REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur 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 : HADJI Asma

TEBBAKH Amira

SUJET

Des métaheuristiques pour un problème d'ordonnancement de type Job Shop

Soutenu publiquement : juin 2022 devant le jury composé de :

Pr. GASMI Abdelkader	**Université de M'sila**	**Président**
Dr. MOUHOUB Nasser Eddine	**Université de M'sila**	**Encadreur**
Mr. BOUDAA Abdelghani	**Université de M'sila**	**Examinateur**

Promotion 2021/2022

Remerciements

Nous remercions en particulier **Mr. MOUHOUB NASSER EDDINE**, pour l'honneur qu'il nous a fait de bien vouloir nous encadrer, et pour les précieux conseils donnés et ses prestations de documentations nécessaires lors de la réalisation de ce travail.

Nous remercions aussi tous les professeurs qui nous ont aidé durant les cinq années d'étude à l'université Mohamed BOUDIAF M'sila.

Nous remercions également **Mr. Hemmak Allaoua** et **Mr. Bounif Mohammed** qui nous ont beaucoup aidés.

On adresse nos remerciements aux membres de jury pour avoir accepté de nous prêter de leur attention et évaluer notre travail.

Les derniers remerciements vont à notre famille en particulier nos parents pour leurs sacrifices et leurs encouragements.

Sommaire

Chapitre 3 : Les métaheuristiques

Chapitre 4 : Conception et implémentation du problème

Listes des figures

Liste des tableaux

Introduction générale

L'ordonnancement consiste à organiser la réalisation des tâches (jobs, travaux) en temps voulu compte tenu des contraintes de temps et de ressources, pour optimiser un ou plusieurs objectifs. Une tâche est donc une entité de base (action ou ensemble d'actions) positionnée dans le temps par date de début et date de fin, sa réalisation se caractérise par la durée positive de la ressource. Cette dernière est un moyen technique ou humain destiné à l'exécution de tâches et en quantité limitée. L'évaluation des solutions de planification se fait par rapport à un ou plusieurs objectifs de Performance.

Parmi les problèmes d'ordonnancement les plus difficiles et les plus étudiés, nous avons le problème de planification d'atelier de type Job Shop. Considéré comme un modèle d'ordonnancement intéressant, le problème Job Shop est très représentatif. Ce problème correspond en fait à la modélisation d'une cellule de production utiliser plusieurs méthodes dans un ordre différent produit. Le but est de programmer la réalisation du produit de manière à optimiser la production, en respectant un certain nombre de machines utilisées effectuer toutes les opérations de base impliquées dans la fabrication de chaque produit.

Dans la plupart des cas, il n'est pas possible de résoudre ce problème de manière optimale en raison de sa nature hautement combinatoire. Les méthodes exactes nécessitent un effort de calcul qui croît de manière exponentielle avec la taille du problème. Ainsi, la méthode solution à ce problème dans un délai raisonnable a été proposée. Parmi ces méthodes, celles dites « méta-heuristiques » ont vu le jour, dont trois ont prouvé leur efficacité dans de nombreuses applications : Les Algorithmes Génétiques appartiennent aux méthodes évolutionnaires ; la recherche taboue et le recuit simulé font partie de la Recherche locale.

Le but de ce travail est de résoudre des problèmes d'ordonnancement de type job shop en appliquant des métaheuristiques. Ce travail porte sur des questions simples du Job Shop représentant le contexte général de l'industrie. Nous espérons à travers ce travail que la méthode de base peut obtenir des résultats satisfaisants et ne nécessite pas de conditions élevées de conception et mettre en œuvre, en temps de résolution et en espace mémoire, plutôt que de trouver une approche plus sophistiquée. Nous concentrons nos efforts sur la mise en œuvre informatique des formes simples (standard) de ces méthodes, chacune avec plusieurs alternatives pour obtenir des résultats démontrant son efficacité.

Ce mémoire comporte quatre chapitres dont nous vous présentons une brève description comme suite :

Dans le premier chapitre, nous allons étudier l'ordonnancement des activités de production en général, les définitions de l'ordonnancement et de ses éléments, les notations des problèmes d'ordonnancement et classification des ateliers (Job Shop, Flow Shop, Open Shop).

Dans le deuxième chapitre, nous allons détailler le problème de job shop : la modélisation graphique, la complexité et les méthodes pour résoudre ce problème.

Le troisième chapitre, nous allons discuter sur les métaheuristiques : l'algorithme génétique, la recherche tabou et le recuit simulé, ainsi que leurs applications dans le problème de job shop.

Le dernier chapitre, contient la conception et l'implémentation du problème e job shop avec les résultats de quelques exemples pour les discuter et les analyser afin d'avoir certaines conclusions.

Chapitre I L'ordonnancement des activités de production

1. Introduction

Dans ce chapitre introductif, nous nous intéressons aux problèmes d'ordonnancement d'une manière générale afin de situer la problématique de notre travail. Il s'agit de présenter les concepts et les bases fondamentales du problème d'ordonnancement. Dans un premier temps, il est nécessaire de replacer l'ordonnancement dans le cadre général des systèmes de production. Nous rappelons alors, dans la deuxième partie, des notions de base concernant la production et la gestion de production, en mettant l'accent sur le rôle de l'ordonnancement au sein de ces systèmes. La troisième partie vise à présenter le problème de l'ordonnancement en précisant sa définition et les différents éléments qui le déterminent. La quatrième partie du chapitre présente les critères d'optimisation comme élément de grande importance. Un formalisme de classification des problèmes d'ordonnancement est évoqué dans la cinquième partie. La dernière partie est consacrée à la description des différentes organisations d'ateliers qui déterminent des problèmes d'ordonnancement spécifiques.

2. La production

La production est une activité économique qui exploite des ressources en travail et en capital, appelées facteurs de production, dans le but de produire des biens ou des services à partir de consommations intermédiaires (biens ou services achetés à d'autres entreprises puis transformés). [A]

La production est le processus qui conduit à la création de produits par l'utilisation et la transformation des ressources [IV]. Le processus de production est alors constitué d'un ensemble d'opérations qui sont les activités qui conduisent à la création de biens et de services. [16]

Le système productif est l'ensemble des ressources qui réalisent une activité productive. C'est un ensemble de moyens divers : humains, matériels, informationnels et autres, constituant un tout, dont l'objectif est la réalisation de biens ou de services.

Les systèmes de production industrielle sont devenus très divers et complexes. En fait, ils peuvent être décomposés en plusieurs sous-systèmes qui s'intègrent les uns aux autres pour assurer la pérennité et la compétitivité de l'entreprise.[4]

3. La gestion de production

La gestion de production est un ensemble d'activités impliquant la conception, la planification, l'ordonnancement, l'enregistrement et la traçabilité des activités de production de ressources (matérielles, financières ou humaines), le contrôle des activités de production d'une entreprise.

Du fait du développement des activités de services dans les économies avancées (ainsi, le secteur tertiaire en France représentait 75 % de l'emploi civil et 76 % de la valeur ajoutée en 2004), la gestion de la production s'est étendue aux services et n'est plus uniquement industrielle. [C]

L'objectif est d'améliorer en permanence la gestion des flux et des stocks dans la chaîne de travail qui commence chez le fournisseur et se termine chez le client intermédiaire ou final.

Mais dans les entreprises de services, la production ne peut pas être stockée par définition en raison de son caractère immatériel. En outre, d'autres caractéristiques des services, telles que l'indissociabilité, que la consommation et la production se produisent nécessairement simultanément, et donc la forte implication des clients dans la production de services, imposent une forme spécifique de gestion de la production de services aux entreprises du secteur tertiaire.

Toutes ces activités doivent être réalisées conformément aux procédures établies par l'entreprise (implicitement ou explicitement), en tenant compte de la qualité de ses produits ou services, ainsi que de la sécurité de ses employés ou de l'environnement.

3.1. Décomposition du système de production

Un système de production peut être divisé en trois sous-systèmes, système physique de production, système décisionnel et système d'information.

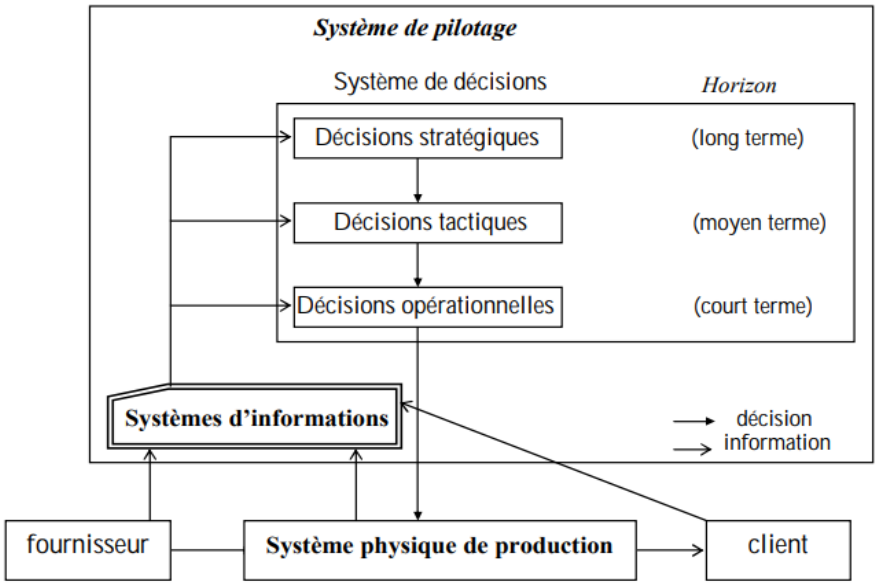


Figure 1.1 : les sous-systèmes constituant le système de production [9][20]

•Système de production physique : conversion de matières premières ou de composants en produits finis. Il se compose de ressources humaines et matérielles. [III]

• Système de décision : contrôle le système de production physique. Il coordonne et organise ses activités en prenant des décisions sur la base des données transmises par les systèmes d'information. [III]

• Systèmes d'information : Interventions à plusieurs niveaux : à l'interface entre les systèmes décisionnels et de production, au sein des systèmes décisionnels pour gérer les

informations utilisées dans le processus décisionnel et au sein des systèmes physiques de production. Son rôle est de collecter, stocker et transmettre différents types d'informations.

3.2. Le rôle de la gestion de production

Utilisation optimisée des ressources : la gestion de la production signifie que la main-d'œuvre, l'équipement et les ressources sont optimisés dans le travail de production. Cela réduit les niveaux de déchets et crée un environnement positif et équilibré pour les employés.

La gestion de la production est "un ensemble de processus qui permettent la fabrication de produits sur la base d'un ensemble de données et de prévisions". [27]

F. Blondel [4] définit la gestion de la production comme « une fonction qui permet aux opérations de production de se dérouler dans le respect des conditions de qualité, de délai et de coût engendrées par les objectifs de l'entreprise ».

En pratique, la gestion de la production implique un ensemble de problèmes liés à la production tels que la gestion des données, la planification, le contrôle de la production (suivi), la gestion des stocks, les prévisions, l'ordonnancement, etc.

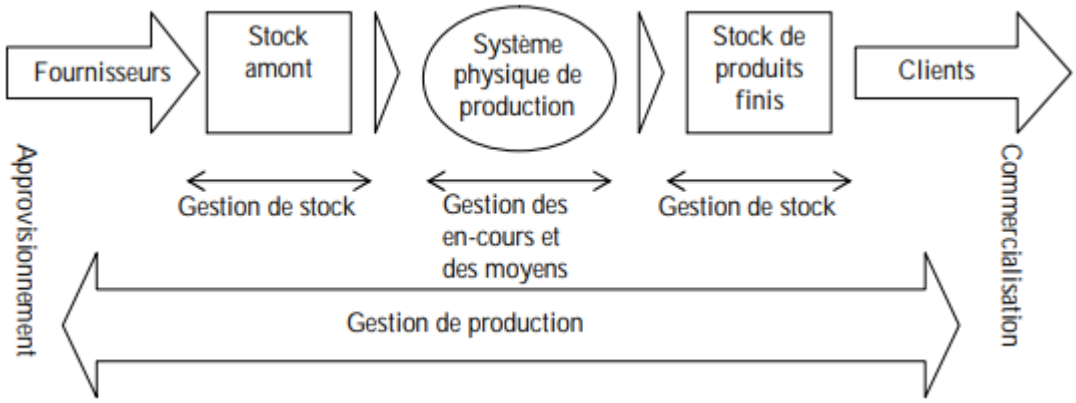


Figure 1.2 : Objectifs de la gestion de production.[16]

3.3. Organisation hiérarchique de la gestion de production

La gestion de la production est généralement divisée en trois niveaux : stratégique, tactique et opérationnel.

• Niveau stratégique : C'est la formulation de la politique à long terme de l'entreprise (plus de deux ans). Il se concentre sur la gestion durable des ressources, leur permettant d'assurer la pérennité de l'entreprise.

• Niveau tactique : Ce sont des décisions à moyen terme. Ils font le lien entre le niveau stratégique et le niveau opérationnel. L'objectif est de produire au moindre coût pour répondre à la demande prévisible dans le cadre fixé par le plan stratégique de l'entreprise.

• Niveau opérationnel : Il s'agit de décisions à court et très court terme. C'est une gestion au quotidien qui répond aux besoins du quotidien tout en respectant les décisions tactiques.

3.4. Le rôle de l'ordonnancement en gestion de production

Le modèle général de gestion de la production divise la prise de décision en trois niveaux : stratégique, tactique (planification à long terme et à moyen terme) et opérationnel (gestion et suivi quotidien des matériaux et du flux de travail). Cette hiérarchie se traduit par la consolidation des domaines de responsabilité (managériale, exécutive, agence, etc.) et des horizons temporels (long terme, moyen terme, court terme). [19]

Dans ce schéma pratique et classique, il est impossible de résoudre le problème d'ordonnancement de manière globale (à un niveau supérieur), d'autant plus que les aléas tant endogènes (pannes machines, grèves, etc.) qu'exogènes (priorités imprévisibles). Les occurrences nécessitent des occurrences fréquentes recalcul du calendrier. Par conséquent, les problèmes d'ordonnancement sont traités à un niveau inférieur. [III]

Le lieu de répartition diffère entre les niveaux tactique et opérationnel. Il gère la mise en œuvre des décisions des niveaux supérieurs.

Il couvre un ensemble d'opérations qui traduisent les décisions de fabrication définies par le programme maître de production en instructions d'exécution détaillées destinées à contrôler et gérer à court terme l'activité du poste de travail.

En sortie de la fonction d'ordonnancement, on obtient un plan ou planning qui restitue les affectations de tâches fournies en entrée à une date précise, pour une durée déterminée de différentes ressources. Le calendrier est conçu pour atteindre les objectifs tout en respectant autant que possible les restrictions imposées. [16]

En pratique, la fonction d'ordonnancement se décompose en trois sous-fonctions (Figure1.3) :

- Elaboration d'Ordre de Fabrication (OF) : Cette tâche consiste à convertir les informations du programme principal de production en OF.
- Créer un plan d'atelier : cette tâche comprend la détermination en fonction des ordres de fabrication et de la disponibilité des ressources, le calendrier prévisionnel de fabrication.
- Le lancement et le suivi des opérations de fabrication.

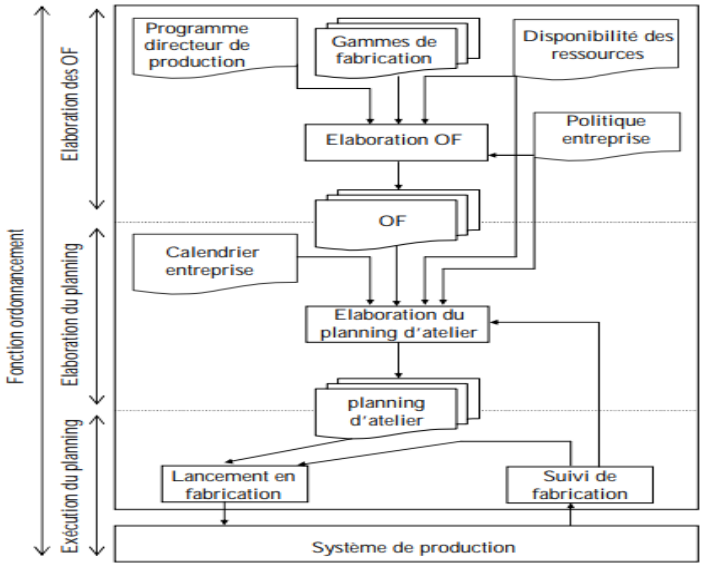


Figure 1.3 : Sous-fonctions de l'ordonnancement dans l'atelier.[16]

4. Présentation du problème d'ordonnancement

4.1. Définition du problème d'ordonnancement

Les problèmes d'ordonnancement consistent à organiser l'exécution des tâches dans le temps, en tenant compte des contraintes temporelles (délais, contraintes de séquence) et des contraintes liées à la disponibilité des ressources nécessaires. [D]

Dans la production (fabrication, biens, services), cela peut se manifester par un problème qui doit déclencher et contrôler la progression d'un ensemble de commandes à travers les différents centres qui composent le système.

L'ordonnancement est une solution au problème d'ordonnancement. Il est défini par des plans d'exécution de tâches (« ordres » et « calendriers ») et des allocations de ressources conçues pour répondre à un ou plusieurs objectifs. Les chronologies sont généralement représentées par des diagrammes de Gantt.

4.2. Éléments du problème d'ordonnancement

Dans la définition du problème d'ordonnancement, quatre éléments fondamentaux interviennent : les tâches, les ressources, les contraintes et les objectifs. Alors, la formulation et la description de ce problème se fait par la détermination de ces quatre éléments dits "de base". [11]

4.2.1. Les tâches

Une tâche est une entité de base localisée dans le temps par date de début et/ou de fin, sa réalisation nécessite une durée, et elle consomme des moyens selon une certaine intensité. Certains modèles incluent la notion de délais, les dates auxquelles les tâches doivent être terminées ; dans ces cas, les retards entraînent des amendes.[D]

Selon le problème, les tâches peuvent être exécutées par segments ou sans interruption ; on parle alors de problèmes préemptifs et non préemptifs séparément. 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.

De manière plus schématique : une tâche, notée i, est l'entité de base de travail, au moment de la date de début ti ou de la date de fin ci, sa réalisation nécessite une durée pi telle que pi = ci - ti, et utilise la ressource intensité kavec une intensité a_k^i. [26]

- La durée opératoire pi (processing time) : c'est la durée d'exécution de la tâche.
- La date de disponibilité ri (release time) : c'est la date de début au plus tôt.
- La date d'échéance di (due date) : c'est la date de fin au plus tard.

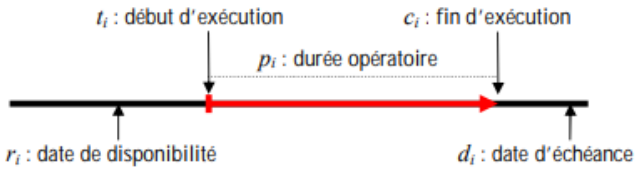


Figure 1.4 : Caractéristiques d'une tâche i. [III]

4.2.2. Les ressources

Une ressource est un moyen technique ou humain conçu pour accomplir une tâche et est fourni en quantités et capacités limitées.

Dans un environnement industriel, les ressources peuvent être des machines, des ouvriers, des équipements, des sites ou encore de l'énergie, des budgets, etc.

Plusieurs catégories de ressources peuvent être distinguées :

• Ressources renouvelables vs ressources durables

Une ressource est dite renouvelable si elle est disponible dans la même quantité après avoir été utilisée par une ou plusieurs tâches, telles que: les personnes, les machines, l'espace, etc.

En revanche, une ressource est consommable lorsqu'elle devient indisponible après utilisation ; comme les matières premières, l'énergie, etc.

• Ressources séparées et ressources non séparées pour accumuler des ressources

Une ressource est dite détachable (ou non partageable) si elle ne peut être affectée qu'à une seule tâche à la fois. Ce type traite principalement des ressources renouvelables.

Par contre, une ressource cumulative (partageable) peut être utilisée par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail, …)

Dans notre problème, les ressources qui sont des machines, sont renouvelables et disjonctives.

4.2.3. Les contraintes

Les contraintes représentent des restrictions sur les valeurs qu'une variable de décision peut prendre en même temps. Nous distinguons :

• Des contraintes temporelles

- Les contraintes de temps imparties, découlant généralement d'impératifs administratifs et liées aux échéances des tâches (heure de livraison, disponibilité des fournitures) ou à la durée globale du projet ;
- Contraintes de cohérence technique ou contraintes de périmètre, décrivant la relation séquentielle entre différentes tâches ;

• Contraintes de ressources

- Les contraintes d'utilisation des ressources, représentant la nature et le nombre de moyens utilisés par la tâche, et les caractéristiques d'utilisation de ces moyens ;
- Les contraintes de disponibilité des ressources, précisant la nature et la quantité de ressources disponibles dans le temps. Toutes ces contraintes peuvent être formalisées en termes de distances entre tâches ou d'apparitions potentielles.

• Contraintes de d'enchaînement

- Nous nous référons à une contrainte d'ordre ou de succession comme une contrainte qui lie le début ou la fin de deux activités par une relation linéaire. Il s'agit généralement de contraintes imposées par la cohérence technique (gamme d'opérations dans le cas d'un atelier), qui décrivent les positions relatives à respecter entre les tâches.

- D'autres contraintes plus spécifiques entre deux ou plusieurs tâches sont également imposées dans certains systèmes, telles que : la synchronisation, la simultanéité, le chevauchement, etc.

Les objectifs (critères) sont cités ci-dessous.

5. Les critères d'optimisation

5.1. Définition d'optimisation

Lors de la résolution de problèmes d'ordonnancement, on peut choisir entre deux principaux types de stratégies, chacune pour l'optimalité des solutions, ou, plus simplement, pour leur acceptabilité.

