الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية التكنولوجيا

قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Réseaux Informatiques et systèmes Répartis

Thème

Les algorithmes génétiques pour des problèmes d'ordonnancement de type Job Shop

Présenté par :

- BENYAMINA Hadj
- MEFTAH Mohamed

Dirigé par :

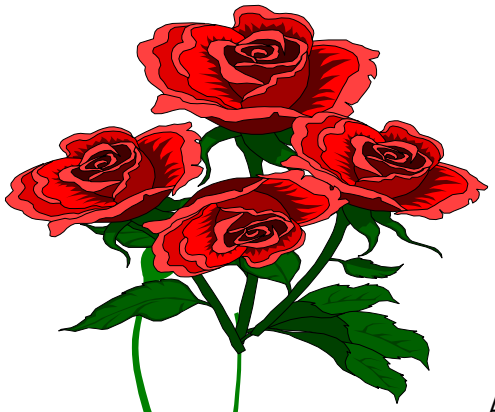
- Dr.MEKOUR Mansour

Année universitaire 2022-2023

Remerciements

Nous remercions, avant tout, Allah le tout puissant, de nous avoir guidé durant toutes nos années d'études, et nous avoir donné la volonté, la patience et le courage pour terminer notre travail.
Nombreuses sont les personnes qui nous ont aidé à franchir les obstacles et contraintes durant la préparation de ce travail, dont nous tenons à soulingner les contributions. Nous voudrons adresser nos remerciements plus particuliérement :
A notre promoteur le professeur Mansour Mekour, qui nous a encadrés pour réaliser ce mémoire. Nous lui reconnaissons son entiére disponibilité, son aide inestimable et ses conseils sans lesquels ce travail n’ourait pu aboutir.
A tous mes professeurs, sans exception, du primaire à l'université.Enfin, nos remercients les plus chaleureux à tous les membres de nos familles, qui nous ont toujours aidé encouragé au cours de la réalisation de ce mémoire.

Dédicace



Je dédie ce travail

A l'âme de ma mère

A mon père

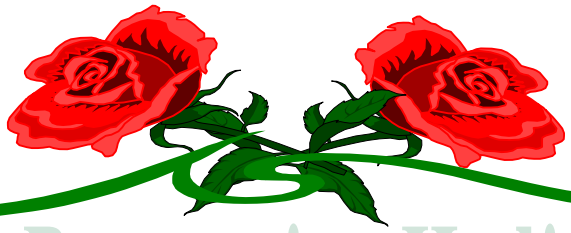
A mes frères

A mes amis

A ceux que j'aime

Et à toute personne qui a contribué à la

réalisation de ce mémoire .



Benyamina Hadj

Benyamina Hadj

Dédicace

Je dédie ce modeste travail:

A ma chère mère et à la mémoire de mon père,

Pour leur patience illimitée, leur encouragement continu, leur aide, en témoignage

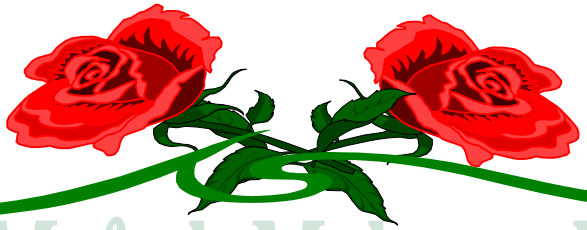
de mon profond amour et respect pour leurs grand sacrifices

A mes frères et sœurs

À mon chère binôme « benyamina hadj »

Et à tous ceux qui ont contribué de près ou de loin pour que ce travail soit

possible, je vous dis merci.



Meftah Mohamed

Meftah Mohamed

Contents

Liste des Figures

Liste des tableaux

Liste des algorithmes

Table des sigles et acronymes

SFP	Systèmes Flexibles de Production
PERT	Program Evaluation and Research Task
RDP	Réseaux de Pétri
JSP	Job-Shop Problem
IDE	Integrated Development Environment
JRE	Java Runtime Environment
JDK	Java Development Kit
XML	Extensible Markup Language
HTML	Hypertext Markup Language
PHP	Hypertext Preprocessor

