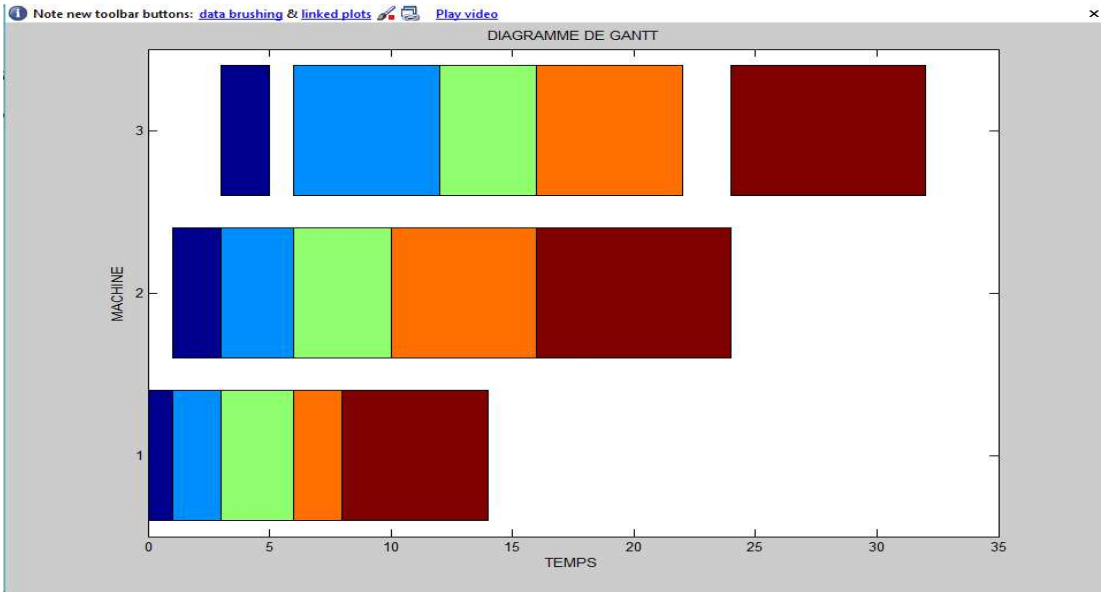


Figure IV.10 : Le diagramme de Gantt après insertion dynamique

Conclusion générale

Dans ce mémoire, nous nous sommes intéressés aux problèmes d'ordonnancements, plus particulièrement aux problèmes d'ordonnancements dans les ateliers. En général, les problèmes d'ordonnancement sont des problèmes difficiles à résoudre, ils sont NP-difficiles.

Nous avons considéré le problème d'ordonnancement de type job-shop à au moins deux machines, dont l'objectif est de minimiser le temps total d'exécution.

Nous avons utilisé l'algorithme de Johnson pour la résolution du problème à deux machines. Cet algorithme est polynomial. Dès que le nombre de machine dépasse deux, nous avons employé l'heuristique de Dannenbring pour pouvoir appliquer la méthode de Johnson.

On a élaboré une application permettant la résolution du problème dans le cas d'ordonnancement statique et le cas d'ordonnancement en temps réel.

Comme perspectives, nous envisageons examiner d'autres méthodes pour la résolution de ce problème, comme les algorithmes génétiques et l'algorithme d'abeille dans la phase de définition des positions d'insertion.il serait aussi intéressant de prendre 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] Adjir H, Les problèmes d'ordonnancement d'atelier M-Machine identique en parallèle, thèse de Master académique en informatique, université de m'silla, Algérie 2018.

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

[3] Belkhous M et Bourahla S , Modélisation d'un processus de fabrication et résolution d'un problème d'ordonnancement de type Flow-Shop. Cas «ENIEM», thèse de Master Professionnel en Mathématiques Appliquées à la Gestion, Université de Mouloud Mammeri, Tizi-Ouzou, Algérie , 2017.

[4] Bensalah Z , Conception et mise en œuvre d'un système de base de connaissances pour l'ordonnancement dynamique de gestion de priorités, basé sur l'approche algorithmique , thèse de Doctorat Université du Québec , Canada, 2000.

[5] Bourzik M , Problèmes d'Ordonnancement d'Atelier, thèse de Licence Sciences et Techniques, Université Sidi Mohamed ben Abdallah, Maroc , 2016.

[6] Dupont D et Daniel R, «Techniques opérationnelles d'ordonnancement De Edmond Maurel».

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

[8] Hadri A , l'ordonnancement par insertion en temps réel de la production dans un atelier flexible , thèse de Magister spécialité génie industriel, Université Hadj Lakhdhar Batna , Algérie , 2012.

[9] Kaabi-Harrath J , contribution a l'ordonnancement des activités de maintenance dans les systèmes de production, l'université de Franche- comté, thèse de Doctorat de l'université de Franche , France 2004.

[10] Laribi I, Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables, Thèse de Doctorat, université de Tlemcen, Algérie, 2018.

[11] Meguireche S, Ordonnancement des systèmes de production flexible, thèse de Master, Université Mohamed Boudiaf - M'sila , Algérie , 2018.

[12] Ramla N , Un problème d'ordonnancement de type Job Shop dans un environnement dynamique, thèse de MASTER, Université Mohamed Boudiaf - M'sila, Algérie, 2018.

[13] Ruhlmann C, Etude du problème de job shop avec un convoyeur, thèse de Doctorat Université du Québec, 2007.

[14] www.etudier.com/dissertations/Recherche-Opérationnelle.

[15] www.visiativ-solutions.fr/ /ordonnancement-de-la-production.

[16] www.wikipedia.org/wiki/MATLAB.

[17] www.wikipedia.org/wiki/Ordonnancement.