Les méthodes d'optimisation supposent que les solutions candidates aux problèmes peuvent être raisonnablement ordonnées selon un ou plusieurs critères d'évaluation numériques, qui sont construits sur la base de mesures de performance. Nous chercherons donc à minimiser ou maximiser ces critères. Par exemple, faites attention à ceux. [D]

Objectifs liés au temps : par exemple, nous avons constaté une minimisation du temps d'exécution total, du temps d'exécution moyen, de la durée totale ajustée ou des retards liés aux dates de livraison

Objectifs liés aux ressources : par exemple, maximiser la charge des ressources ou minimiser le nombre de ressources nécessaires pour effectuer un ensemble de tâches

Objectifs liés aux coûts : ces objectifs consistent généralement à minimiser les coûts, le démarrage, la production

5.2. Classification des critères d'optimisation

Les critères que doit satisfaire un ordonnancement sont variés. D'une manière générale, on peut distinguer trois catégories importantes : [11]

5.2.1. Les critères liés aux dates de fin et de livraison

Ce sont des critères liés aux paramètres temporels caractérisant les tâches. Les plus utilisés sont : [8][22][25]

- Cmax = max(c_i) : (Makespan), est la plus grande date d'achèvement des différentes tâches. C'est le temps nécessaire pour terminer toutes les tâches. Le makespan, est le critère le plus fréquemment utilisé.
- C$\sum$ = $\sum(c_i)$: la somme des dates de fin des tâches.
- $\sum$wiCi : la somme des dates de fin pondérées.
- $\overline{C}$: la moyenne arithmétique des dates de fin des tâches.
- Fmax = $max(F_i)$: le maximum des temps de séjour de l'ensemble des tâches.
- Lmax = $max(L_i)$: le maximum des retards algébriques de l'ensemble des tâches.
- $Tmax$ = $max(T_i)$: le maximum des retards absolus de l'ensemble des tâches.
- Emax = $max(E_i)$: le maximum des temps d'avancement de l'ensemble des tâches.

F$\sum$, L$\sum$, T$\sum$, E$\sum$ qui correspondent aux temps de : séjour total, retard total, retard absolu total et l'avancement total de l'ensemble des tâches, sont utilisés aussi.

D'autres critères correspondant aux moyennes arithmétiques de ces grandeurs, sont également utilisé : $\overline{F}$, $\overline{L}$, $\overline{T}$, $\overline{E}$. [26]

Tous ces critères sont à minimiser. Certains d'entre eux ne sont pas utilisés seuls comme : Emax ,E$\sum$… mais habituellement en association avec d'autres critères afin de construire un critère composé plus performant par exemple: $\alpha T\sum + \beta E\sum$. [18]

5.2.2. Les critères liés aux volumes des encours

Soit to la date de début de l'opération O et po sa durée d'exécution. A chaque instant t, on peut calculer les mesures suivantes caractérisant les volumes des encours. [11]

- Le nombre de tâches en cours d'exécution :

$$N_p(t) = \sum_{i=1}^{n} e_i(t) \;;\; \text{avec } e_i(t) = \begin{cases} 1 & si\ (t_o \le t < t_o + p_o) \\ 0 & sinon \end{cases}$$

L'optimisation de l'ordonnancement vise la maximisation de ce critère. Plus la valeur de ce critère est grande, plus le nombre de machines en attente est faible.

- Le nombre de tâches en attente d'exécution :

$$N_w(t) = \sum_{i=1}^{n} e_i(t) \;;\; \text{avec } e_i(t) = \begin{cases} 1 & si\ (t_o + p_o \le t < t_{o+1}) \\ 0 & sinon \end{cases}$$

La minimisation de ce critère permet de diminuer le nombre de tâches en attente dans les stocks intermédiaires des machines, et par conséquent de réduire la capacité intrinsèque des stocks.

- Le nombre de tâches terminées sur la dernière machine :

$$N_f(t) = \sum_{i=1}^{n} e_i(t) \;;\; \text{avec } e_i(t) = \begin{cases} 1 & si\ (C_i \le t) \\ 0 & sinon \end{cases}$$

La maximisation de ce critère permet de répondre dans les meilleurs délais aux demandes des clients.

On peut trouver dans certains cas que : $N_o(t)+ Nw(t)+ Nf(t) = N$, où N est le nombre de tâches à ordonnancer pour un horizon fixé. Par conséquent, les objectifs de production dans ce cas peuvent être complètement antagonistes. [11]

5.2.3. Les critères liés à l'utilisation des ressources

L'exploitation des ressources peut également être un objectif majeur du gestionnaire. Maximiser la charge d'une ressource ou l'utilisation moyenne des ressources, ou encore minimiser le temps d'inactivité de l'ensemble des ressources, sont des objectifs de ce type. [III]

- L'utilisation moyenne des ressources :

$$\overline{U} = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\right) \Big/ \left(\sum_{j=1}^{m} M_j \times C_{max}\right)$$

Ce critère permet de signaler éventuellement la sous-utilisation des ressources de l'atelier, en indiquant les périodes pleines et les périodes creuses de la chaîne de fabrication

- Le temps d'inactivité de l'atelier (Idle time) : [11]

$$I = \sum_{j=1}^{m} \left(C_{max} - \sum_{i=1}^{n} p_{ij} \right)$$

C'est la somme des tranches de temps perdu pendant l'exécution des tâches sur les machines. La minimisation de cet indicateur permet une meilleure exploitation des ressources.

5.3. Propriétés des critères

Deux propriétés importantes des critères de performance qui sont : la régularité des critères et la notion de dominance décrivent certaines relations entre un critère donné et l'ordonnancement correspondant. [III]

5.3.1. Critères réguliers

Soient $C_i(x)$ et $\delta(x)$, respectivement la date de fin de la dernière opération de la tâche Ji et la valeur d'un critère d appliqué à un ordonnancement x ; δ est fonction des dates de fin d'ordonnancement C_i.

Considérons deux ordonnancements x, x'. Si : $C_i(x) \leq C_i(x')$ $\forall$ i = 1, 2, …n => $\delta(x)$ £ $\delta(x')$, alors le critère δ est dit régulier. [26]

Autrement dit, un critère est régulier, s'il est une fonction décroissante des dates de fin d'exécution des opérations. Donc, on peut le dégrader en avançant l'exécution d'une tâche.

Parmi les critères réguliers : $Cmax$, $Tmax$, $Lmax$, $\bar{L}$, $\bar{T}$. Par contre, $\bar{E}$, $Emax$, I sont des critères non réguliers. [20]

5.3.2. Notion de dominance

Un sous-ensemble de solutions est dit dominer l'optimisation pour un critère donné s'il contient au moins une valeur optimale pour ce critère. De même, un avantage est défini par rapport à un ensemble de contraintes lorsque le sous-ensemble d'avantages contient au moins une solution acceptable qui satisfait toutes les contraintes, le cas échéant. Ainsi, la recherche de solutions optimales ou acceptables peut être restreinte à un sous-ensemble dominant. [III]

6. Classes d'ordonnancement

Afin de générer une solution au problème de planification de l'atelier de travail, le temps doit être maximisé. En effet, certaines solutions comportent des "trous" de chronométrage qui peuvent être exploités pour obtenir des horaires plus "compacts" grâce à de simples transferts d'opérations différées. Plus le planning est "compact", meilleure est sa qualité, actif, semi-actif, sans délai

6.1. Ordonnancement admissible (acceptable)

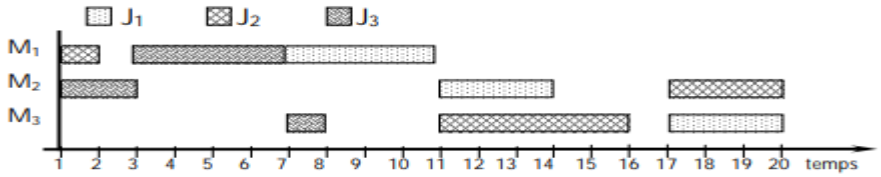


Figure 1.5 : Exemple d'un ordonnancement admissible. [III]

Un ordonnancement est dit admissible ou acceptable s'il respecte toutes les contraintes du problème. Par exemple, étant donné le problème Job Shop 3-tâches 3-machines, défini comme :

J1[M1:4 ; M2:3 ; M3:3] ; J2[M1:1 ; M3:5 ; M2:3] ; J3[M2:2 ; M1:4 ; M3:1] ;

L'ordonnancement de la figure 1.5, bien que non optimal, est acceptable (puisqu'il respecte toutes les contraintes).

Dans certains cas, certaines opérations nécessitent un décalage vers la gauche. Selon que l'ordre des opérations reste le même, on distingue deux cas :[22]

- On parle de décalage à gauche "local", lorsqu'on avance le début d'une opération sans remettre en cause l'ordre relatif entre les autres opérations ;
- On parle de décalage à gauche "global", lorsqu'on avance le début d'une opération en modifiant l'ordre relatif entre au moins deux opérations.

6.2. Ordonnancement semi-actif

Si un décalage local à gauche n'est pas possible, l'ordonnancement est dit semi-actif. Aucune opération ne peut être effectuée à l'avance sans changer l'ordre relatif d'au moins deux opérations. Toutes les opérations sont mises en correspondance, soit avec des opérations précédentes dans la gamme, soit avec des opérations précédentes sur la machine utilisée. Par exemple, l'horaire illustré à la figure 1.6 est semi-actif et les transferts locaux ne sont pas possibles.

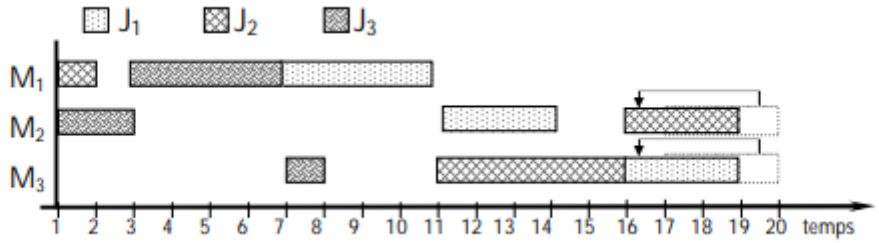


Figure 1.6 : Décalage à gauche local de O1,3 et O2,2 dans l'ordonnancement de la figure 2.1, conduit à un ordonnancement semi-actif. [III]

6.3. Ordonnancement actif

Un ordonnancement est dit actif si un décalage à gauche local ou global n'est pas possible. Par conséquent, en ordonnancement actif, il est impossible d'avancer une opération sans retarder le début d'une autre opération. [22]

Le programme illustré à la Figure 1.7 est actif car les décalages à gauche, les décalages locaux ou globaux ne peuvent pas être effectués.

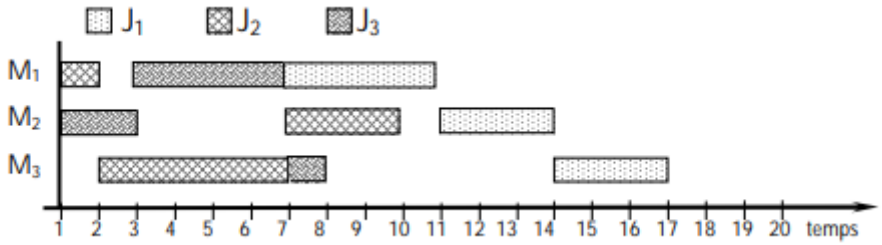


Figure 1.7 : Exemple d'ordonnancement actif. [II]

6.4. Ordonnancement sans délai

On dit que la planification n'a pas de retard ou sans délai si et seulement si aucune opération n'est en attente pendant que la machine est disponible pour l'exécution. [26]

Par conséquent, le programme illustré à la figure 1.8, bien qu'il soit actif, n'appartient pas à cette classe (c'est-à-dire qu'il a un retard). En fait, l'opération O_1,1 en attente de la machine M1 est mise en attente, alors que M1 est inactif à t = 1, elle peut basculer sans délai, comme le montre la figure 2.4

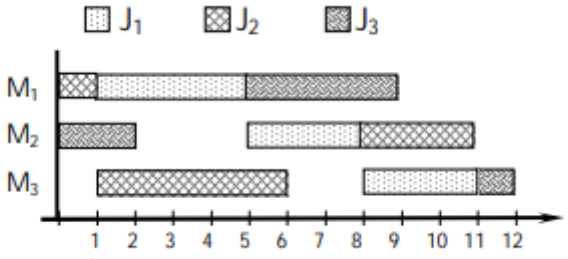


Figure 1.8 : Ordonnancement sans délai. [III]

Notez que la transition vers un ordonnancement sans délai peut conduire à une solution moins bonne du point de vue du temps de fabrication

La figure 2.5 montre l'inclusion des catégories d'ordonnancement vues précédemment. La figure montre que les ordonnancements sans délai sont inclus dans un sous-ensemble d'ordonnancement actifs, eux-mêmes inclus dans un sous-ensemble d'ordonnancement. Ordonnancement semi-actif. Un ordonnancement acceptable comprend toutes les contraintes qui satisfont le problème.

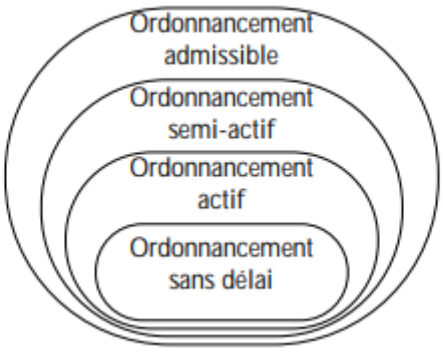


Figure 1.9 : Relations d'inclusion entre les différentes classes d'ordonnancements.[22]

7. Typologie des problèmes d'ordonnancement

On peut classer les problèmes d'ordonnancement en 5 grandes familles. Cette classification est représentée en arborescence dans la figure suivante :

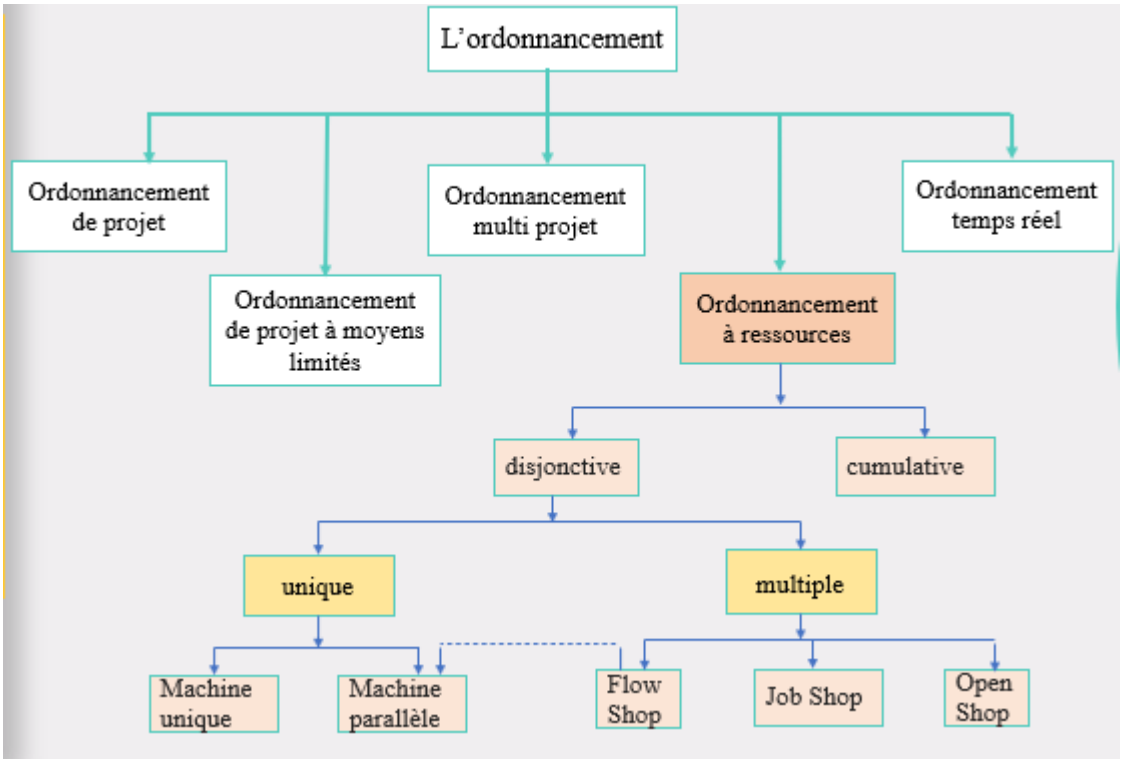


Figure 1.10 : typologie des problèmes d'ordonnancement

8. Classification des problèmes d'ordonnancement à ressources

Les problèmes d'ordonnancement sont généralement divisés en deux modèles principaux basés sur le nombre d'opérations requises pour un travail : les modèles à opération unique (machines simples et parallèles) et les modèles à opérations multiples (Flow shop, open shop et job shop).

8.1. Modèles à une opération

8.1.1. Modèle à machine unique

Dans un modèle de machine unique, toutes les tâches à effectuer sont réalisées par une seule machine. Une situation intéressante où nous pouvons rencontrer cette configuration est lorsque nous sommes devant un système de production qui contient une machine à goulot d'étranglement qui affecte l'ensemble du processus. Le modèle est illustré à la figure 1.11

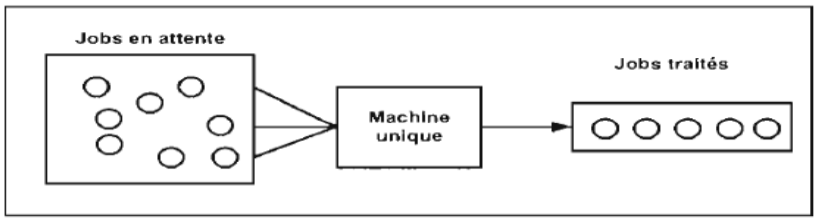


Figure 1.11 : modèle à machine unique. [II]

8.1.2. Modèle à machines parallèle

Le deuxième modèle est le modèle parallèle. Ce modèle est principalement utilisé dans le secteur industriel, tel que : l'industrie alimentaire, la plasturgie, les fonderies et surtout l'industrie textile. Le processus de ce système de production est le suivant : chaque fois que la machine i devient inactive, une tâche j lui est affectée, comme le montre la figure1.12. Par exemple, dans un processus d'assemblage industriel, si une des étapes d'assemblage prend beaucoup de temps, il peut être très intéressant d'avoir plusieurs machines en parallèle pour effectuer la même tâche. D'autre part, les machines parallèles sont classées en fonction de leur vitesse. Toutes les machines de l'ensemble sont identiques si elles ont la même vitesse de traitement et exécutent les mêmes tâches. Les machines sont dites uniformes si elles ont des vitesses de traitement différentes mais linéaires. Dans le cas où les vitesses des machines sont indépendantes les unes des autres, on parle alors de modèle de machines parallèles non reliées ou indépendantes. [II]

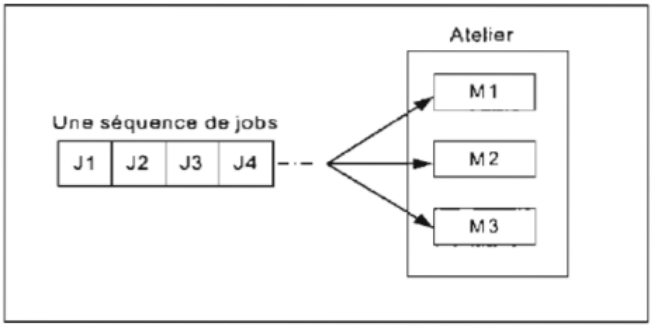


Figure 1.12 : Modèle à machines parallèle. [II]

8.2. Modèles à plusieurs opérations

Les modèles à opérations multiples incluent des situations où le travail à effectuer doit passer par plusieurs machines, chacune ayant ses particularités. Il se décompose en trois modes selon l'ordre de passage des travaux dans la machine, à savoir le mode atelier flux, l'atelier travail et le mode atelier ouvert.

8.2.1. Modèle flow-shop

Dans le modèle flow shop, les ordres de production visitent les machines dans le même ordre et le temps de fonctionnement peut varier. Chaque travail sera exécuté séquentiellement sur M machines, et tous les travaux suivent le même ordre de passage sur ces machines. Ce modèle est aussi appelé modèle linéaire. La figure 1.13 illustre la situation d'un flow shop avec quatre machines et quatre jobs. Les quatre travaux suivent le même ordre de traitement sur les quatre machines.

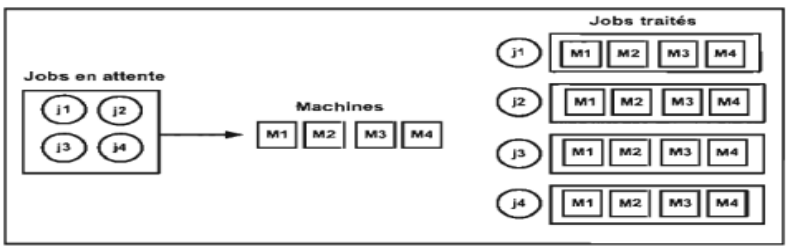


Figure 1.13: Modèle flow-shop. [II]

8.2.2. Modèle job-shop

Concernant le modèle du job shop, chaque job a une commande, et chaque job peut être exécuté plusieurs fois sur la même machine ; ce n'est pas le cas avec un flow shop. Dans ce cas, le problème est de déterminer les dates de fin des différentes ressources pour les ordres de fabrication avec des tournées différentes dans l'atelier. Étant donné que ces ordres de production partagent des ressources communes, des conflits sont susceptibles de survenir, causés par l'intersection des processus illustrés à la figure 1.14. Dans ce diagramme, l'ordre dans lequel chaque travail est exécuté est représenté par un chemin de la même couleur que le travail. Par exemple, le travail J1 passe par toutes les machines, tandis que le travail J3 passe uniquement par les deuxième et quatrième machines. Bref, le problème est de gérer ces conflits en respectant les contraintes données, et d'optimiser ce qui est poursuivi. Les types de ressources et de contraintes prises en compte peuvent toutefois considérablement compliquer le problème. Plus on intégrera de contraintes, plus on se rapprochera d'un cas réel, mais moins on disposera de méthodes de résolution satisfaisantes.