Résumé

Face à la difficulté de la modélisation des processus, notamment l'exigence d'une prise de décision en temps réel, traiter les problèmes d'ordonnancement de manière précise s'est avéré inapproprié. C'est pourquoi, les chercheurs se sont orientés vers des méthodes heuristiques et méta-heuristiques qui ne garantissent pas toujours des résultats optimaux, dans les différentes situations, mais offrent le plus souvent des résultats très acceptables,en tenant compte de la fonction objectif, et fournit un meilleur partage de ressource dans un temps raisonnable.Dans ce mémoire, une application est développé, elle permet L'utilisation d'algorithmes génétiques pour minimiser le temps de fabrication du problème d'ordonnancement de type job shop. Cependant, le problème d'ordonnancement de type job shop dans la littérature est considéré comme un problème difficile dans le domaine de l'optimisation combinatoire, et sa complexité est NP-complète au sens fort. Notre objectif est de montrer la performance des algorithmes génétiques (métaheuristiques) dans la résolution de tels problèmes.

Mots clés : Système de production , Ordonnancement, Job-shop, Méta-heuristique, Algorithmes génétiques, Makespan.

Abstract

Faced with the difficulty of process modeling, in particular the requirement of real-time decision-making, dealing with scheduling problems in a precise manner has proven to be inappropriate. This is why researchers have turned to heuristic and metaheuristic methods which do not always guarantee optimal results, in different situations, but most often offer very acceptable results, taking into account the objective function, and provides better resource sharing in a reasonable time. In this thesis, an application is developed,it allows the use of genetic algorithms to minimize the manufacturing time of the job shop type scheduling problem. However, the job shop type scheduling problem in the literature is considered as a hard problem in the field of combinatorial optimization, and its complexity is NP-complete in the strong sense. Our objective is to show the performance of genetic algorithms (meta-heuristics) in solving such problems.

Keywords: Production system, Scheduling, Job-shop, Meta-heuristics, Genetic algorithms, Makespan.

ملخص

في مواجهة صعوبة نمذجة العملية ، لا سيما متطلبات اتخاذ القرار في الوقت الفعلي ، ثبت أن التعامل مع مشاكل الجدولة بطريقة دقيقة غير مناسب. هذا هو السبب في أن الباحثين تحولوا إلى طرق الكشف عن مجريات الأمور والميتاهوريستية التي لا تضمن دائماً النتائج المثلى ، في مواقف مختلفة ، ولكنها غالباً ما تقدم نتائج مقبولة للغاية ، مع مراعاة الوظيفة الموضوعية ، وتوفر مشاركة أفضل للموارد في وقت معقول. في هذه الأطروحة ، تم تطوير تطبيق يسمح باستخدام الخوارزميات الجينية لتقليل وقت التصنيع لمشكلة جدولة نوع متجر العمل. ومع ذلك ، تعتبر مشكلة جدولة متجر العمل في الأدبيات مشكلة صعبة في مجال التحسين الاندماجي ، وتعقيدها مكتمل ص بالمعنى القوي. هدفنا هو إظهار أداء الخوارزميات الجينية (متَهُرِستِس) في حل مثل هذه المشاكل.

الكلمات الرئيسية: نظام الإنتاج ، الجدولة ، محل العمل ، الاستدلال الفوقي ، الخوارزميات الجينية ، شكسن.

Introduction Générale

L'industrie actuelle se caractérise par une forte demande de produits personnalisés de bonne qualité, des prix bas et des délais de livraison de plus en plus courts. En effet, l'ouverture des marchés internationaux, ainsi que l'évolution et la mondialisation ont poussé les industriels vers des systèmes de fabrication de plus en plus flexibles, qui nécessitent de remettre en cause certaines habitudes de production, notamment la gestion de l'atelier de production, qui joue un rôle essentiel dans la productivité et la réduction des délais de production.

Un système de production est dit flexible s'il peut assurer la production simultanée de nombreuses quantités différentes de pièces et peut s'adapter à la production de nouveaux produits qui n'ont pas été étudiés par le système. La productivité est directement affectée par la qualité de la planification des opérations de la machine.car un atelier de production peut réaliser une grande variété de produit avec des coûts réduits, grâce à une meilleure utilisation des ressources. Le domaine d'application de l'ordonnancement est vaste : par exemple, la gestion de la charge des processus en informatique, la gestion de la production dans l'industrie, la gestion de projets, etc.