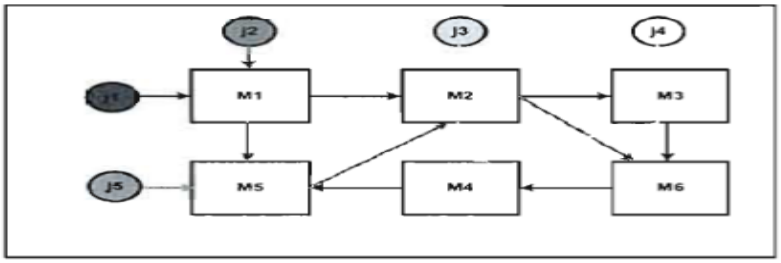


Figure 1.14: Modèle job-shop. [II]

8.2.3. Modèle open-shop

Dans le modèle open shop, l'ordre de passage de n jobs sur m machines n'est pas connu à l'avance, et cet ordre est déterminé lors de la construction de la solution. Chaque travail j peut avoir son propre ordre d'exécution sur toutes les machines. Le fait qu'il n'y ait pas d'ordre prédéterminé complique la résolution de ce type de problème d'ordonnancement, mais offre tout de même des degrés de liberté intéressants. Dans la figure 1.15, nous avons un ensemble de quatre tâches et un ensemble de quatre machines. [II]

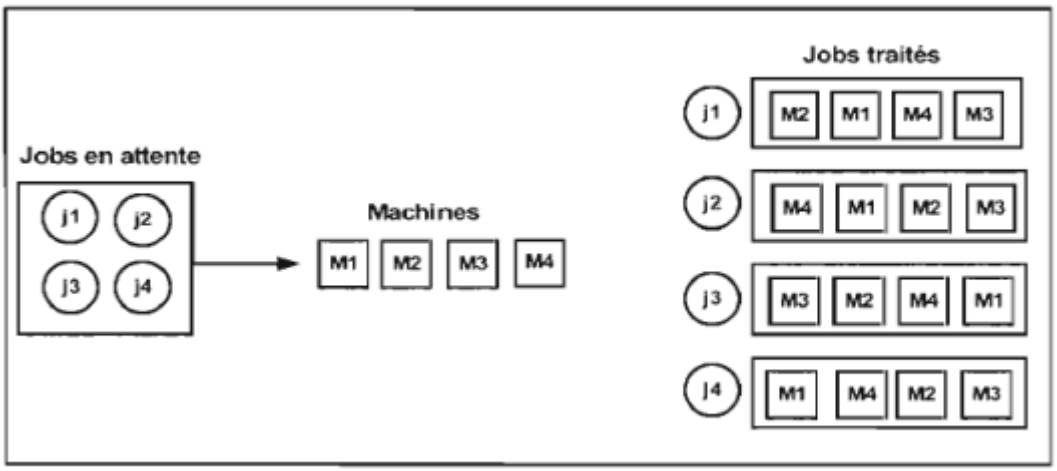


Figure 1.15: Modèle open-shop. [II]

9. Méthodes de résolution

Les méthodes de résolution d'un problème d'ordonnancement sont celles des méthodes de résolutions d'un problème d'ordonnancement combinatoire.

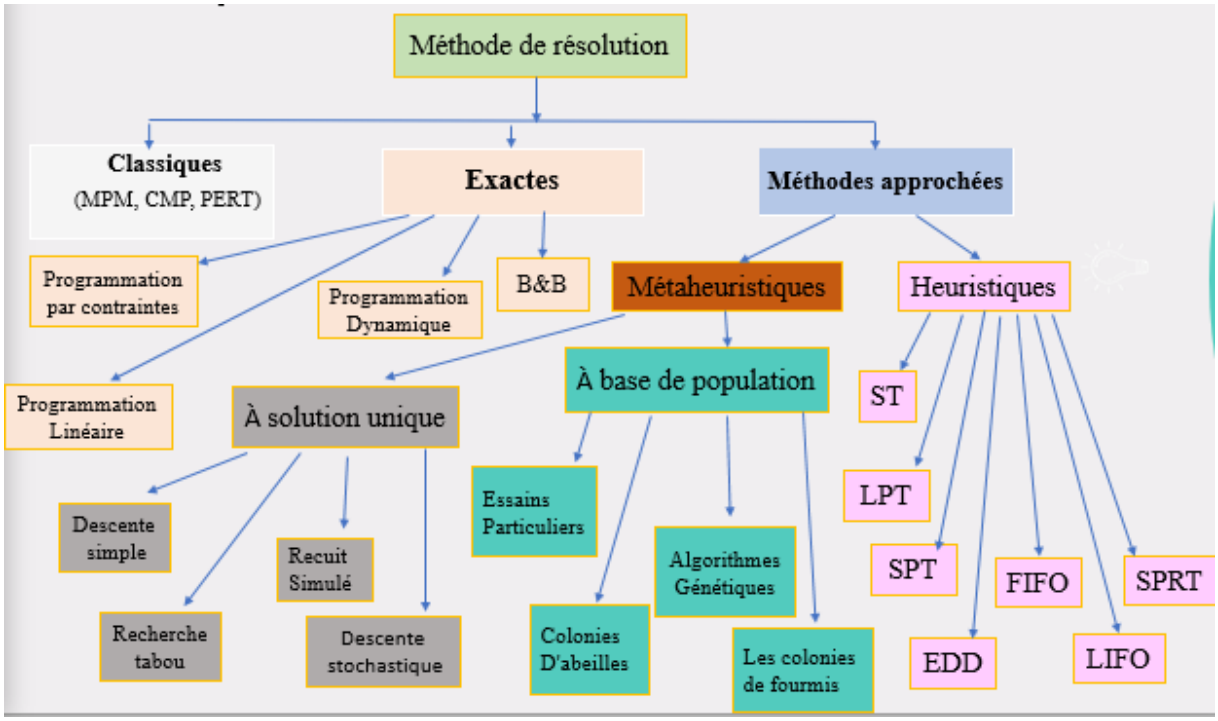


Figure 1.14: méthodes de résolutions des problèmes d'ordonnancement

10. Conclusion

Nous discutons des aspects communs des problèmes d'ordonnancement dans ce chapitre : production, notamment en expliquant les notions de base sur ces problèmes. D'abord, le cadre général de la planification en tant que fonction originelle du système de gestion des ordres de travail à discuter de la production, une brève introduction aux systèmes de production, à la gestion, le rôle de la production et de l'ordonnancement dans ces systèmes.

Le deuxième point exposé dans ce chapitre est la caractérisation du problème programmation de l'atelier en proposant sa définition et en précisant les différents éléments qui le déterminent: tâches, ressources, contraintes et critères l'optimisation ainsi que les méthodes de classification qui s'appuient sur un ensemble de classifications, les paramètres liés aux différentes caractéristiques du problème, y compris l'organisation de l'atelier (Job Shop, Flow Shop, Open Shop) tel que décrit ci-dessous et qui constituent la classification.

Chapitre 2 Le problème d'ordonnancement Job shop

1. Introduction

Le problème d'ordonnancement de Job Shop est un cas particulier du problème général d'ordonnancement. C'est l'un des problèmes de la théorie de l'ordonnancement et de l'optimisation combinatoire les plus traités.

Dans ce chapitre, nous présentons d'importantes bases théoriques et pratiques liées à ce problème, en se concentrant sur les contraintes, la difficulté, la modélisation et

La solution. Par conséquent, la deuxième partie décrit spécifiquement le problème : données, contraintes et critères d'évaluation. La troisième partie présente différentes classes d'ordonnancement. Manières importantes de poser des problèmes donnés dans la quatrième section. Dans la partie 5, nous avons discuté de la complexité de problème. La section 6 est consacrée à une synthèse des solutions de contournement connues énumérez les moyens importants de résoudre le problème. Enfin, quelques problèmes typiques sont soulevés, connus sous le nom de "benchmarks".

2. Présentation de problème Job shop

Le problème d'ordonnancement job shop est un cas particulier du problème d'ordonnancement général. C'est l'un des problèmes les plus étudiés en théorie de l'ordonnancement et en optimisation combinatoire.

Il existe de nombreuses variantes du problème Job Shop. Nous donnons ici une formulation générale du problème simple de Job Shop, puis spécifions les contraintes qui caractérisent le cas de Job Shop discuté dans cet article. [III]

Un problème d'ordonnancement job shop implique l'exécution d'un ensemble de n tâches sur un ensemble de m ressources (machines) cherchant à atteindre un objectif. Chaque tâche J_i consiste en une séquence de n_i opérations qui seront exécutées sur différentes ressources selon un ordre préalablement défini. De plus, un ensemble de contraintes concernant les ressources et les tâches doivent être respectées.

Par conséquent, la détermination du problème Job Shop m*n composé de n tâches et de m machines se fait en spécifiant les données, les contraintes et les objectifs du problème.

2.1. Les données

Les données du problème sont : [8][27]

- Un ensemble M de m machines : Une machine est notée M_K avec k = 1,., m. Chaque machine ne peut effectuer qu'un seul type d'opérations. Le nombre total d'opérations exécutées par cette machine est noté m_K
- Un ensemble J de n tâches : Une tâche est notée J_i avec i = 1,., n. Chaque tâche est composée d'une gamme opératoire, i.e. une séquence linéaire fixée de n_i opérations. Cette séquence ne dépend que de la tâche, et peut varier d'une tâche à l'autre.

Cet ensemble d'opérations d'une tâche $J_i \in J$, est défini par :
$Oi = \{Oi,1 \bullet \dots \bullet Oi,k\}$, tel que : $Oi \subset O$;

Où • est l'opérateur de précédence. Il définit un ordre total sur l'ensemble des opérations de la même gamme.

- L'opération Oij est la j-ème opération dans la gamme opératoire de Ji. Elle se caractérise par :
  -La machine sur laquelle elle s'exécute : Mк . Le fait que la machine demandée par l'opération Oij soit Mк s'écrit : R(Oi,j) = Mk.
  -Le temps opératoire pi,j qui correspond à la durée de Oi,j sur Mк.
  Le nombre total des opérations dans l'atelier est noté : no = $\sum_{i=1}^{n} ni$.

2.2. Les contraintes

La définition du cas général de Job Shop se limite aux données décrites ci-dessus, et aucune contrainte supplémentaire n'est imposée. Cependant, les recherches menées sur ce sujet ont ajouté diverses contraintes afin de formuler des cas spécifiques. Souvent, ces contraintes affectent la probabilité d'utilisation de la machine et les liens possibles entre les opérations.
De plus, les contraintes varient d'une formule (selon le type d'atelier). Cependant, nous ne considérons ici que le cas d'un Job Shop simple avec les contraintes suivantes : [III]
• Les machines sont indépendantes les unes des autres (par exemple, aucun outil commun n'est utilisé).
• Les tâches sont indépendantes les unes des autres. En particulier, il n'y a pas d'ordre de priorité attaché aux tâches.
• Une tâche ne peut s'exécuter que sur une seule machine à la fois. Deux opérations d'une même tâche ne peuvent pas être effectuées en même temps.
• Une machine ne peut effectuer qu'une seule opération à la fois.
• Seul le temps d'exécution réel est pris en compte. Temps d'expédition, temps de préparation, etc. d'une machine à l'autre. Ne pas envisager.
• Les machines sont disponibles jusqu'à la fin du programme. En particulier, les pannes de machines n'ont pas été prises en compte.
• Une opération en cours ne peut pas être interrompue (pas de préemption).
• Les tâches peuvent attendre des ressources aussi longtemps qu'elles en ont besoin. Il n'y a pas de date limite.
• La définition de la tâche est déterminée avant le démarrage de la planification, c'est-à-dire qu'il n'y a pas d'événements aléatoires pendant l'exécution.

Des formulations plus rigoureuses du problème de Job Shop sont souvent trouvées. Dans ce cas, chaque tâche passe par m machines dans l'atelier une fois et une seule. Ainsi, toutes les tâches consistent en m opérations, et chaque machine doit effectuer n opérations (n : nombre de tâches). Alors, le nombre total d'opérations est n_o = n × m. Si n = m, le problème est dit carré.
Le problème Job Shop de cette forme est dit simple : chaque tâche est constituée d'un planning, et chaque opération ne peut être exécutée que sur une seule machine. Un job shop est dit générique (ou étendu) si une tâche est constituée d'un ou plusieurs plannings

(qui peuvent être répétitifs) et qu'une machine peut exister dans une ou plusieurs répliques.

2.3. Les objectifs

Le problème d'ordonnancement a pour but de déterminer la date de début de l'opération. Pour cela, il faut déterminer l'ordre de passage de toutes les tâches sur chaque machine, tout en respectant les contraintes du problème. Le but est alors de minimiser ou de maximiser la fonction objectif pour trouver la meilleure solution. Cette fonction objectif est également appelée dans ce contexte : critère de performance, critère d'évaluation ou objectif de planification. [5]

Le résultat de la planification, généralement représenté dans un diagramme de Gantt, dépend de la date de début calculée de l'opération. [21]

Nous avons déjà vu un certain nombre de normes de performance pour la planification des ateliers au chapitre 1, section 4, qui s'appliquent dans une large mesure à la situation de l'atelier. Ces normes guident la solution des problèmes d'ordonnancement.

Nous n'avons examiné ici que quelques-uns des critères les plus importants : [III]

- Cmax : le makespan, est la durée totale de l'ordonnancement.
- $\overline{C}$: est la moyenne des temps d'achèvement des tâches.
- $\sum wiCi$: est la somme des dates de fin pondérées.
- Tmax, Emax : sont respectivement, le retard maximum des tâches et le retard maximum des tâches par rapport à une date d'achèvement prévue.
- $\overline{T}$: est la moyenne des retards de l'ensemble des tâches.

3. Modélisation graphique du problème job shop

Traiter un problème complexe comme le Job Shop nécessite une modélisation claire et précise. Il existe plusieurs façons de modéliser ce problème et donc de choisir un mode de description approprié.

Si la modélisation des graphes disjonctifs est la plus classique et la plus universelle, la modélisation en programmation mathématique est la plus ancienne [7]. Il y a aussi l'algèbre, les polyèdres, les graphes de tâches latentes, les réseaux de Petri, la modélisation basée sur UML, et plus encore.

3.1. Modélisation par graphe disjonctif

La modélisation sous forme de graphes disjonctifs est largement utilisée pour représenter les problèmes d'ordonnancement, en particulier les problèmes de job shop. Cette formule a été proposée pour la première fois par B. Roy et B. Sussmann en 1964 et est à l'origine de la première méthode de résolution de problèmes. [14][21]

Un problème de Job Shop peut alors se modéliser par un graphe disjonctif G (X , C∪D) où:

- X : est l'ensemble des sommets : chaque opération correspond à un sommet, avec deux opérations fictives S (source) et P (puits) désignant le début et la fin de l'ordonnancement.

- C : est l'ensemble des arcs conjonctifs représentant les contraintes d'enchaînement des opérations d'une même tâche (gammes opératoires). Pour un arc (ij,iк) de la partie conjonctive, on a :

  $T_{i\,j} - t_i\,к \geq p_{i,к} \forall\, (ij,_iк) \subset C$

On aura donc un arc entre tous les sommets (i, j), (i, j+1) pour i = 1…n et j = 1…ni-1 ; (n: nombre de tâches, ni : nombre d'opérations de la tâche i). De plus, il existe des arcs entre S et tous les sommets (i,1), et d'autres arcs entre les sommets (i, ni) et le puits P

- D : l'ensemble des arcs disjonctifs associés aux conflits d'utilisation d'une machine. Pour un arc (ij,iк) de la partie disjonctive, on a : [19]

  $t_jк - t_iк \geq p_iк$ ou $t_iк - t_jк \geq p_iк \forall\, (iк,jк) \in D$

  Que l'on modélise par un arc et un arc retour entre les sommets (tjк, tiк) représentant deux opérations utilisant la même machine.

Toutes les paires disjonctives associées à la même machine forment une clique disjonctive.

Cette forme de graphe disjonctif modélise un problème d'ordonnancement. Pour établir un ordonnancement acceptable sur ce graphe, il suffit de sélectionner pour chaque paire d'arcs qui déterminera l'ordre de passage des deux opérations sur la machine, c'est-à-dire d'arbitrer pour chaque disjonction. L'arbitrage doit être complet (toutes les disjonctions arbitrées), et compatible (pas de circuit dans le schéma). [12]
La figure 2.6 correspond au graphe disjonctif non arbitré du problème précédent (section 3.1). Un diagramme d'arbitrage représentant le calendrier de la Figure 2.3 est présenté à la Figure 2.7.

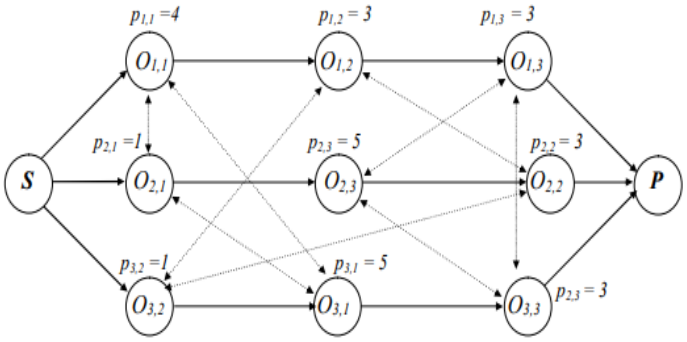


Figure 2.1 : Le graphe disjonctif du problème de Job Shop du § 3.1

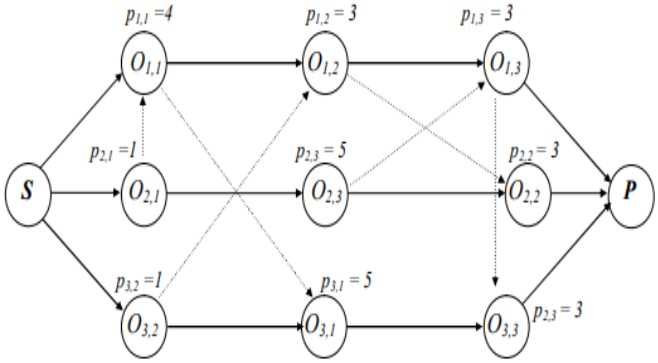


Figure 2.2 : Le graphe disjonctif arbitré simplifié de l'ordonnancement de la figure 2.3

Pour un ordonnancement du temps donné, quelques concepts très évidents sur les graphes disjoints méritent d'être mentionnés :

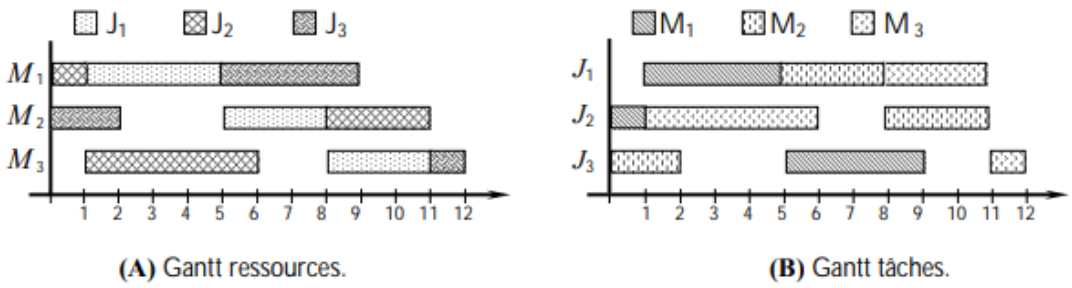
- Opérations critiques : Etant donné un programme actif, une opération est dite opération critique si elle entraîne nécessairement une augmentation de la période de fabrication du programme lorsqu'elle est retardée (alors que l'ordre d'exécution des opérations défini par le programme ne change pas).
- Chemin critique : Un chemin critique est une série d'opérations critiques liées par des relations de précédence. La longueur d'un chemin critique est égale à la somme des durées des opérations qui le composent. Pour un ordonnancement donné, il peut y avoir plusieurs chemins critiques.
- Bloc critique : Un bloc critique est une série d'opérations critiques s'exécutant sur la même machine. Tout chemin critique peut être décomposé en b blocs critiques. Pour un problème n tâche m machine, b est compris entre 1 et n :$1 \leq b \leq n$

3.2. Présentation de l'ordonnancement

La représentation la plus courante d'un planning est un diagramme de Gantt. Cela représente une opération à travers un segment ou une barre horizontale dont la longueur est proportionnelle à son temps d'opération.

Ainsi, sur ce graphique, selon l'échelle de temps sont indiqués : l'occupation de la machine par les différentes tâches, le temps mort, et l'éventuelle indisponibilité de la machine due aux changements entre produits. [18]

Utilisez deux types de diagrammes de Gantt : Gantt des ressources et Gantt des tâches. [19]



(A) Gantt ressources. **(B)** Gantt tâches.

Figure 2.3 : Diagrammes de Gantt (problème de Job Shop du § 3.1).

4. Complexité du problème job shop

La complexité du problème est liée au problème à résoudre et à la solution de contournement utilisée pour développer la meilleure solution par rapport aux critères de rétention.

La théorie de la complexité divise les problèmes en deux catégories, P et NP. La classe P est une collection de problèmes que les algorithmes polynomiaux peuvent résoudre. Un algorithme est dit polynomial lorsque son temps d'exécution est O(P(x))où P est le polynôme et x est la longueur d'entrée de l'instance du problème [8]. Les algorithmes polynomiaux représentent une classe stable, la combinaison de deux algorithmes polynomiaux donne un algorithme polynomial, et un algorithme construit à partir d'un

polynôme à partir d'un processus d'invocation de la complexité polynomiale est toujours polynomial.

Les algorithmes dont la complexité ne peut être bornée par un polynôme sont appelés algorithmes exponentiels et correspondent à la classe NP « temps polynomial non déterministe ». Les problèmes de NP peuvent être résolus en énumérant un ensemble de solutions possibles et en les testant à l'aide d'un algorithme polynomial. On peut associer à tous les problèmes de décision à chacun d'eux un ensemble de solutions potentielles (de la cardinalité au pire exposant) de sorte qu'il existe un algorithme polynomial qui puisse vérifier si les solutions trouvées sont valides. [IV]

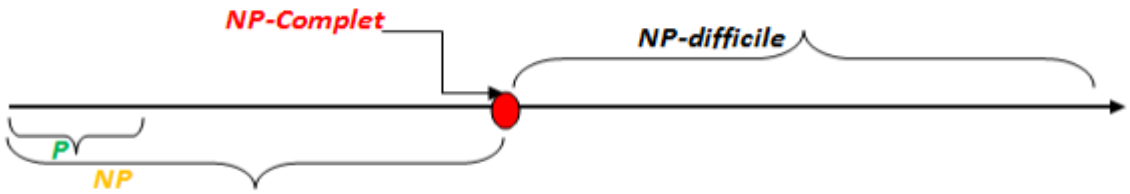


Figure 2.4 : Représentation des Classes P, NP, NP-complet et NP-difficile. [IV]

4.1. L'importance de l'espace de solutions

Le problème Job Shop est classé par la théorie de la complexité comme un problème NP dur (difficile) au sens strict [2]. Pour illustrer l'importance de l'espace des solutions pour les instances de problème, rappelons que pour un problème de n tâches à exécuter sur m machines, il existe $(n\,!)^m$ solutions différentes. Ainsi pour un problème de taille 10×10, il existe 39594×1065 solutions différentes (à titre de comparaison, la Terre n'a pas plus de 1018 secondes). Connaître ce nombre ne compte pas comme une planification semi-active. Il n'est pas réaliste d'énumérer toutes ces possibilités pour arriver à la meilleure solution.

A noter que ce nombre évolue plus vite en fonction de J qu'en fonction de M. En d'autres termes, l'ajout de tâches à un impact beaucoup plus important que l'ajout de machines. Par exemple : [III]

- Pour un problème de 4 tâches et 5 machines (4´5) : $(n!)^m$= 7962624 ;
- Alors que, pour un problème de 5 tâches et 4 machines (5´4) : $(n!)^m$= 207360000.

La difficulté du Job Shop est fonction du nombre de tâches, du nombre de machines, du nombre d'opérations par tâche, et de la durée des opérations.

Le tableau 2.1 ci-dessous résume les cas où le problème peut être résolu en temps polynomial et les cas où il devient NP. [6]

<table>
<tr><th>P</th><th colspan="2">NP</th></tr>
<tr><td>$m = 2, \forall j \;\; n_j \leq 2$</td><td>$m = 2, \forall j \;\; n_j \leq 3 \;\; [\exists k, n_k = 3]$</td><td rowspan="2">$m = 2$
autre cas</td></tr>
<tr><td>$m = 2, \forall o \;\; d(o) = 1$</td><td>$m = 2, \forall o \;\; d(o) \leq 2 \;\; [\exists k, d(k) = 3]$</td></tr>
<tr><td rowspan="2"></td><td>$m = 3, \forall j \;\; n_j \leq 2$</td><td rowspan="2">$m \geq 2$</td></tr>
<tr><td>$m = 3, \forall o \; d(o) = 1$</td></tr>
<tr><td>$n = 2$</td><td colspan="2">$n \geq 3$</td></tr>
<tr><td>$C_{max} \leq 3$</td><td colspan="2">$C_{max} \geq 3$</td></tr>
</table>

Tableau 2.1 : Complexité du Job Shop.

n : nombre de tâches,m : nombre de machine,
nj : nombre d'opération par tâche, d(o) : durée de l'opération.

4.2. Difficulté relative des instances

La taille des instances en cours de traitement est importante pour déterminer leur difficulté. B. Penz a proposé de classer le problème selon le nombre de jobs, le nombre d'opérations par job et le nombre de machines. [21]

Cette classification n'est même pas si stricte [21], la taille d'un problème donné peut être facilement identifiée en identifiant trois catégories :

Problèmes mineurs : problèmes avec moins de 20 tâches, moins de 10 machines et moins de 10 opérations par tâche.

Problèmes moyens : 20 à 50 tâches, 5 à 15 machines, entre 5 et 15 opérations par tâche.

Gros problème : problème avec plus de 50 tâches, plus de 5 machines, et plus de 5 opérations par machine.

La taille n'est pas le seul déterminant de la difficulté, en fait d'autres paramètres affectent la difficulté du problème. Par conséquent, il est montré que le rapport M/j détermine la difficulté de l'instance.Ce rapport explique la conjecture de Taillard, qui stipule :[6]

"Les instances de n ≈ m sont plus difficiles à résoudre que les instances de n >>m; la frontière entre les instances faciles et difficiles est d'environ n = 5m".

5. Méthode de résolution du problème job shop

Nous avons vu précédemment que le problème d'ordonnancement de job shop est NP-difficile sauf dans certains cas. Les problèmes appartenant à la classe P ont des algorithmes polynomiaux pour les résoudre. Pour les problèmes appartenant à la classe NP, l'existence d'algorithmes polynomiaux semble irréaliste. Différentes solutions (exactes ou heuristiques) sont largement utilisées pour résoudre les problèmes de Job Shop NP-difficiles.

5.1. Les métaheuristiques choisi pour notre problème

Les métaheuristiques d'optimisation sont des algorithmes d'optimisation à usage général applicables à une variété de problèmes. Ils existent depuis les années 80 et sont conçus pour résoudre au maximum les problèmes d'optimisation. Les métaheuristiques travaillent sur la résolution de tout type de problème d'optimisation. Ils se caractérisent par leur caractère aléatoire ainsi que par leur origine discrète.

5.1.1. Classification des métaheuristiques

Nous pouvons diviser les méta-heuristiques en deux grandes catégories : les méta-heuristiques à solution unique (c'est-à-dire qui évoluent à l'aide d'une seule solution) et les méta-heuristiques à solutions multiples ou groupes de solutions. La méthode d'optimisation basée sur la population de solutions améliore la population de solutions à mesure que l'itération progresse. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

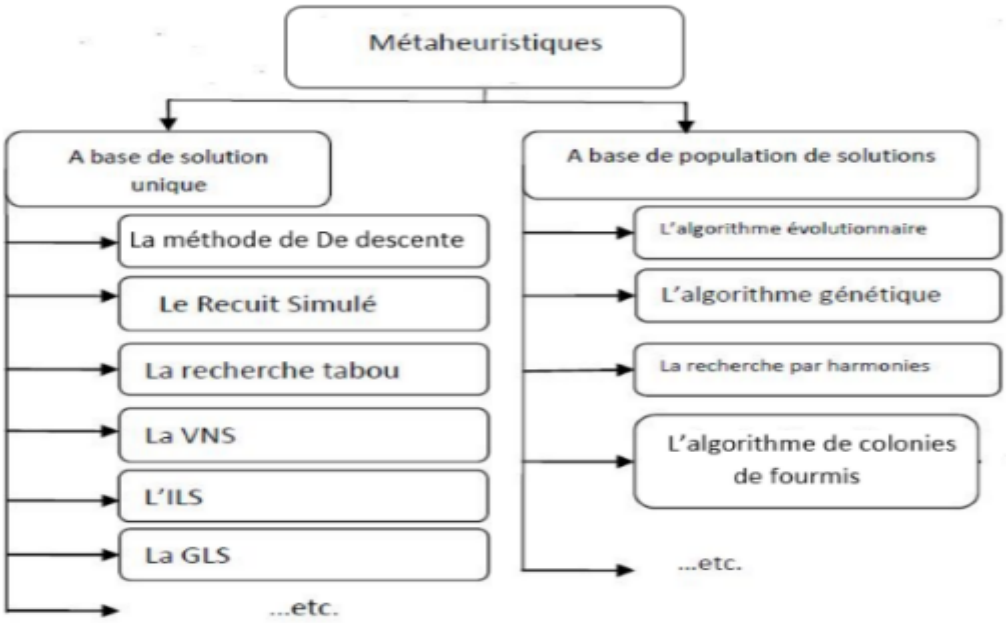


Figure 2.5 : Classification des métaheuristiques. [II]

5.1.2. Les algorithmes génétiques

Les algorithmes génétiques tentent de reproduire l'évolution naturelle des individus en respectant ce que Darwin appelait la loi de la survie. Les principes de base des algorithmes génétiques, initialement développés par Holland pour répondre aux besoins spécifiques de la biologie, ont été rapidement appliqués pour résoudre avec succès des problèmes d'optimisation combinatoire en recherche opérationnelle et des problèmes d'apprentissage en intelligence artificielle. [D]

5.1.3. La Recherche Tabou

Cette technique a été inventée par Glover et représente une variante du paradigme de la recherche locale. Le composant principal de la recherche tabou est la fonction de voisinage qui détermine l'efficacité de la recherche. Le principe est d'augmenter la valeur de la fonction objectif à chaque étape. Utilisez la mémoire qui contient des informations sur les solutions auxquelles vous avez accédé.

Le contenu de cette mémoire représente la liste tabou, utilisée pour interdire l'accès à la dernière solution accédée. Lorsque l'optimum local est atteint, le retour en arrière est interdit. [D]

5.1.4. Recuit simulé

Inspiré du recuit physique, ce procédé est utilisé en métallurgie pour améliorer la qualité des solides et recherche l'état énergétique minimum correspondant à la structure stable du solide. Ainsi, pour qu'un métal retrouve une structure proche d'un cristal parfait, on le chauffe à haute température et on le laisse refroidir lentement afin que les atomes aient le temps de s'ordonner régulièrement.

En optimisation, le processus de recuit simulé répète un processus itératif qui recherche des configurations à moindre coût tout en acceptant des configurations qui réduisent la fonction de coût de manière contrôlée. Ainsi, si une amélioration de la norme est

constatée, l'état neuf est conservé,sinon une diminution ΔE du critère est calculée et le nouvel état est retenu avec une probabilité $P = e^{-\Delta E/T}$

Ainsi, à partir de la solution initiale S_0, à chaque nouvelle itération, les voisins S'□N(s) de la configuration courante S sont générés aléatoirement. Selon les cas, ce voisin est soit choisi pour le remplacer, soit rejeté.

- si les performances de ce voisin sont supérieures ou égales aux performances de la configuration courante (f(s') ≤ f(s)), le conserver systématiquement ;

- Sinon, il est accepté avec une probabilité P qui dépend de deux facteurs :

□ Importance de la dégradation Δf = f (s') - f (s), la dégradation la plus faible est la plus acceptable.

□ D'après le paramètre de contrôle T (température), une température élevée correspond à une plus grande probabilité d'accepter la dégradation.

Typiquement, la température est contrôlée par une fonction décroissante qui définit le mode de refroidissement. Il y a trois options :

□ Réduction étagée : La température est maintenue constante pendant un certain nombre d'itérations, ce qui entraîne une réduction étagée.

□ Réduction continue : La température est modifiée à chaque itération.

□ Diminution non monotone : La température diminue à chaque itération et augmente occasionnellement. [D]

6. Conclusion

L'ordonnancement d'ateliers de type Job Shop est considéré un problème NP-Difficile. En dépit de sa simple modélisation et définition par précision des données (tâches et machines), des contraintes et des objectifs, la résolution du problème est une tâche difficile, le classant ainsi parmi les problèmes NP-Difficiles au sens fort. Ce qui le rendant un terrain d'essai privilégié pour le test d'une multitude de méthodes notamment les nouvelles approches de résolution, comme les Métaheuristiques, qui font le sujet de notre application, et qui sont présentées dans le prochain chapitre.

Chapitre 3 Les métaheuristiques

1. Introduction

Les métaheuristiques sont un paradigme très important des méthodes d'approximation qui visent à résoudre des problèmes combinatoires comme Job Shop. Grâce à eux, nous pouvons excellente solution à un plus gros problème aujourd'hui, et De nombreuses demandes qui ne pouvaient pas être traitées auparavant. Nous introduisons trois principes de base des métaheuristiques dans ce chapitre est le sujet de cet article : Algorithmes génétiques, recherche taboue et recuit simulé. Nous donnons d'abord un aperçu général des métaheuristiques. Ensuite nous Dans les trois prochaines sections, nous aborderons les concepts de base liés à ces trois méthodes retenues, notamment les principes généraux et les opérations sont précisées chaque.

2. Problématique de notre travail

Nous disposons d'un problème Job Shop que nous tentons à résoudre. Aussi, faut-il trouver la méthode la plus éfficace parmi trois méthodes métaheuristiques choisies : Algorithme Génétique, Recuit Simulé, Recherche Tabou-qui répondraient aux critères : makespan, temps d'attente totale, temps d'exécution et qui apporterait un meilleur rendement ?

3. Définition de métaheuristique

Le mot métaheuristique vient d'une combinaison de deux mots grecs : Heuristique vient du verbe heuriskein (euriskein), qui signifie « trouver », et META est un suffixe, qui signifie « au-delà », « à un niveau supérieur ». [D]

Selon la définition proposée par Osman, "les métaheuristiques sont des processus itératifs d'heuristiques subordonnées et d'amorçage qui combinent intelligemment plusieurs concepts pour explorer et exploiter l'ensemble de l'espace de recherche. Les stratégies d'apprentissage sont utilisées pour construire des informations afin de trouver efficacement la solution optimale ou proche de la meilleure. Solution. [I]

En résumant ces définitions, nous pouvons dire que les propriétés de base des métaheuristiques sont les suivantes.

_ Les métaheuristiques sont des stratégies qui guident la recherche de la meilleure solution ;

_ Le but des métaheuristiques est d'explorer efficacement l'espace de recherche pour déterminer des solutions (proche) optimales ;

_ Les techniques qui composent les métaheuristiques vont des simples processus de recherche locale aux processus d'apprentissage complexes ;

_ les métaheuristiques sont généralement non déterministes et ne garantissent pas l'optimalité ;

_ les méta-heuristiques peuvent contenir des mécanismes qui permettent d'éviter le blocage dans la région de l'espace de recherche ;

_ Les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, plutôt que d'être spécifiques à un problème spécifique ;

_ les méta-heuristiques peuvent invoquer des heuristiques prenant en compte la spécificité du problème traité, mais ces heuristiques sont contrôlées par des politiques de niveau supérieur.

4. Présentation des Métaheuristiques

Les problèmes d'optimisation combinatoire tels que le problème Job Shop sont pour la plupart NP-difficiles, et il n'existe actuellement aucune solution algorithmique efficace qui fonctionne sur toutes les données. L'utilisation d'approximations ou d'heuristiques est une alternative essentielle. Depuis deux décennies, les heuristiques les plus populaires, et aussi les plus efficaces [12], conçues pour résoudre des problèmes d'optimisation combinatoire, sont des techniques générales, appelées métaheuristiques, qui s'appliquent à chaque problème spécifique.

4.1. Concept métaheuristique

Les métaheuristiques sont une méthode générale principalement utilisée pour résoudre des problèmes d'optimisation combinatoire avec un temps de résolution en augmentation exponentielle.

"Les métaheuristiques sont des stratégies de recherche itératives avancées conçues pour explorer l'espace des solutions en utilisant différentes techniques. Plutôt que d'essayer d'examiner toutes les solutions possibles, ces approches trouvent des solutions satisfaisantes dans un délai raisonnable grâce à une investigation intelligente de l'espace basée sur deux principes : la diversité en recherchant tous les "lieux" depuis l'espace ; et l'intensification, en utilisant chaque point de l'espace. Territoire de l'État". [3]

Toutes les métaheuristiques sont basées sur un équilibre entre le renforcement de la recherche et la diversification. Le non maintien de cet équilibre peut conduire à une convergence trop rapide vers des minima locaux (manque de diversification) ou à un temps d'exploration trop long (manque d'intensification). [26]

4.2. Classification des métaheuristiques

En général, on distingue deux grandes catégories de métaheuristiques : les méthodes basées sur la population ou évolutionnaires, et les méthodes à point de recherche unique. Cette classification est basée sur le nombre de solutions opérées à chaque itération.

4.2.1. Les méthodes à base de population

Comme leur nom l'indique, ces méthodes ne peuvent gérer qu'un seul ensemble de solutions à la fois. Leur principe général est de regrouper des solutions pour former de nouvelles solutions et d'essayer d'hériter des bonnes propriétés de la solution mère. Ce processus est répété jusqu'à ce que les critères d'arrêt soient remplis. Dans les métaheuristiques basées sur la population, il existe deux grandes catégories : les algorithmes génétiques et les colonies de fourmis. [III]

4.2.2. Les méthodes de recherche locale

Elles sont également connues sous le nom de métaheuristiques de trajectoire ou de voisinage [3] est une méthode dont le principe général est de partir d'une seule solution x et d'essayer de l'améliorer à chaque étape de son avancement. L'ensemble des solutions qui peuvent être dérivées d'une solution x est appelé le voisinage de la solution N(x). La détermination d'une solution proche de x dépend bien sûr du problème traité.

Cette classe comprend plusieurs méthodes dont les plus importantes sont : Descente, Recuit Simulé, Recherche Tabou, Recherche Locale Itérative, GRASP (Recherche Adaptative Stochastique Greedy), Recherche Locale Guidée. Les méthodes de recuit simulé et tabou sont plus anciennes et sans doute plus populaires.

Il existe de nombreuses autres approches importantes qui combinent les avantages des deux paradigmes grâce à des méthodes d'hybridation. Parmi ces méthodes, nous avons MemeticAlgorithms (Genetic Local Search), ScatterSearch, GA/PM.

5. Les Algorithmes génétiques

Les algorithmes génétiques sont des méthodes permettant de trouver des solutions approchées basées sur des mécanismes (méthodes évolutives) inspirés des processus évolutifs naturels.

Les premiers travaux d'algorithmes génétiques ont commencé dans les années cinquante. Entre 1960 et 1970, John Holland a développé les principes de base des algorithmes génétiques, en s'appuyant sur des travaux antérieurs. [III]

Cette section présente le modèle de base de l'algorithme génétique, car il existe plusieurs variantes de la méthode impliquant des mécanismes plus complexes.

5.1. Les Principes généraux des algorithmes génétiques

Dans le contexte des problèmes d'optimisation combinatoire, un individu dans une population représente une solution parmi un ensemble de solutions possibles à un problème particulier. L'application d'un algorithme génétique à un problème spécifique nécessite les cinq éléments suivants :[28]

- **Le codage des solutions (individus)**

Associez chaque point de l'espace des solutions à une structure de données. Elle intervient généralement après la phase de modélisation mathématique du problème traité. La qualité de l'encodage détermine le succès de l'algorithme.

- **Une méthode de génération d'une population initiale**

La population initiale sur laquelle la descendance sera basée doit être non homogène.

- **Fonctionnalités à optimiser**

Cette fonction est utilisée pour évaluer chaque individu et elle renvoie une valeur réelle appelée "fitness" qui est utilisée pour calculer la probabilité que l'individu soit sélectionné.

- **Opérateurs génétiques**

La puissance des algorithmes génétiques réside dans leurs opérateurs génétiques (croisement et mutation). En effet, ces opérateurs peuvent construire des scans partiels de l'espace de recherche, réduisant leur temps de réponse.

Le croisement est considéré comme l'opérateur génétique le plus important.

Ce qu'il fait, c'est recombiner les gènes des individus d'une population pour balayer l'espace à la recherche de solutions.

Le deuxième opérateur génétique (mutation) est moins important que le premier opérateur. Il est utilisé pour introduire de petites perturbations au niveau de la population afin d'éviter une convergence prématurée de l'algorithme génétique vers un ou plusieurs optima locaux.

- **Algorithme général**

Les étapes d'un algorithme génétique peuvent varier d'un problème à l'autre. Cependant, ils sont basés sur le même principe. Le schéma suivant (Figure 3.1) illustre le principe de base de l'algorithme génétique.