Le problème d'ordonnancement est un problème combinatoire fort, qui peut toujours être mis à jour, car il n'existe pas de méthode de résolution générale à l'heure actuelle, et la complexité de l'algorithme est faible. Il existe divers problèmes d'ordonnancement au regard de plusieurs paramètres : tâches ou opérations, paramètres liés aux ressources, types de contraintes de tâches, critères d'optimalité (minimisation du temps de réalisation total "makespan"). Chaque tâche consiste en une séquence d'opérations, chacune nécessitant une machine opérationnelle. Les décisions concernant la séquence des opérations de la machine doivent être optimisées pour améliorer les performances du système. Face à la difficulté de la modélisation des processus, notamment l'exigence d'une prise de décision en temps réel, traiter tous ces problèmes de manière précise s'est avéré inapproprié. C'est pourquoi, les chercheurs se sont orientés vers des méthodes heuristiques et méta-heuristiques qui ne garantissent pas toujours des résultats optimaux, dans les différentes situations, mais offrent le plus

souvent des résultats très acceptables, en tenant compte de la fonction objectif, et fournit un meilleur partage de ressource dans un temps raisonnable. C'est dans ce cadre que cet mémoire a été rédigé.

Une méthode d'optimisation est proposé, à l'aide d'algorithmes génétiques .Ces dernières peuvent être appliquées à des problèmes très divers car elles sont indépendantes du processus à optimiser et n'utilisent pas les dérivées.

le travail présenté dans ce mémoire traite un problèmes d'ordonnancement de type job shop.Pour résoudre ce problème une démarche est proposé. elle consiste à trouver un ordonnancement des jobs dont Le critère retenu ici est la minimisation de la durée totale de l'exécution des tâches, soit l'optimisation du Cmax (Makespan). Ce mémoire est organisé en 3 chapitres :

- Dans le premier chapitre, un présentation concise des systèmes de production et de la gestion de production ou apparaît le rôle de l'ordonnancement. La suite du chapitre est consacrée pour les différents types d'ateliers.Le chapitre se termine par la présentation des problèmes d'ordonnancement ainsi que les différentes méthodes de résolution qui existe dans la littérature.

- Le deuxième chapitre aborde le problème d'ordonnancement du job-shop on donnant la présentation de : le problème d'ordonnancement job shop et ses types avec des exemples .Dans la deuxième partie Une description complète du fonctionnement des algorithmes génétiques sera faite. avec la présentation des opérateurs génétiques participant à l'exploration de l'espace de recherche et les paramètres nécessaires pour la convergence vers de bonnes solutions.

- Dans le troisième et le dernier chapitre le problème d'ordonnancement d'atelier de type Job-Shop sera présenté puis la démarche de résolution proposée pour la problématique ainsi que les résultats obtenus après l'application sur plusieurs exemples, et en fin l'implémentation.

Chapitre 1

L'ordonnancement de la production

1.1 introduction

La gestion des systèmes de production, de biens ou de services pose de très nombreux problèmes touchant la gestion de production, le marketing, et des ressources humaines. La résolution de ces derniers nécessite l'utilisation de techniques d'optimisation et/ou d'évaluation de performances issues de domaines très variés. Ce chapitre est dédie à l'ordonnancement de la production et plus particulièrement aux problèmes d'atelier, où les ressources sont des machines et les tâches à ordonnancer sont des opérations à réaliser sur ces machines.

1.2 La production

1.2.1 Définition de la production

La production est le processus conduisant à la création de produits par l'utilisation et la transformation de ressources [01]. Le processus de production est alors, constitué d'un ensemble d'opération qui est les activités conduisant à la création de biens et de services. [02]

1.2.2 La gestion de production

Le rôle de la gestion de production

La gestion de production s'occupe d'un ensemble de problèmes liés à la production comme la planification, la gestion des données, le contrôle de la production, la prévision, la gestion des stocks, l'ordonnancement etc. [03].

La gestion de production est une fonction complexe. Cette complexité conduit généralement à l'hiérarchiser afin de la simplifier et de rendre possible la gestion du système. Les niveaux hiérarchiques de la gestion de production couramment retenus sont trois : stratégique, tactique et opérationnel.

- **Le niveau stratégique** : Il s'agit de la formulation de la politique à long terme de l'entreprise (à un horizon de plus de deux ans). Elle porte essentiellement sur la gestion des ressources durables, afin que celles-ci soient en mesure d'assurer la pérennité de l'entreprise.