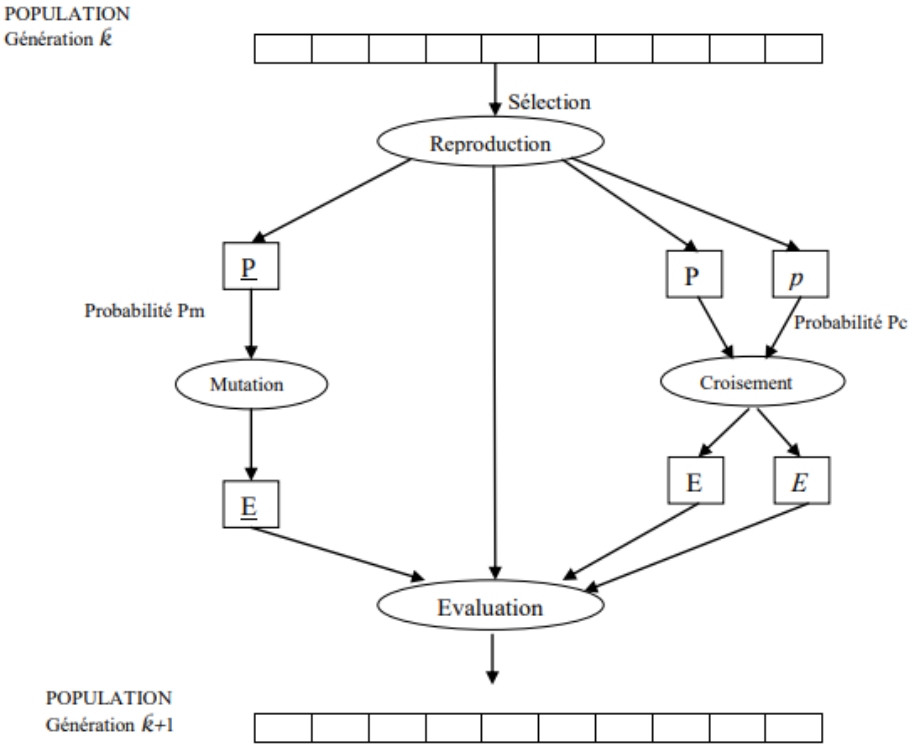


Figure 3.1 : Principe général des algorithmes génétiques

L'algorithme suivant résume le principe de base de l'algorithme génétique :

Algorithme Génétique ;

Début

Génération d'une population initiale de k individus ;

Répéter

1. Evaluation de la fonction objectif f de chaque individu ;
2. Sélection des meilleurs individus ;
3. Croisement entre deux individus sélectionnés : obtention de deux enfants issus de deux parents sélectionnés ;
4. Mutation : transformation aléatoire des gènes de certains individus de la population ;

Jusqu'à (condition d'arrêt)

Retourner la meilleure solution ;

Fin

Algorithme 3.1 : Principe de l'Algorithme Génétique.

5.2. Codage

Les algorithmes génétiques sont appliqués à des groupes d'individus, dont chacun est codé par un chromosome ou un génotype. Par conséquent, une population est représentée par un ensemble de chromosomes. Le processus d'encodage d'un individu consiste à trouver une structure de données pour l'individu. Il existe plusieurs types d'encodages tels que : l'encodage binaire, la permutation par valeurs entières, et par valeur, etc. [9][12]

5.2.1. Code binaire

Les premiers algorithmes génétiques utilisaient un codage binaire. Les chromosomes sont représentés par des chaînes binaires (chaînes de bits) qui spécifient les informations nécessaires pour décrire un individu.

5.2.2. Encodage Par valeur entière

Chaque gène est codé par une valeur entière. Par conséquent, les chromosomes sont représentés par une chaîne d'entiers. Ce codage est très adapté à l'optimisation de problèmes industriels réels.

5.2.3. Le codage par valeur

Dans ce type de codage (Figure 3.2), chaque gène est codé par une valeur qui appartient à un ensemble fini ou infini. Ces valeurs sont liées au problème à résoudre.

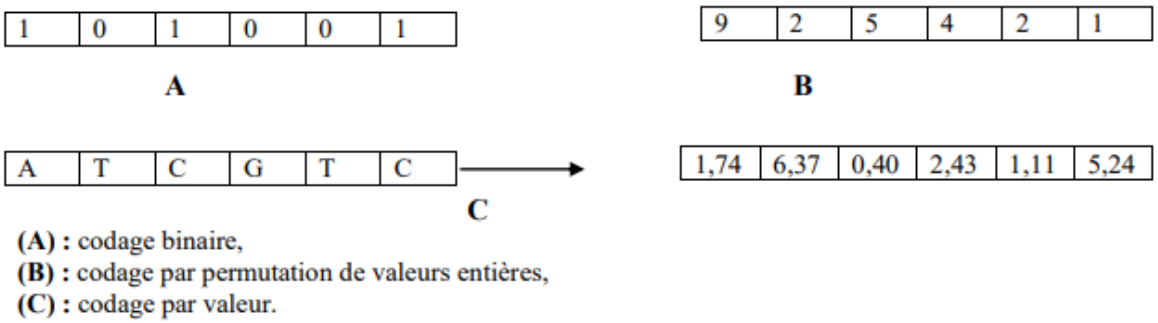


Figure 3.2 : Exemples de codage par valeurs

5.3. Les Opérateurs génétiques

Nous distinguons trois types d'opérateurs de reproduction génétique : les opérateurs de sélection, les opérateurs de croisement et les opérateurs de mutation.

5.3.1. Le croisement

L'opérateur de croisement est un opérateur binaire qui consiste à sélectionner une paire d'individus afin de recombiner les deux chromosomes sélectionnés, résultant en deux nouveaux chromosomes filles avec les meilleures caractéristiques des deux chromosomes parents. Il existe plusieurs stratégies croisées : [94]

• **Croisement à point unique** Dans ce type de croisement (Figure 3.3), on définit un point de coupure au niveau des deux chromosomes parents, puis chaque chromosome enfant est construit à partir de l'un des deux brins terminaux des deux chromosomes parents.

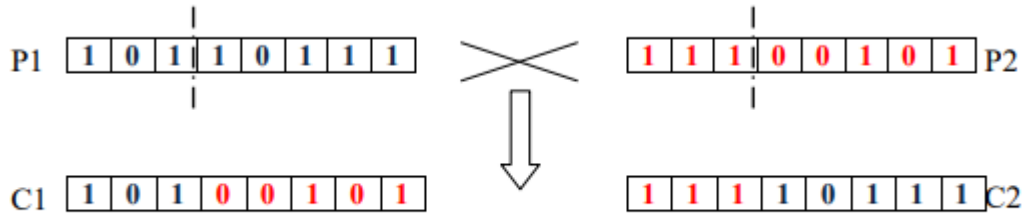


Figure 3.3 : Exemple d'un croisement à point simple

• **Croisement multi-point** Le croisement est dit multipoint, dans lequel la division des chromosomes peut se faire non seulement en 2 chaînes filles, mais aussi en 3 ou 4 chaînes filles, etc. Ainsi, ce croisement permet d'échanger plusieurs segments entre les deux chromosomes.

Un croisement à deux points (figure 3.4) est un cas particulier de croisement multipoints, dans lequel deux intersections sont choisies pour échanger deux segments des deux nœuds parents déterminés par les deux points.

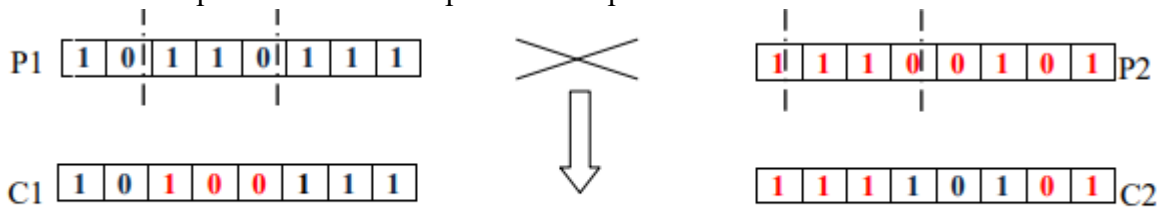


Figure 3.4 : Exemple d'un croisement bipoints

• **Croisement uniforme** Nous avons trouvé plusieurs méthodes liées à ce type de croisement (Figure 3.4), et parmi les méthodes les plus connues, celles qui utilisent des vecteurs masques. Le masque est un vecteur de la taille du chromosome qui contient une série de 0 et de 1 déterminée aléatoirement. Il est utilisé pour déterminer quel parent reprendra le gène

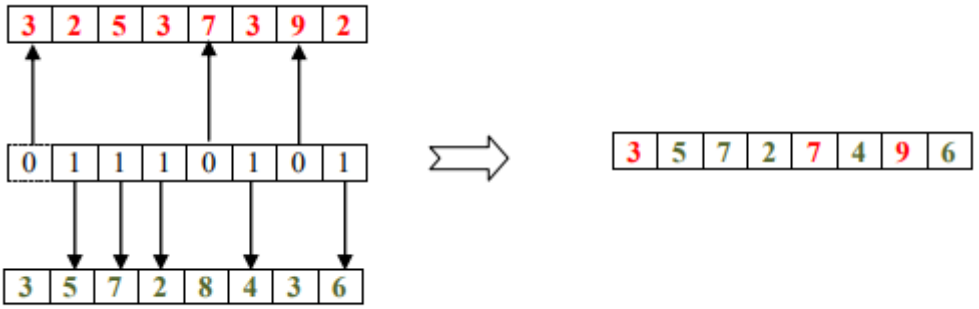


Figure 3.5 : Exemple de croisement uniforme

Dans les cas où les gènes sont soumis à certaines contraintes de codage, des croisements uniformes peuvent être utilisés (Figure 3.4), donc ces restrictions sont imposées aussi aux chromosomes enfants.

•**Le croisement multi-parental** Consiste à combiner les gènes de plus de deux parents en utilisant par exemple la méthode du simplexe pour orienter la recombinaison.

5.3.2. La Mutation

L'opérateur de mutation assure le brassage et la recombinaison des gènes parentaux, permettant une future diversification de la population pour éviter une convergence rapide vers des solutions optimales locales. L'opérateur est aléatoire et fonctionne selon une probabilité Pc déterminée par l'utilisateur en fonction du problème à optimiser. Il existe plusieurs stratégies de mutation : [28]

• **Mutation uni-point** Ce type de mutation se produit en modifiant une seule valeur sur un chromosome.

• **Mutations bipoints et multi points** Ces mutations se produisent en modifiant plusieurs valeurs sur le chromosome.

• **Mutation par valeur** La mutation de valeur se fait en convertissant une valeur donnée en une autre valeur déterminée sur tous les gènes du chromosome.

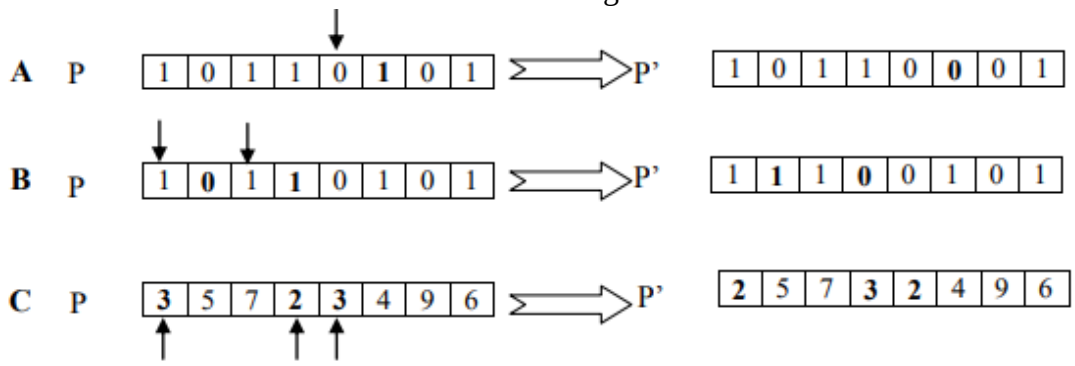


Figure 3.6 : Exemple d'opérateurs de mutation.

5.3.3. La Sélection

Le but de l'opérateur de sélection est de sélectionner des individus capables de survivre et/ou de se reproduire afin de transmettre leurs caractéristiques à la génération suivante. Le principe de la sélection est de garder les individus les plus aptes et d'éliminer les individus les moins aptes. Il existe plusieurs stratégies de sélection dont nous citons les plus importantes : [28]

- **La sélection par la roue de fortune (roulette Wheel)**

C'est une roue décomposée en parties (figure 3.7), chaque partie correspondant à un chromosome de population. La superficie de chaque secteur est liée à la valeur de la fonction objectif. Par conséquent, plus la forme physique d'un individu est grande, plus la superficie de son secteur est grande.

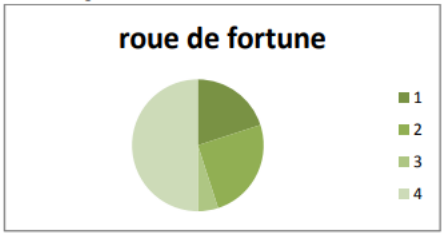


Figure 3.7 : Sélection par la « roue de fortune. [II]

Donc, la probabilité avec laquelle chaque individu sera sélectionné est :

$$P(i) = f(d(ci)) / \sum_{i=1}^{N} f(d(ci)) \; Equ. 0-1$$

Où, $((ci))$ est la fitness du chromosome i.

- **La sélection par rang (Ranking)**
  Dans cette stratégie de sélection, les individus d'une population sont classés de 1 à N selon leur fitness. Par conséquent, les pires chromosomes auront un rang de 1 et les meilleurs chromosomes auront un rang de N. On associe à chaque chromosome i une probabilité Pi d'être sélectionné, et la sélection sera proportionnelle à leur rang dans la liste de population :

$$Pi = Ri / \sum_{i=1}^{N} Ri \qquad Equ. 0-2$$

L'exemple suivant illustre la méthode de sélection par rang (Table 3.1) :

Chromosomes	**1**	**2**	**3**	**4**	**5**	**6**	**Totale**
Probabilités de fitness	87 %	7 %	5 %	2 %	1 %	94 %	100 %
Rangs	5	4	3	2	1	6	21
Probabilités de rang	24 %	19 %	15 %	5 %	9 %	29 %	100 %

Tableau 3.1 : Exemple de sélection par rang

Cette probabilité est ensuite comparée à un nombre aléatoire afin de prendre une décision. Le principe de cette technique est illustré dans l'algorithme suivant :

Algorithme de sélection par rang

Début
Classer les individus de la population courante selon leurs valeurs de fitness (ordre croissant),
Pour i = 1 To N faire
Calculer la probabilité de sélection Pi
Générer un nombre x appartenant à l'intervalle [0, 1]
Si x < Pi alors
Sélectionner individu i
Fin si
Fin pour
Fin

Algorithme 3.2 : Principe de l'Algorithme Génétique 2.

- **La sélection steady-stat(**l'état stable)
  Le principe de cette stratégie de sélection est de suivre les étapes suivantes au niveau de chaque génération :
  - Sélectionner quelques chromosomes parmi les chromosomes les plus coûteux pour créer des chromosomes filles (croisements et mutations) ;
  - Remplacer les pires chromosomes par de nouveaux ;
  - Les chromosomes restants apparaissent dans la nouvelle génération.

  Par conséquent, le principe de cette sélection est de sélectionner les meilleurs et d'exclure les pauvres, afin que les pauvres n'aient aucune chance de survie.
- **La Sélection par tournoi**
  La stratégie consiste à tirer des paires d'individus à partir d'une population de taille N. Au niveau de chaque paire, les individus seront sélectionnés en fonction de la probabilité dite la plus forte de gagner. Cette probabilité représente la chance de l'individu d'être sélectionné. Il existe d'autres types d'options de tournoi, telles que :le tournoi à trois, où la sélection se fait par trois individus dont un seul sortira vainqueur avec une certaine probabilité de victoire.[III]
- **L'élitisme**
  Certains des meilleurs chromosomes peuvent être perdus après l'application d'opérateurs génétiques de croisement et de mutation. Des stratégies élitistes sont utilisées pour éviter ce problème. Cela fonctionne en répliquant un ou plusieurs des meilleurs chromosomes d'une nouvelle génération, puis nous utilisons l'algorithme de réplication habituel pour générer le reste de la population.

5.4. Paramètres de dimensionnement

Il existe plusieurs paramètres prédéfinis qui jouent un rôle crucial dans la performance de chaque algorithme génétique. Parmi ces paramètres de base (on peut en trouver d'autres selon le problème auquel on est confronté) : [11]

- Taille de la population N : plus la valeur (taille) de N est grande, plus le temps de recherche est long. D'autre part, si la taille est trop petite, l'algorithme peut converger vers une mauvaise solution trop rapidement.
- Probabilité de croisement Pc : La valeur de cette probabilité reflète la force du changement appliqué à la population. Typiquement, la probabilité de croisement est comprise entre 0,5 et 0,9.
- Probabilité de mutation Pm : un rapport élevé de Pm risque de converger vers une solution sous-optimale et de perdre la population d'origine, de sorte que la probabilité de mutation est généralement faible.
- Le nombre de générations peut être défini comme une condition d'arrêt.

6. La recherche Tabou

Le terme recherche avec tabou ou simplement recherche tabou est apparu pour la première fois dans un article publié par Glover en 1986, dans lequel le terme métaheuristique apparaissait également.

6.1. Principes généraux de la méthode

La recherche tabou est une méthode de recherche locale. Il continue d'explorer la solution courante s, tous les voisinages N(s) à chaque itération. La meilleure solution s' pour ce

voisinage est conservée comme nouvelle solution, même si sa qualité est inférieure à celle de la solution actuelle s.

Cependant, cette stratégie peut conduire à des boucles, comme l'ordre de la solution : s1• s2• s1• s2. Détermine une boucle de longueur 2.

Pour éviter ce type de boucle, on mémorise les k configurations les plus récemment accédées en mémoire à court terme, et on interdit tout déplacement qui aboutit à l'une de ces configurations. Cette mémoire est appelée mémoire taboue ou liste taboue (nomme la méthode),est l'unedes éléments importants de cette méthode. Il évite toutes les boucles de longueur inférieure ou égale à k. La valeur de k dépend du problème à résoudre et peut changer au cours de la recherche.

En gardant une liste tabou, il est possible d'obtenir une meilleure solution avec un statut tabou. Dans ce cas, on peut encore accepter cette solution, en ignorant son tabou qui est d'appliquer le critère du désir.

Le principe de la recherche tabou est proposé par l'algorithme 3.2

Le principe de la Recherche Tabou est présenté par l'algorithme 3.2

Algorithme : Recherche Tabou ;

Début

1. Trouver une solution initiale s_0 , et poser § = s_0 , i = 0 et TL = f ; (TL : liste taboue)
2. Poser i = i + 1 ;
3. Déterminer le sous-ensemble de voisinage N*(si) ⊂ N(si) tel que :

$\forall s_{i+1} \in N^*(si)$: $(s_i, s_{i+1}) \notin TL$; ((s_i, s_{i+1}) : le mouvement de s_i à s_{i+1}) ;

4. Remplacer s_i par s_{i+1} tel que s_{i+1} est la meilleure solution de $N^*(s_i)$;
5. Mettre à jour TL ;
6. Si la condition d'arrêt n'est pas vérifiée, alors aller à 3 ;

Fin.

Algorithme 3.3 : Pseudo-code de la Recherche Tabou.

6.2. Les mécanismes principaux de la méthode

Dans la recherche tabou, le stockage des solutions récemment visitées dans une liste tabou pour éviter les boucles est le mécanisme de base de la méthode. Mais d'autres mécanismes sont également utilisés pour améliorer le processus de recherche. [10][11]

6.2.1. Les Listes tabous

Une liste tabou est une mémoire à court terme qui évite de revenir sur les solutions récemment visitées. Il est mis à jour à chaque itération. Stocker l'intégralité de la configuration est trop coûteux en temps de calcul et en espace mémoire, et peut ne pas être le plus efficace. Habituellement, seules les informations sur les caractéristiques ou le mouvement de la solution sont conservées dans cette liste, pas la configuration complète.

6.2.2. Redimensionnement de la liste des tabous

Dans certains cas, la longueur de la liste tabou doit être ajustée en fonction de la qualité du quartier. Si la liste est trop courte, la recherche finit par explorer des optima locaux avec un rayon légèrement plus grand. Les différentes solutions explorées forment une boucle

infinie. A l'inverse, si la liste est trop longue, toutes les actions deviennent taboues et les recherches sont bloquées. L'ajustement de la longueur de la liste dépend principalement de la topologie de l'espace des solutions et donc de la qualité du voisinage

6.2.3. Le critère d'aspiration

Bien que certaines solutions soient interdites, il existe encore des solutions que les gens veulent accepter. Ce mécanisme permet de lever la contre-indication de configuration sans introduire de risque de circulation au cours de l'étude. La fonction d'aspiration peut être définie de plusieurs manières. Le moyen le plus simple consiste à supprimer le statut tabou d'un déménagement si la qualité de la solution du déménagement est supérieure à la meilleure solution trouvée.

6.2.4. Détection de cycle

Parfois la recherche peut se coincer dans une boucle dont la longueur est supérieure à la taille de la liste taboue, ce qui ne permet pas de l'éviter. Par conséquent, la détection de ce cycle se fait en mémorisant l'historique de la solution grâce à la mémoire à long terme Recherche dans cet historique deux sous-chaînes qui ont la même configuration (même valeur, même longueur) et qui sont consécutives. Si une telle boucle est détectée, elle doit être évitée. Cette mémoire évite de rester dans une seule région de l'espace de recherche et étend la recherche à des régions plus intéressantes.

6.2.5. Liste de vérification des solutions d'élite

Permet de mémoriser des solutions de haute qualité afin de pouvoir les utiliser comme nouveau point de départ si votre recherche devient inefficace pendant plusieurs itérations successives.

L'algorithme et le mécanisme de la méthode de recherche tabou mentionnés ici ne sont qu'une introduction de base à la méthode. Évidemment, il existe d'autres mécanismes et d'autres versions plus complexes, généralement adaptées au problème traité.

7. Le Recuit Simulé

Le recuit simulé est l'une des plus anciennes méthodes de voisinage. Son succès est principalement dû aux résultats pratiques obtenus sur de nombreux problèmes NP-difficiles.

Le recuit simulé a été introduit par Kirk Patrick et al en 1983 et indépendamment par V. Cerny en 1985. Mais les origines de la méthode remontent aux expériences menées dans les années 1950 par Metropolis et al.

7.1. Le Principe général de la méthode

Le principe de la méthode s'inspire du procédé d'amélioration de la qualité des métaux solides en trouvant l'état d'énergie minimum correspondant à la structure stable du métal. L'état optimal correspondra à une structure moléculaire parfaitement régulière.

Débutant à une température élevée où le solide devient liquide, la phase de refroidissement fait revenir la matière liquide à sa forme solide en diminuant progressivement la température.Chaque température est maintenue jusqu'à ce que le matériau trouve l'équilibre thermodynamique.

L'application de ce principe à l'optimisation commence par la génération d'une solution initiale. A chaque itération, l'énergie de l'état (solution) est calculée et la température est

diminuée. Le passage de la solution actuelle à la nouvelle solution (ou même pire) se fait avec une certaine probabilité, en fonction de la nature des deux solutions et de l'avancement de la méthode.

7.2. Processus et composants de la méthode

Le processus de recuit simulé répète un processus itératif qui recherche des configurations à moindre coût tout en acceptant des configurations qui réduisent la fonction de coût de manière contrôlée.

Nous utilisons une approche stochastique pour générer une séquence d'états successifs du système à partir d'un état initial donné. Tout nouvel état est obtenu par déplacement aléatoire (perturbation) du système.

Soit : [III)

- $\Delta E = E_2 - E_1$: la différence d'énergie occasionnée par une telle perturbation,
- T : la température absolue du système,
- k : une constante physique appelée : constante de Boltzmann, $k = 1{,}380510^{-23}$ J/K;

Le nouvel état est accepté si l'énergie du système diminue ($\Delta E \leq 0$) ; sinon, il est accepté avec une probabilité définie par : $e^{-\Delta E/k}$

A chaque nouvelle itération, un voisin s'ϵ N(s) de la configuration courante s est généré de manière aléatoire. Selon les cas, ce voisin sera soit retenu pour remplacer celle-ci, soit rejeté.

- Si ce voisin est de performance supérieure ou égale à celle de la configuration courante, i.e. $f(s') \leq f(s)$, il est systématiquement retenu ;
- Dans le cas contraire, s'est accepté avec une probabilité p qui dépend de deux facteurs : - De l'importance de la dégradation $\Delta f = f(s') - f(s)$; les dégradations plus faibles sont plus facilement acceptées. [11]

- Dans le paramètre de contrôle T (température), une température élevée correspond à une plus grande probabilité d'accepter la dégradation.

La température est contrôlée par une fonction décroissante qui définit le mode de refroidissement. Le schéma de refroidissement de la température est l'un des paramètres les plus difficiles à régler. Ce modèle est essentiel pour une mise en œuvre efficace.

Sans être exhaustif, on rencontre typiquement trois grandes catégories de graphes :

• Réduction étagée : La température est maintenue constante pendant un certain nombre d'itérations, ce qui entraîne une réduction étagée.

• Réduction continue : La température est modifiée à chaque itération.

• Réduction non monotone : La température diminue à chaque itération et augmente occasionnellement.

Une des difficultés de cette méthode est de déterminer certains paramètres : la valeur initiale de la température (T0) et le coefficient de chute de température (γ). Le réglage de ces paramètres est assez délicat et repose sur des essais. Certains utilisateurs de cet algorithme posent γ ϵ [0,85, 0,95]. Quant à la température initiale, celle-ci est déterminée empiriquement ou par des méthodes plus sophistiquées. [9]

L'algorithme 3.4 présente la version simplifiée du Recuit Simulé.

Algorithme : Recuit Simulé ;

Début

1. Déterminer la solution initiale s0 et la température initiale T_0 ;

2. Poser § = s_0 et Ti = T_0 ;

3. Calculer $f(s_0)$;

4. Répéter pour chaque itération i / i = 1,.,n (ou jusqu'à ce que T est proche de 0) :

a. Choisir s' ϵN(si) ;

b. Calculer $\Delta f = f(s') - f(si)$;

c. SiΔf < 0 Alors §= s_i ;

d. **Sinon**

Début

tirer p dans [0, 1] suivant une distribution uniforme ;

Si p $\leq e^{-(\Delta f/T)}$: Alors § = s_i ;

Sinon s_i est rejetée ;

Fin ;

e. Calculer T = γT ;

5. Retourner § .

Fin

Algorithme 3.4: Pseudo-code du Recuit Simulé.

8. Conclusion

Les Métaheuristiques sont représentées essentiellement par les méthodes de voisinage comme le Recuit Simulé et la Recherche Tabou, et les méthodes évolutives comme les Algorithmes Génétiques. Ces trois Métaheuristiques étaient et restent l'objet d'une recherche intense pour la résolution de divers problèmes NP-difficiles dont le problème d'ordonnancement de Job Shop fait partie. Elles ont révélé leur grande efficacité de fournir des solutions approchées de bonne qualité pour un grand nombre de problèmes.

La présentation générale menée au cours de ce chapitre sur les Algorithmes Génétiques, le Recuit Simulé et la Recherche Tabou portant sur leur définition et l'explication de leur principe de fonctionnement a permis de faire ressortir l'idée de base de chacune de ces Métaheuristiques.

Chapitre 4 Conception et implémentation du problème

1. Introduction

Dans ce chapitre, nous avons commencé par présenter les outils et environnement de développement que nous avons utilisé. Ensuite, on a présenté l'application. Après Nous avons procédé quelques exemples pour tester, ces tests sont réalisés sur des données sont changer à chaque fois (nombre de job, nombre de machine). Les tests sont réalisés avec les trois méthodes déjà vu dans le chapitre 3, avec l'algorithme génétique et recuit simulé et Recherche tabou. Enfin on a parlé sur les résultats obtenus avec une petite comparaison entre les trois méthodes.

2. Langage de programmation et environnement de développement

2.1.Environnement matériel

Pour développer l'application, nous utilisons un environnement matériel,

Un ordinateur ASUS avec les caractéristiques suivantes :

- Un processeur Intel(R) Core (TM) i5-3337U CPU @ 1.80GHz 1.80 GHz.
- Une mémoire de 8,00 Go.
- Système d'exploitation 64 bits, processeur x64.
- Windows 10 Professionnel.
- Disque dur SSD 256.

2.2. Environnement logiciel

Sélection de la langue de développement à l'aide de la plate-forme pour Windows 10 Professional, C'est du C# Sharp.

2.3.Le langage de programmation "C#/Sharp"

C# est un langage de programmation fortement typé et orienté objet dérivé de C et C++.Similaire au langage Java. Il est utilisé pour développer des applications Web, et Applications de bureau, services Web, commandes, widgets ou bibliothèque de classe.

En C#, une application est un ensemble de classes, dont l'une avoir une méthode Main, comme cela se fait en Java.

2.4. L'environnement Microsoft Visual Studio

Microsoft Visual Studio est un environnement de développement intégré (IDE) de Microsoft. Peut être utilisé pour développer des applications d'interface utilisateur de console et graphiques avec les applications Windows Forms, les sites Web, les applications Web et les services Web dans le code natif avec le code géré pour toutes les plates-formes prises en charge par Microsoft Windows, Windows Phone, Windows CE, NET Framework, NET Compact Framework et Microsoft Silver light.

Figure 4.1 : Visual studio 2022

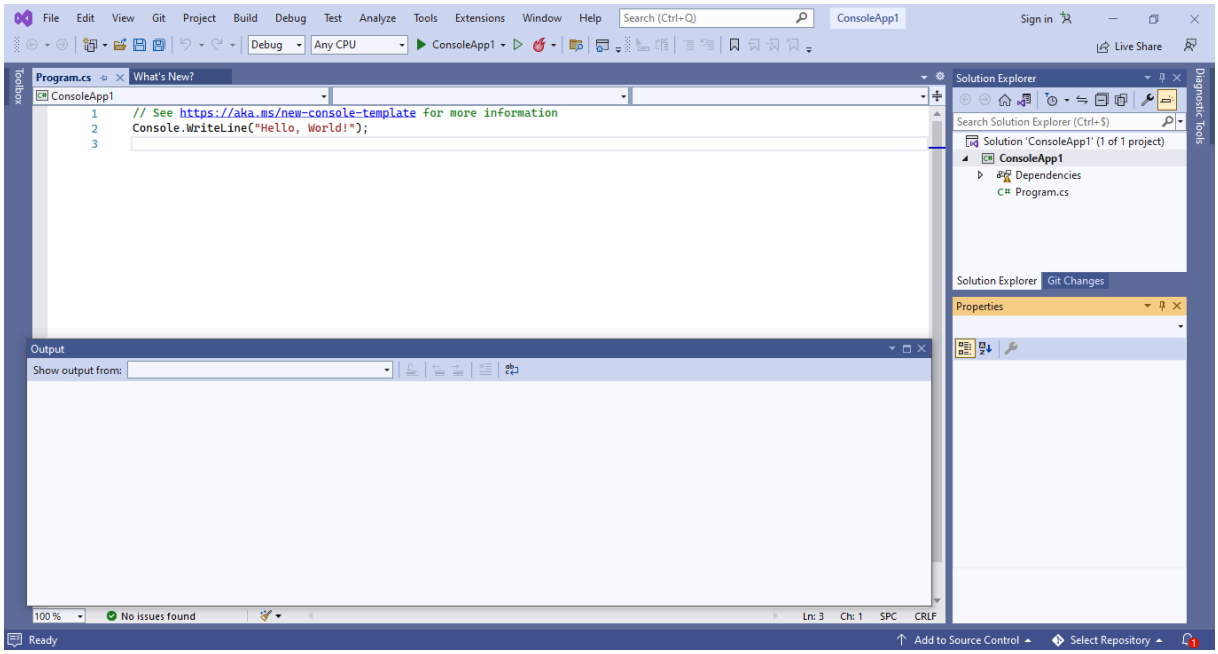


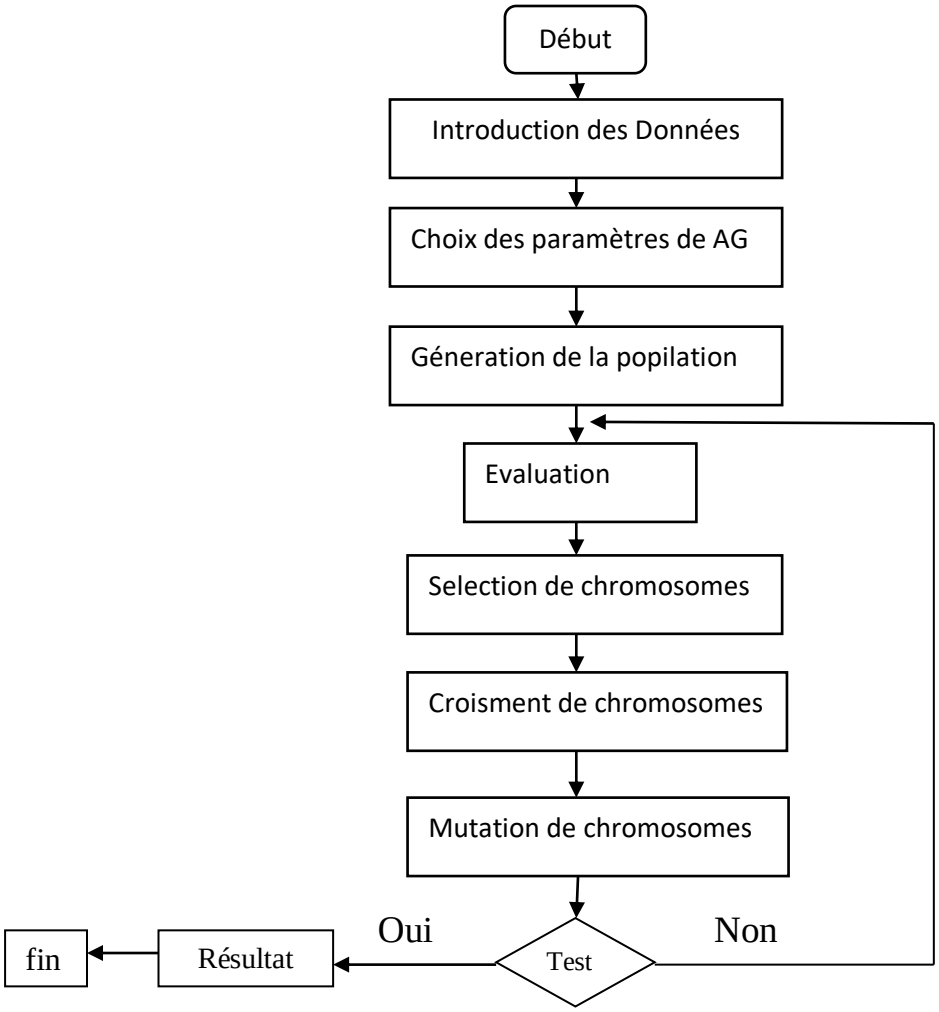
Figure 4.2 : interface Visual studio 2022.

3. Conception de l'application

La mise en œuvre de notre Algorithme Génétique passe par :

- L'implémentation de la représentation (codage) convenable au problème ;
- L'implémentation des opérateurs génétiques : croisement, mutation, sélection ;
- La détermination des différents paramètres de la méthode : taille de la population, nombre de jobs, nombre de machines, type de croisement, type de mutation,…

L'organigramme général de l'algorithme génétique implémenté est présenté ci-dessous



4. Illustration de l'application

En principe une seule application devrait exécuter les trois méthodes métaheuristiques. Mais vue les difficultés de rassembler ces trois méthodes en une application unique et commune, nous avons procédé à la réalisation de deux applications distinctes dans le but de résoudre le problème Job shop.

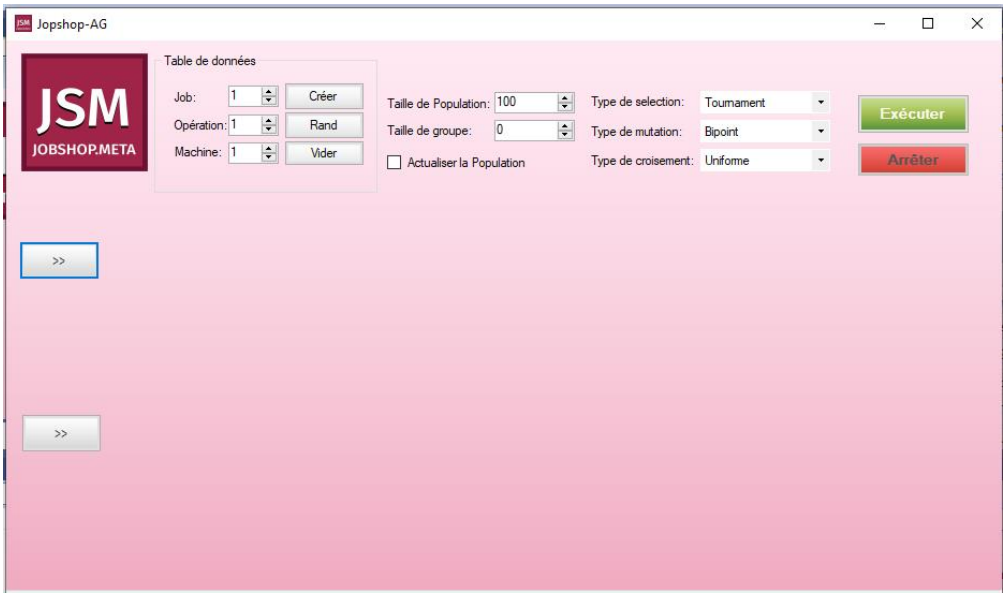


Figure 4.3 : interface principale de l'application (algorithme génétique)

5. Les résultats de quelques exemples sur l'implémentation

Pour le problème à étudier, normalement en utilise les benchmarks qui propose beaucoup d'exemple aléatoire qui seront analyser selon les trois méthodes choisis afin d'aboutir à des résultats qu'on va comparer par la suite. Cette comparaison sera selon méthodes : mathématique et statique.

On a appliqué la méthode statique pour les dizaines d'exemple choisi car la méthode mathématique est difficile à appliquer pour ces exemples.

5.1. Exemple 1

Dans le premier exemple nous avons 3 Jobs, 3 opérations et 3 machines :

Le tableau 4.1 présente un problème de type de job shop composé de 3 jobs et 3 machines. Dans lequel chaque tâche est étiquetée par une paire de nombres (m, p) où m est le numéro de la machine sur laquelle la tâche doit être traitée et p le temps de traitement de la tâche.

Job 1 = [(1, 30), (2, 30), (3, 30)].
Job 2 = [(1, 20), (2, 40), (3, 30)].
Job 3 = [(1, 20), (2, 30), (3,10)].

Dans l'exemple, le travail 1 comporte trois tâches. Le premier, (1, 30), doit être en Machine 1 à 30 unités de temps. Le deuxième (2, 30) doit manipuler 30 unités sur la machine 2. Et ainsi de suite. Il y a neuf missions au total.

Job	Numéro de machine	Temps de traitement
1	1	30
	2	30
	3	30
2	1	20
	2	40
	3	30
3	1	20
	2	30
	3	1

Tableau 4.1 : Exemple de problème job shop (3 machines, 3 jobs)

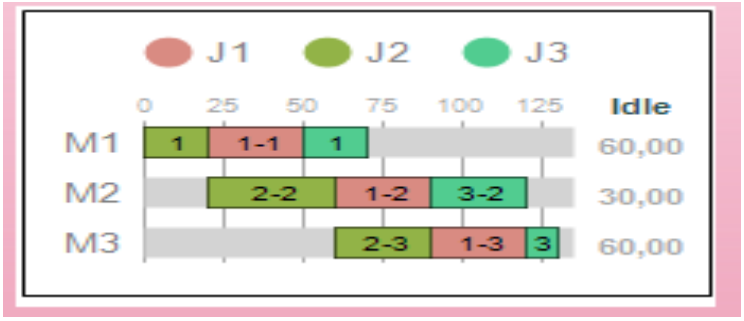


Figure 4.4 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 3x3

Makespan= 130
Temps d'attente totale= 150
Meilleur trouvé à = 00.30

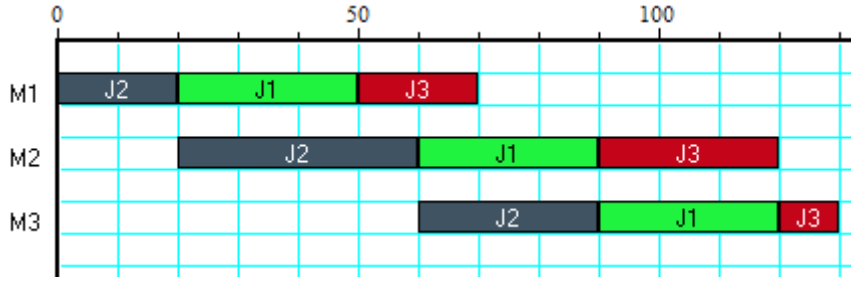


Figure 4.5 : Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 3x3

Makespan = 130
Temps d'attente totale = 150
Meilleur trouvé à = 00.33

Résultats obtenus par la Recherche tabou

Makespan = 130
Temps d'attente totale = 150
Meilleur trouvé à = 00.48

5.2.Exemple 2

Le deuxième exemple, contient 4 Jobs, 2 opérations et 2 machines :

Job 1 = [(1, 20), (2, 40)].
Job 2 = [(2, 10), (1, 30)].
Job 3 = [(1, 10), (2, 20)].
Job 4 = [(2, 30), (1, 40)].

Jobs	Numéro de machines	Temps de traitement
1	1	20
	2	40
2	2	10
	1	30
3	1	10
	2	20
4	2	30
	1	40

Tableau 4.2 : Exemple de problème job shop (2 machines, 4 jobs)

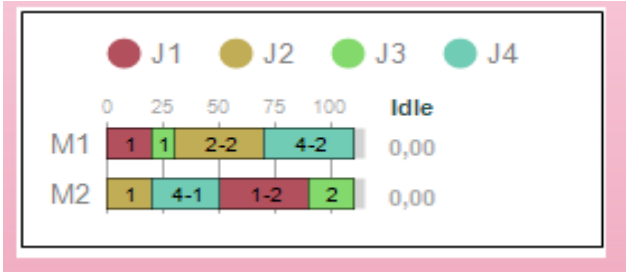


Figure 4.6 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 4x2

Makespan = 110

Temps d'attente totale = 0

Meilleur trouvé à =00.35

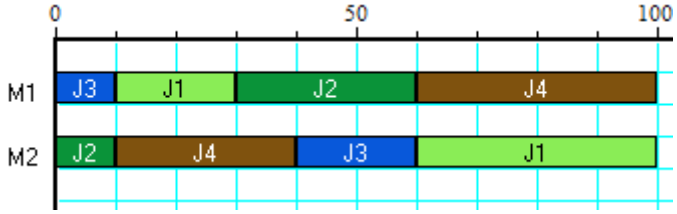


Figure 4.7 : Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 4x2

Makespan = 100

Temps d'attente totale = 100

Meilleur trouvé à =00.45

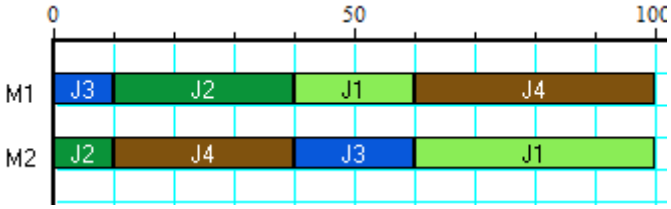


Figure 4.8 : Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 4x2

Makespan = 100

Temps d'attente totale = 100

Meilleur trouvé à =00.46

5.3.Exemple 3

Cet exemple constitué de 8 Jobs, 3 opérations et 3 machines :

Job 1 = [(1, 10), (2, 30), (3, 50)].
Job 2 = [(2, 20), (1, 40), (3, 30)].
Job 3 = [(3, 25), (2, 10), (1, 35)].
Job 4 = [(1, 30), (3, 42), (2, 15)].
Job 5 = [(3, 10), (1, 20), (2, 40)].
Job 6 = [(2, 30), (1, 50), (3, 20)].
Job 7 = [(1, 45), (3, 10), (2, 20)].
Job 8 = [(3, 10), (2, 30), (1, 20)].