- **Le niveau tactique** : Il s'agit des décisions à moyen terme. Elles assurent la liaison entre le niveau stratégique et le niveau opérationnel. L'objectif est de produire au moindre coût pour satisfaire la demande prévisible, en s'inscrivant le cadre fixé par le plan stratégique de l'entreprise.

- **Le niveau opérationnel** : Il s'agit des discisions à court et à très court terme. C'est une gestion quotidienne pour faire face à la demande au jour le jour, dans le respect des décisions tactiques.[04]

Les systèmes flexibles de production

a. Définition des Systèmes Flexibles de Production (SFP)

Un SFP est un système de production capable de produire différents types de pièces et qui est composé d'un ensemble de machines à commande numérique interalliées par un système automatisé de T/M. La gestion et le contrôle de ce système sont informatisés [05].

b. Décomposition du Système Flexibles de Production

Un SFP se décomposer en trois sous-systèmes :

1- Le système physique de production (le système de fabrication) : Transforme les matières premières ou composantes en produits finis. Ce système est composé d'un ensemble des machines à commande numérique capables de faire des changements autonomes d'outils, un système automatisé de transport/manutention/stockage (chariots filoguidés, tapis roulants, robots dédiés au déplacement des pièces. . .)

2- Le système de décision (le système de contrôle) : Contrôle le système physique de production. Il en coordonne et organise les activités en prenant des décisions basées sur les

données transmises par le système d'information.

3- Le système d'information : intervient à plusieurs niveaux :

- A l'interface entre les systèmes de décision et de production .

- A l'intérieur du système de décision, pour la gestion des informations utilisées lors de prises de décisions .

- A l'intérieur du système physique de production. Son rôle est de collecter, stocker et transmettre des informations de différents types. [06]

1.2.3 Rôle de l'ordonnancement en gestion de production

L'ordonnancement couvre un ensemble d'action qui transforment les décisions de fabrication définies par le programme directeur de production en instructions d'exécution détaillées destinées à contrôler et piloter à court terme l'activité des postes de travail. En sortie de la fonction ordonnancement, on obtient un planning ou ordonnancement, qui restitue l'affectation des tâches fournies en entrée à des dates précises pour des durées déterminées sur les différentes ressources. Ce planning cherche à satisfaire des objectifs, en respectant le plus possible les contraintes imposées.

1.3 L'ordonnancement

1.3.1 Définition

L'ordonnancement est une branche de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. On peut rencontrer les problèmes d'ordonnancement dans de très nombreux domaines : les systèmes industriels de production (activités des ateliers en gestion de production et problèmes de logistique), les systèmes informatiques (les tâches sont les programmes et les ressources sont les processeurs, la mémoire...), les systèmes administratifs (gestion du personnel, emplois du temps,...), les systèmes de transport, la construction, ... etc. Les différentes données d'un problème d'ordonnancement sont les tâches, les ressources, et les contraintes. Dans le système de production, un problème d'ordonnancement se définit comme étant la localisation dans le temps et l'espace, la réalisation d'un ensemble de tâches, compte tenu des contraintes

temporelles (une date de début avec une durée ou une date de fin) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches.

1.3.2 Les types (classes) d'ordonnancement

Plusieurs classes d'ordonnancement sont distinguées

- Ordonnancement sans délai : un ordonnancement O est dit sans délai lorsqu'aucune machine n'est restée inoccupée alors que dans le stock d'entrée se trouve une tâche en attente.

- Ordonnancement actif : un ordonnancement O est dit actif si tout décalage à gauche oblige à retarder l'exécution d'une autre opération ou à violer une contrainte.

- Ordonnancement semi actif : Un ordonnancement O est dit semi-actif si on ne peut décaler à gauche aucune opération sans modifier l'ordre de traitement des jobs sur les ressources (machines) ce qui oblige à retarder l'exécution d'une autre opération ou à violer une contrainte.

1.3.3 Les éléments d'un problème d'ordonnancement

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

Les taches

Une tâche est une entité élémentaire organisée dans le temps, par une date de début et/ou de fin, et dont la réalisation nécessite une durée préalablement définie. Elle est constituée d'un ensemble d'opérations qui requiert, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif. On distingue deux types de tâches :

- Les tâches morcelables (préemptives) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.

- Les tâches non morcelables (indivisibles) qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

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

$p_i = c_i - t_i$.