Jobs	Numéro de machine	Temps de traitement
1	1	10
	2	30
	3	50
2	2	20
	1	40
	3	30
3	3	25
	2	10
	1	35
4	1	30
	3	42
	2	15
5	3	10
	1	20
	2	40
6	2	30
	1	50
	3	20
7	1	45
	3	10
	2	20
8	3	10
	2	30
	1	20

Tableau 4.3 : Exemple de problème job shop (3 machines, 8 jobs)

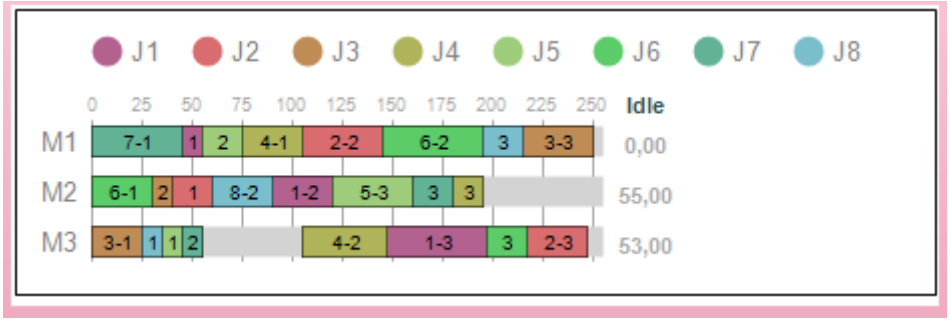


Figure 4.9 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 8x3

Makespan= 250

Temps d'attente totale = 108

Meilleur trouvé à = 01.45

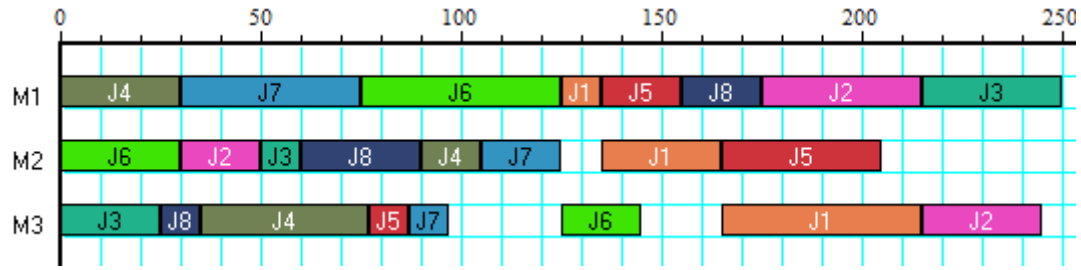


Figure 4.10 : Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 8x3

Makespan= 250

Temps d'attente totale = 250

Meilleur trouvé à = 00.37

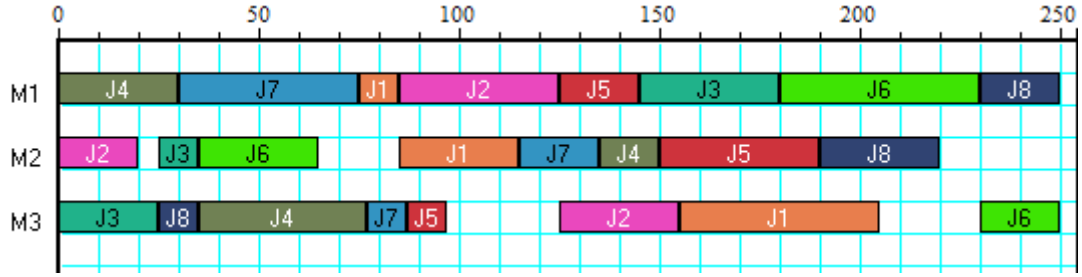


Figure 4.11 : Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 8x3

Makespan= 250

Temps d'attente totale = 250

Meilleur trouvé à = 00.36

5.4. Exemple 4

Dans cet exemple on a 5 jobs, 2 opérations et 2 machines :

J1= [(2,35) (1,40)]
J2= [(1,50) (2,25)]
J3= [(1,45) (2,10)]
J4= [(2,30) (1,40)]
J5= [(1,50) (2,20)]

Jobs	Numéro de machine	Temps de traitement
1	2	35
	1	40
2	1	50
	2	25
3	1	45
	2	10
4	2	30
	1	40
5	1	50
	2	20

Tableau 4.4 : Exemple de problème job shop (2 machines, 5 jobs)

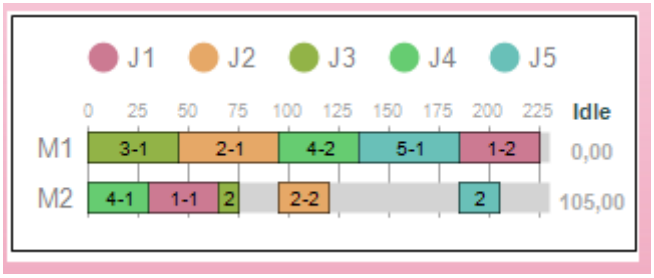


Figure 4.12 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 5x2

Makespan= 225

Temps d'attente totale =105

Meilleur trouvé à = 00.22

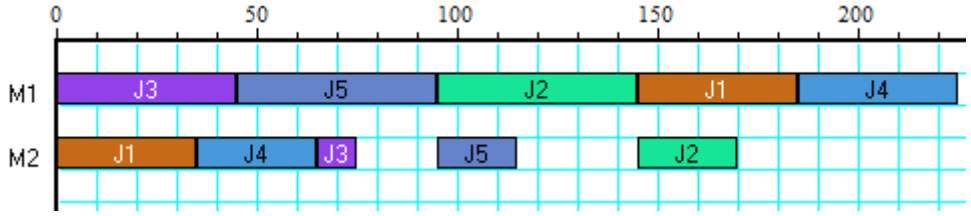


Figure 4.13 : Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 5x2

Makespan= 225

Temps d'attente totale = 225

Meilleur trouvé à = 00.21

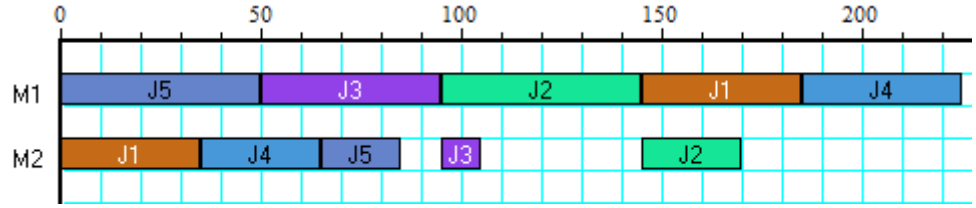


Figure 4.14 : Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 5x2

Makespan= 225

Temps d'attente totale = 265

Meilleur trouvé à = 00.24

5.5. Exemple 5

Dans le cinquième exemple nous avons 7 jobs, 4 machines et 3 opérations :

Job 1 = [(2, 25), (4, 40), (1, 60), (3,30)].
Job 2 = [(2, 25), (1, 75), (3, 20), (4,49)].
Job 3 = [(2, 30), (4, 42), (1, 15), (3,50)].
Job 4 = [(1, 35), (4, 37), (2, 40), (3,25)].
Job 5 = [(3, 43), (4, 40), (1, 50), (2,35)].
Job 6 = [(1, 50), (3, 41), (2, 10), (4,30)].
Job 7 = [(2, 29), (3, 23), (1, 32), (4,23)].

Jobs	Numéro de machine	Temps de traitement
1	2	25
	4	10
	1	60
	3	30
2	2	25
	1	75
	3	20
	4	49
3	2	30

	4	42
	1	15
	3	50
4	1	35
	4	37
	2	40
	3	25
5	3	43
	4	40
	1	50
	2	35
6	1	50
	3	41
	2	10
	4	30
7	2	29
	3	23
	1	32
	4	23

Tableau 4.5 : Exemple de problème job shop (4 machines, 7 jobs)

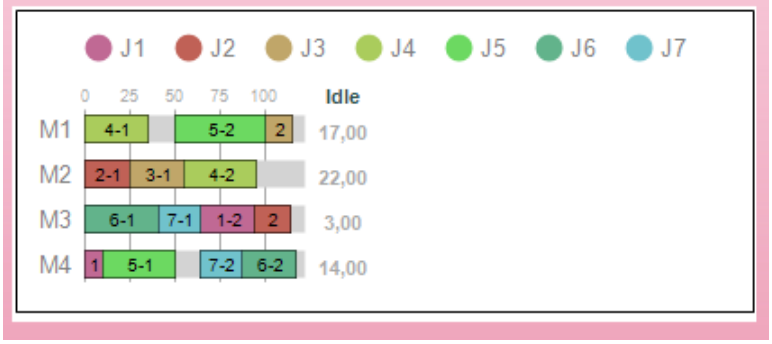


Figure 4.15 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 7x4

Makespan=114

Temps d'attente totale =44

Meilleur trouvé à = 00.93

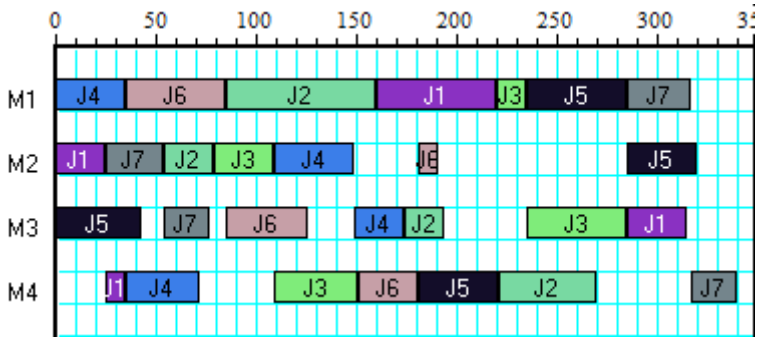


Figure 4.16 : Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 7x4

Makespan= 340

Temps d'attente totale =340

Meilleur trouvé à = 00.31

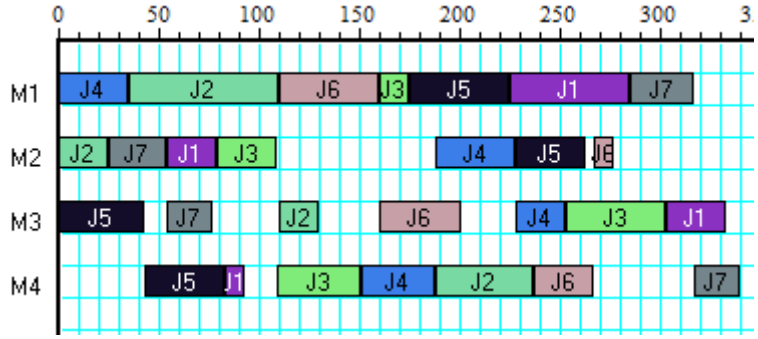


Figure 4.17: Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 7x4

Makespan= 340

Temps d'attente totale =363

Meilleur trouvé à = 00.32

5.6.Exemple 6

Nous avons 2 jobs, 3 machines et 3 opérations tel que :

J1= [(1,60), (2,30), (3,10)]

J2= [(2,40), (1,20), (3,50)]

Jobs	Numéro de machine	Temps de traitement
1	1	60
	2	30
	3	10
2	2	40
	1	20
	3	50

Tableau 4.6 : Exemple de problème job shop (3 machines, 2 jobs)

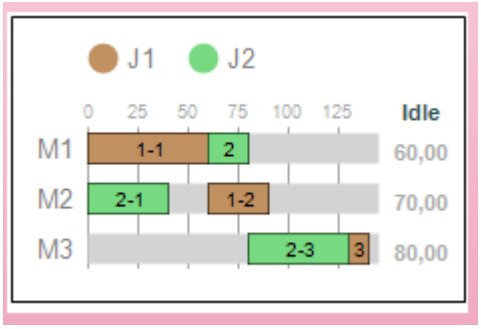


Figure 4.18 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 2x3

Makespan=140
Temps d'attente totale = 210
Meilleur trouvé à = 00.66

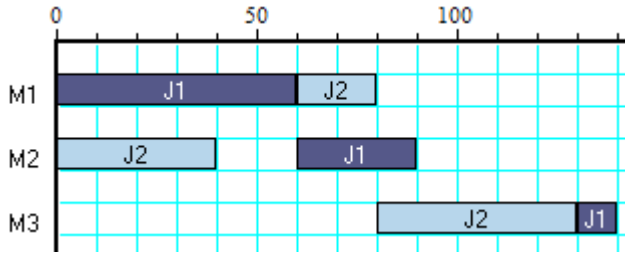


Figure 4.19 : Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 2x3

Makespan=140
Temps d'attente totale =140
Meilleur trouvé à = 00.36

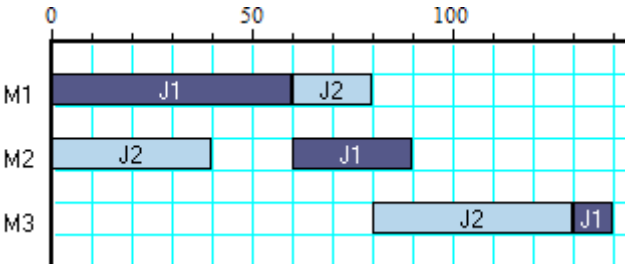


Figure 4.20: Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 2x3

Makespan=140

Temps d'attente totale =140

Meilleur trouvé à = 00.37

5.7.Exemple 7

Cet exemple contient 6 jobs, 2 machines et 2 opérations :

J1= [(2,45) (1,65)]
J2= [(1,15) (2,25)]
J3= [(1,29) (2,25)]
J4= [(1,52) (2,35)]
J5= [(2,45) (1,23)]
J6= [(1,67) (2,10)]

Jobs	Numéro de machine	Temps de traitement
1	2	45
	1	65
2	1	15
	2	25
3	1	29
	2	25
4	1	52
	2	35
5	2	45
	1	23
6	1	67
	2	10

Tableau 4.7: Exemple de problème job shop (2 machines, 6 jobs)

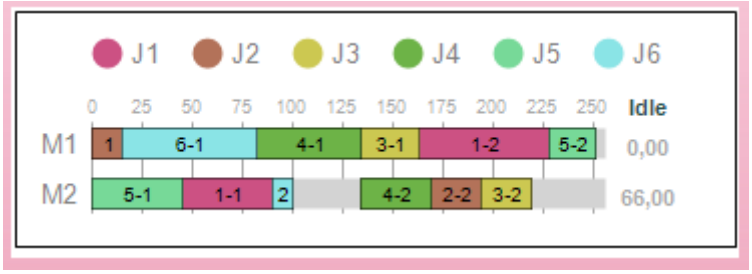


Figure 4.21 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 6x2

Makespan=251
Temps d'attente totale =66
Meilleur trouvé à = 00.37

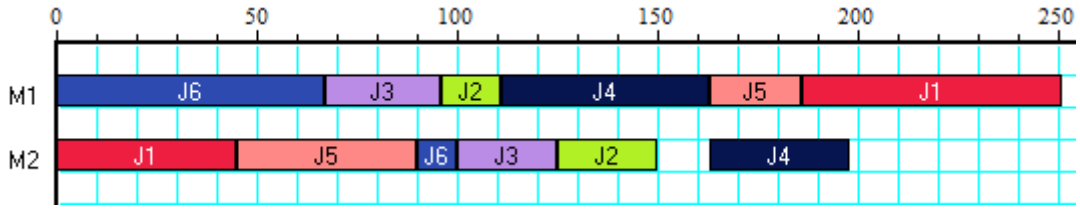


Figure 4.22: Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 6x2

Makespan=251
Temps d'attente totale =251
Meilleur trouvé à = 00.41

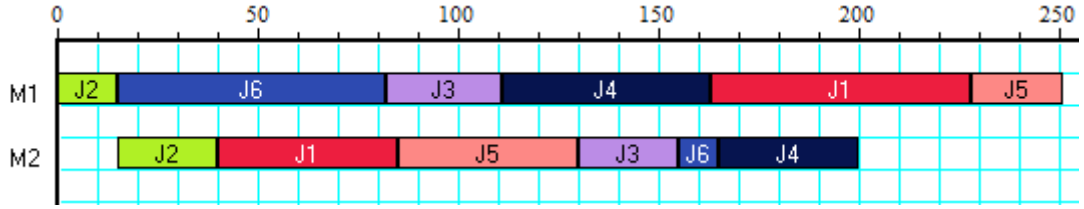


Figure 4.23: Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 6x2

Makespan=251
Temps d'attente totale =251
Meilleur trouvé à = 00.33

5.8. Exemple 8

Cet exemple constitué de 3 jobs, 4 machines et 2 opérations :

Job 1 = [(2, 74), (3, 36), (1, 61), (4,68)].
Job 2 = [(2, 54), (4, 58), (1, 31), (3,45)].
Job 3 = [(1, 81), (3, 15), (2, 20), (4,51)].

Jobs	Numéro de machine	Temps de traitement
1	2	74
	3	36
	1	61
	4	68
2	2	54
	4	58

	1	31
	3	45
3	1	81
	3	15
	2	20
	4	51

Tableau 4.8: Exemple de problème job shop (4 machines, 3 jobs)

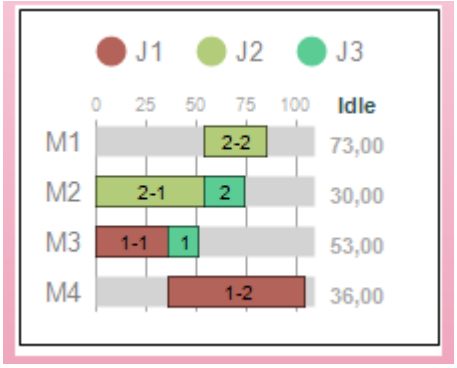


Figure 4.24 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 3x4

Makespan=104

Temps d'attente totale =192

Meilleur trouvé à = 00.59

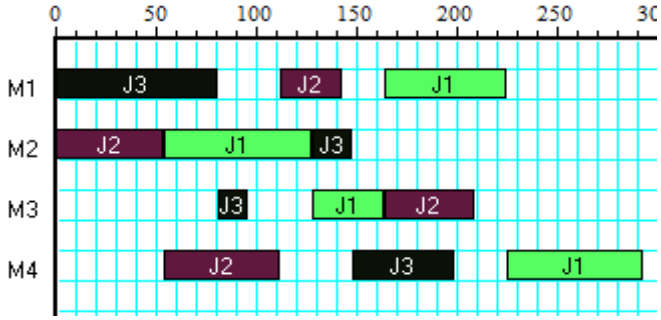


Figure 4.25: Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 3x4

Makespan=293

Temps d'attente totale =293

Meilleur trouvé à = 00.32

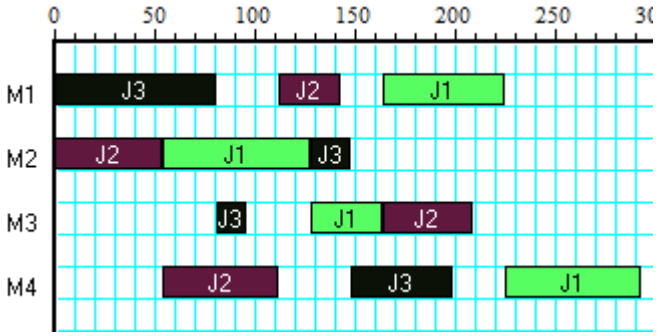


Figure 4.26: Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 3x4

Makespan=293

Temps d'attente totale =346

Meilleur trouvé à = 00.37

5.9. Exemple 9

Le neuvième exemple nous avons 10 jobs, 5 machines et 2 opérations :

J1= [(1,10), (2,25), (3,40), (4,15), (5,20)]

J2= [(2,30), (4,25), (1,10), (3,15), (5,35)]

J3= [(3,40), (5,45), (1,20), (2,10), (4,30)]

J4= [(4,20), (5,40), (1,15), (2,25), (3,35)]

J5= [(5,50), (1,42), (2,37), (3,28), (4,12)]

J6= [(3,66), (5,28), (1,56), (2,31), (4,42)]

J7= [(2,38), (4,23), (1,44), (3,60), (5,40)]

J8= [(5,10), (1,50), (2,30), (3,15), (4,25)]

J9= [(4,14), (5,20), (1,33), (2,52), (3,68)]

J10= [(1,30), (3,20), (5,10), (2,40), (4,50)]

Jobs	Numéro de machine	Temps de traitement
1	1	10
	2	25
	3	40
	4	15
	5	20
2	2	30
	4	25
	1	10
	3	15
	5	35
3	3	40
	5	45
	1	20
	2	10
	4	30
4	4	20
	5	40
	1	15
	2	25
	3	35
5	5	50
	1	42
	2	37
	3	28
	4	12
6	3	66
	5	28
	1	56
	2	31
	4	42
7	2	38

	4	23
	1	44
	3	60
	5	40
8	5	10
	1	50
	2	30
	3	15
	4	25
9	4	14
	5	20
	1	33
	2	52
	3	68
10	1	30
	3	20
	5	10
	2	40
	4	50

Tableau 4.9: Exemple de problème job shop (5 machines, 10 jobs)

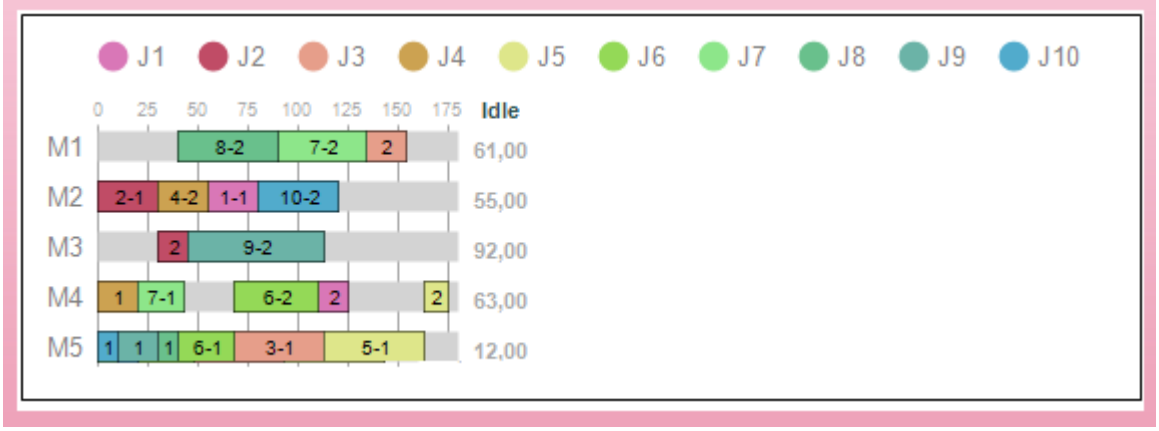


Figure 4.27 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 10x5

Makespan=115
Temps d'attente totale =69
Meilleur trouvé à = 00.44

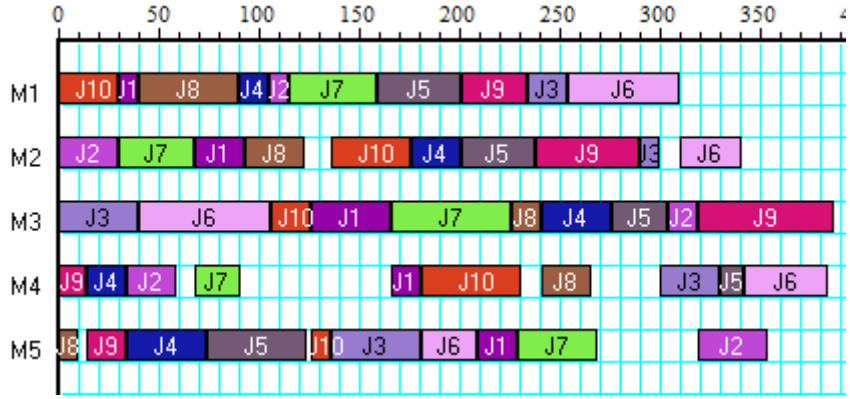


Figure 4.28: Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 10x5

Makespan=387
Temps d'attente totale =387
Meilleur trouvé à = 00.34

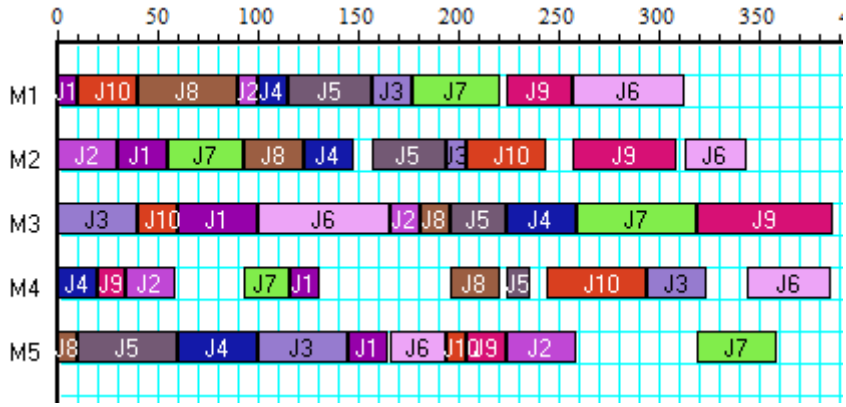


Figure 4.29: Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 10x5

Makespan=387
Temps d'attente totale =387
Meilleur trouvé à = 00.36

5.10. Exemple 10

Le dernier exemple contient 4 jobs, 3 machines et 3 opérations :

J1= [(1,15), (3,23), (2,40)]
J2= [(2,42), (3,10), (1,30)]
J3= [(3,25), (1,35), (2,50)]
J4= [(1,56), (2,25), (3,33)]

Jobs	Numéro de machine	Temps de traitement
1	1	15
	3	23
	2	40
2	2	42
	3	10
	1	30
3	3	25
	1	35
	2	50
4	1	56
	2	25
	3	33

Tableau 4.10:exemple de problème job shop (3 machines, 4 jobs)

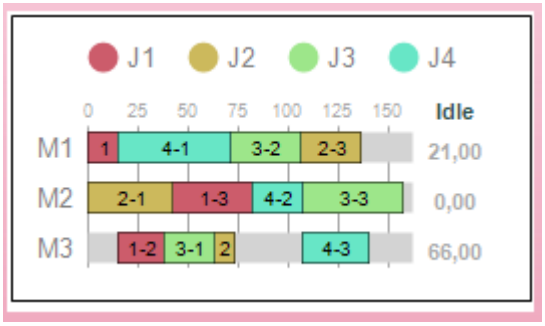


Figure 4.30 : Exemple de Diagramme de Gant obtenu par Algorithme Génétique de problème 4x3

Makespan=157
Temps d'attente totale =87
Meilleur trouvé à = 00.35

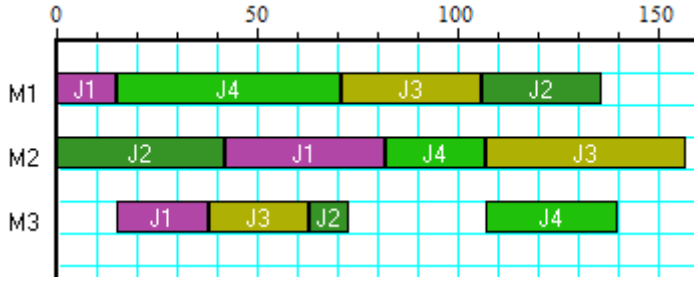


Figure 4.31: Exemple de Diagramme de Gant obtenu par Recuit simulé de problème 4x3

Makespan=157
Temps d'attente totale =157
Meilleur trouvé à = 00.33

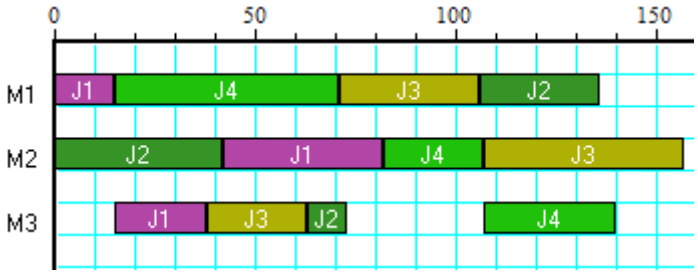


Figure 4.32: Exemple de Diagramme de Gant obtenu par Recherche tabou de problème 4x3

Makespan=157
Temps d'attente totale =219
Meilleur trouvé à = 00.35

6. Comparaison

Après les exemples qu'on a faits, nous avons trouvé défirent résultat d'algorithme génétique, recuit simulé et recherche tabou que nous avons implémenté sur le problème de job shop :

Méthode / Exemple	Algorithme génétique	Recuit simulé	Recherche tabou	**Meilleur Résultats**
1	Makespan = 130 T att = 150 M t à =00.30	Makespan = 130 T att = 150 M t à = 00.33	Makespan = 130 T att = 150 M t à = 00.48	Algorithme génétique
2	Makespan = 110 T att = 0 M t à = 00.35	Makespan = 100 T att = 100 M t à = 00.45	Makespan = 100 T att = 100 M t à = 00.46	Algorithme génétique
3	Makespan = 250 T att = 108 M t à = 01.45	Makespan = 250 T att = 250 M t à = 00.37	Makespan = 250 T att = 250 M t à = 00.36	Algorithme génétique
4	Makespan =225 T att = 105 M t à = 00.22	Makespan = 225 T att = 225 M t à = 00.21	Makespan = 225 T att = 265 M t à = 00.24	Algorithme génétique
5	Makespan = 114 T att = 44 M t à = 00.93	Makespan = 340 T att = 340 M t à = 00.31	Makespan = 340 T att = 363 M t à = 00.32	Algorithme génétique
6	Makespan = 104 T att = 210 M t à = 00.66	Makespan = 140 T att = 140 M t à = 00.36	Makespan = 140 T att = 140 M t à = 00.37	Algorithme génétique
7	Makespan = 251 T att = 66 M t à = 00.37	Makespan = 251 T att = 251 M t à = 00.41	Makespan = 251 T att = 251 M t à = 00.33	Algorithme génétique
8	Makespan = 104 T att = 192 M t à = 00.59	Makespan = 293 T att = 293 M t à = 00.32	Makespan = 293 T att =346 M t à = 00.37	Algorithme génétique
9	Makespan = 115 T att = 69 M t à = 00.44	Makespan = 387 T att = 387 M t à = 00.34	Makespan = 387 T att = 387 M t à = 00.36	Algorithme génétique
10	Makespan = 157 T att = 87 M t à = 00.35	Makespan = 157 T att = 157 M t à = 00.33	Makespan = 157 T att = 219 M t à = 00.35	Algorithme génétique

Tableau 4.11 : comparaison des résultats
T att : temps d'attente
M t à : meilleur trouvée à

Nous n'avons pas remarqué beaucoup de différence entre eux :

- Dans le premier exemple on a trouvé les mêmes résultats de makespan et le même temps d'attendre totale mais les temps d'exécution différents, meilleur temps d'exécution est dans l'algorithme génétique.
- Le deuxième exemple le résultat de makespan trouvé dans l'algorithme génétique est supérieure que les résultats de recuit simulé et de recherche taboue, mais le temps d'exécution est inférieur que les autres et il n'y a pas de temps d'attente.
- Le troisième exemple, nous constatons que les résultats de makespan dans l'algorithme génétique, recuit simulé et recherche tabou sont identiques, et le temps d'attente dans le recuit simulé et la recherche taboue est supérieur que l'algorithme génétique.
- Dans le quatrième exemple nous remarquons les mêmes résultats dans le makespan, la différence est dans le temps d'attente totale et le temps d'exécution. Les meilleurs résultats sont les résultats de l'algorithme génétique.

- Le cinquième exemple, le makespan dans l'algorithme génétique ayant une valeur plus petite par rapport aux autres méthodes, ainsi que le temps d'attente est le plus petit.
- Le sixième exemple, l'algorithme génétique nous a donné le meilleur résultat.
- Dans l'exemple 7, le makespan est identique dans les trois méthodes, mais le temps d'attente dans l'algorithme génétique est inférieur que les résultats de recuit simulé et recherche tabou.
- Les meilleurs résultats sont les résultats donnés par l'algorithme génétique dans l'exemple 8 par rapport aux autres méthodes.
- Le neuvième exemple, on a remarqué que les résultats de l'algorithme génétique sont inférieurs que les résultats de recuit simulé et recherche tabou.
- Dans le dernier exemple, le makespan est en accord dans les trois méthodes, donc la déférence est dans le temps d'attente totale et le temps d'exécution.

Alors l'algorithme génétique surpasse le recuit simulé et la recherche taboue.

Nous concluons que les trois méthodes ont la capacité de résoudre ces problèmes.

7. Conclusion

Dans ce chapitre nous avons présenté le logiciel développé, On a commencé par présenter l'environnement matériel et logiciel de travail réaliser, ensuit nous avons présenté notre étude comparative entre l'algorithme génétique, recuit simulé et recherche tabou après nous avons discuté sur les résultats obtenus.

Conclusion générale

Le problème d'ordonnancement existe dans tous les secteurs économiques et C'est une fonction importante dans la gestion de la production. Un problème d'ordonnancement consiste à attribuer des tâches dans le temps à celles qui existent dans un nombre limité, tout en respectant un ensemble de contraintes. Le problème d'ordonnancement des ateliers de type Job Shop est l'un des problèmes calendrier de recherche étendu. C'est un ordonnancement extrêmement complexe. Il est Classé comme problème combinatoire dur au sens fort. Cette complexité est due à La combinaison du nombre de solutions explose, augmentant de manière exponentielle avec la taille de la solution d'ordonnancement. L'utilisation de méthodes exactes pour obtenir des solutions optimales semble Pas réaliste. L'utilisation de méthodes d'approximation telles que l'heuristique est devenue inévitable. Parmi ces méthodes, le paradigme métaheuristique se distingue démarche très prometteuse.

Outre l'adaptabilité des métaheuristiques à différents problèmes combinatoires, A l'avantage de ne parcourir qu'une petite partie de l'espace des solutions pour obtenir une solution acceptable.

Dans ce travail, nous nous intéressons à l'application de métaheuristiques à Problème de planification de l'atelier de travail Job shop. Les méthodes utilisées sont : Algorithme Génétique, Recherche tabou et recuit simulé. Le cadre théorique de notre sujet comprend trois aspects qui le proposent : les problèmes généraux d'ordonnancement, les problèmes Ateliers de type job shop et ordonnancement méta-heuristique. La seconde partie le travail s'efforce de mettre en œuvre les trois approches conformément aux principes énoncés par les Nations unies plusieurs chercheurs. Des tests numériques sont ensuite effectués pour évaluer et comparer les performances des différentes méthodes mises en place et vérifiées pour notre application.

Notre étude malgré sa simplicité, elle nous a permis de dire que les méthodes approchées ne sont pas toujours mauvaises. Elles présentent, dans certains cas de meilleures solutions.

Nous pouvons également observer que trois métaheuristiques sont appliquées à résoudre les problèmes complexes de la planification des ateliers job shop peut on peut conclure que l'une des méthodes est meilleure que les autres.

Enfin, notre travail nécessite de fusionner les deux applications pour obtenir une application efficace pour résoudre n'importe quel problème de type job shop avec les trois métaheuristiques (algorithme génétique, recherche taboue et recuit simulé)à la fois.

Références bibliographiques

[1] Baptiste P., une étude théorique et expérimentale de la propagation des contraintes de ressources, Thèse de Doctorat, Université de Technologie de Compiègne, 1998.

[2] Brinkk•tter W. & Brucker P., solving open benchmark for the job shop, Journal of Scheduling, vol. 4 (2001), pp. 53-64

[3] Blum C. & Andrea R., Metaheuristics in Combinatorial Optimization: Overview And Conceptual Comparison, ACM Computing Surveys, vol. 35, n°. 3, September 2003, pp. 268-308.

[4] Blondel F., Gestion de la production, 3ème édition, DUNOD, Paris 2004.

[5] D. Duvivier, Ph. Preux, C. Fonlupt, D. Robilliard, E-G. Talbi, The fitness function and its impact on Local Search Methods, IEEE Systems, Man. and Cybernetics (IEEE SMC'98), pp. 2478-2483, San Diego, USA, 1998.

[6] Duvivier D. Etude de l'hybridation des méta-heuristiques, Application à un problème d'ordonnancement de type jobshop, Thèse de Doctorat, Université du Littoral Côte d'Opale, LIL, Calais 2000.

[7] D. Taillard, Principes d'implémentation des métaheuristiques, Chapter 2 of J. Teghem, M. Pirlot (dir.), Optimisation approchée en recherche opérationnelle, Hermès, 2002, pp. 57-79.

[8] French S., Sequencing and scheduling: An introduction to the mathematics of the Job Shop, Wiley, New York 1982.

[9] Fontanili F., Intégration d'outils de simulation et d'optimisation pour le pilotage d'une ligne d'assemblage multiproduit à transfert asynchrone, thèse de doctorat, Université Paris XIII, 1999.

[10] Giard V., Gestion de la Production, 2ème édition, Economica, Paris, 1988.

[11] Hentous H., contribution au pilotage des systèmes de production de type Job Shop, Thèse de Doctorat, INSA Lyon, 1999.

[12] Hurink, J.L. &Knust, S. Tabu search algorithms for Job-Shop problems with a single transport robot. European journal of operational research, vol. 162 (1), 2005

[13] J. Carlier et Ph. Chrétienne, *Problèmes d'ordonnancement : modélisation, complexité, algorithmes*, Masson, Paris, 1988.

[14] Jain, A. S., A Multi-Level Hybrid Framework for the Deterministic Job-Shop Scheduling Problem, PhD. Thesis, Dept. of APEME, University of Dundee, Scotland, UK, October 1998

[15] Jain, A.S. &Meeran, S., A State-of-the-Art Review of Job-Shop Scheduling Techniques, European Journal of Operations Research, vol. 113, pp. 390-434, Elsevier 1999.

[16] Javel G., Organisation et Gestion de la Production, 3ème édition, DUNOD, Paris 2004.

[17] Joseph Y-T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapman & Hall/CRC Computer & Information Science Series, 2004.

[18] Jeffrey W. Herrman, & al. Handbook Of Production Scheduling, Springer NY, 2006.

[19] Lopez P. & Esquirol P. L'ordonnancement, Economica, Paris 1999.

[20] Letouzey A., Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs, Thèse de Doctorat, Institut National Polytechnique de Toulouse, 2001.

[21] Penz B., Constructions agrégatives d'ordonnancements pour des Job-Shops statiques, dynamiques et réactifs, Thèse de Doctorat, Université Joseph Fourier - Grenoble I, 1994.

[22] Pinedo, M.: Scheduling Theory, Algorithms and Systems. Prentice-Hall, Inc. New Jersey, 1995.

[23] P. Esquirol et P. Lopez, *L'ordonnancement* ,Economica, Paris, 1999.

[24] Philippe Baptiste, Emmanuel Néron, Francis Sourd, *Modèles et algorithmes en ordonnancement*, Ellipses, Paris, 2004.

[25] Tanaev V.S, Sotskov Y.N. &Strusevich V. A., Scheduling Theory, Multi-Stage Systems, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1994.

[26] T'kindt V. &Billaut J.-C., Multicriteria Scheduling Theory, Models and Algorithms, Translated from French by H. Scott, Second Edition, Springer-Verlag Berlin 2006.

[27] VACHER J. Ph., Un système adaptatif par agents avec utilisation des algorithmes génétiques multi-objectifs : Application à l'ordonnancement d'atelier de type job-shop N×M, Thèse de Doctorat, Université du Havre, 2000.

[28] [V•z & Whi, 00] M. V•zquez, L. Darrell Whitley, A Comparison of Genetic Algorithms for the Static Job Shop Scheduling Problem, Source Lecture Notes In Computer Science, vol. 1917, Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, pp. 303-312, Publisher Springer-Verlag London, 2000.

Thèses et mémoires

[I] Ahm2clic

[II] Mr. CHERGUI Abderrahman, Mr. DAHMANI Abd Naceur, Mr. ADDA ABBOU Abdellah mémoire de Master en Génie industriel Université Abou BekrBelkaid – Tlemcen. Spécialité : Productique

[III] Mr. KEBABLA Mebarek Ingénieur en Informatique Université de Batna. Spécialité : Génie Industriel

[IV] Mr. : Mohammed El-Amine Meziane Diplôme de Doctorat en science. Spécialité : Informatique université Ahmed Ben Bella - Oran

Les sites web

[A] Site web : https://www.journaldunet.fr/business/dictionnaire-economique-et-financier/1198795-production.definition/#:~:text=D%C3%A9finition%20de%20production&text=La%20production%20est%20une%20activit%C3%A9,'autres%20entreprises%20puis%20transform%C3%A9s)

[B] Site web: https://www.clicours.com/decomposition-du-systeme-de-production/

[C] Site web: https://fr.wikipedia.org/wiki/Gestion_de_la_production

[D] Site web: www.Wikipedia.com

الملخص

يتضمن هذا العمل تطبيق علم metaheuristics على مشاكل جدولة ورش العمل من نوع متجر العمل. الطرق المستخدمة هي: الخوارزمية الجينية، البحث التبو، التلدين المحاكى. مشكلة البحث هي جدولة ورشة عمل بسيطة من نوع ورشة العمل، مهام n، آلات m. الهدف ليس اقتراح تقنيات جديدة، ولكن تطبيقات الاستكشاف والتوضيح والمقارنة لخصائص metaheuristics.
للقيام بذلك، تم تنفيذ الكمبيوتر وفقًا لعدة طرق تم اعتمادها. أدى تنفيذ الطريقة إلى تطوير تطبيق الكمبيوتر. يتيح لك التطبيق إنشاء عدة سلاسل تجارب بخيارات مزروعة مختلفة.
الكلمات المفتاحية: الجدولة، محل العمل، الاستنتاج، الخوارزمية الجينية، بحث الطابو، التلدين المحاكى.

Abstract

This work involves the application of metaheuristics to the scheduling problems of job shop type workshops. The methods used are: Genetic Algorithm, Tabu Search and Simulated Annealing. The research problem is the scheduling of the simple job shop type workshop, n tasks, m machines. The goal is not to propose new techniques, but applications of exploration, demonstration and comparison of standard metaheuristics.
To do this, a computer implementation according to several methods adopted the implementation of the method led to the development of a computer application. The application allows you to create several series Experiment with different implanted options.
Keywords: scheduling, job shop, metaheuristics, genetic algorithm, Tabu search, simulated annealing.

Résumé

Ce travail implique l'application de métaheuristiques aux problèmes d'ordonnancement des ateliers de type job shop. Les méthodes utilisées sont : Algorithme Génétique, Recherche Tabou et Recuit Simulé. La problématique de recherche est l'ordonnancement de l'atelier de type job shop simple, n tâches, m machines. Le but n'est pas de proposer de nouvelles techniques, mais des applications d'exploration, de démonstration et de comparaison de métaheuristiques standards.
Pour ce faire, une implémentation informatique selon plusieurs méthodes adoptées. La mise en œuvre de la méthode a conduit au développement d'une application informatique. L'application permet de réaliser plusieurs séries et expérimenter différentes options implantées.
Mots clés : ordonnancement, job shop, métaheuristique, algorithme génétique, recherche Tabou, recuit simulé.