Certaines caractéristiques relatives à l'exécution d'une tâche sont définies ainsi :

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

La figure 1.1 donne une réprésentation de la tâche en désignant ses principales caractéristiques :

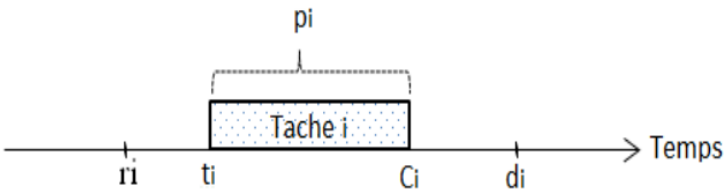


Figure 1.1: Représentation de la tâche

Les resources

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources :

- Les ressources renouvelables :Qui, après avoir été allouées à une tâche, redeviennent disponibles (machines, personnel, . . . etc.).

- Les ressources consommables :Qui lorsqu'après sa libération, elle n'est pas disponible en même quantité (argent, matières premières, . . . etc.).

Dans le cas des ressources renouvelables, on distingue principalement, les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité.

Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre simultanément les variables représentant les relations reliant les tâches et les ressources. On distingue deux types de contraintes, les contraintes temporelles et les contraintes de ressources.

• **Les contraintes temporelles :** concernent les délais de fabrication imposés. Ces contraintes peuvent être :

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

• **Les contraintes de ressources :** Ces contraintes concernent la limitation de la quantité de ressources de chaque type.

Dans ce cadre, deux types de contraintes de ressources sont distinguées:

- Les contraintes disjonctives : induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource.
- Les contraintes cumulatives : impliquant la limitation du nombre de tâches à réaliser en parallèle.

Les critères

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

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

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

- La minimisation des dates d'achèvement des actions,

- La minimisation du maximum des dates d'achèvement des actions,

- La minimisation de la moyenne des dates d'achèvement des actions,

- La minimisation des retards sur les dates d'achèvement des actions,

- La minimisation du maximum des retards sur les dates d'achèvement des actions.

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

- La minimisation des encours,

- La minimisation du cout de stockage des matières premières,

- L'équilibrage des charges des machines,

- L'optimisation des changements d'outils.

Les objectifs

Dans La résolution d'un problème d'ordonnancement, on peut choisir entre deux grands types de stratégies, visant respectivement à l'optimalité des solutions, ou plus simplement à leur admissibilité [07].

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

Liés au temps :

- Le temps total d'exécution ou le temps moyen d'achèvement d'un ensemble de taches.

- Le stock d'en-cours de traitement.

- Différents retards (maximum, moyen, somme, nombre, etc.) ou avances par rapport aux dates limites fixées.

Liés aux ressources :

- La quantité totale ou pondérée de ressources nécessaires pour réaliser un ensemble de taches.

- La charge de chaque ressource.

- Liés à une énergie ou un débit.

- Liés aux couts de lancement, de production, de transport, etc.., mais aussi aux revenus, aux retours d'investissements.

1.3.4 Domaine d'utilisation

L'ordonnancement peut apparaitre dans plusieurs domaines, on peut les illustrés par la figure suivante [07].

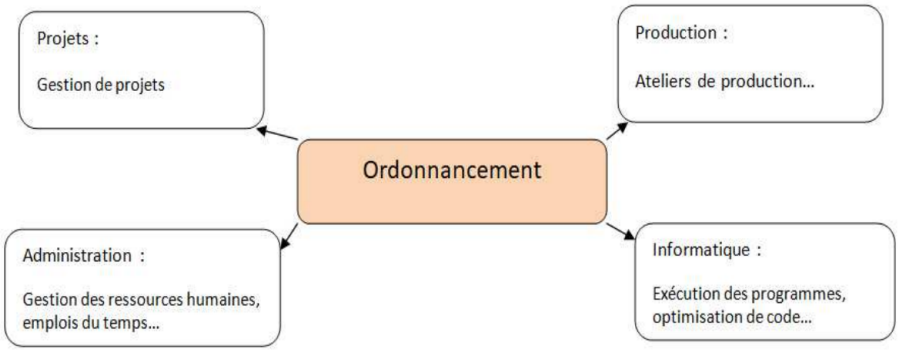


Figure 1.2: Domaine d'utilisation de l'ordonnancement

1.3.5 Notation et définition des problèmes d'ordonnancement

La notation $\alpha \mid \beta \mid \gamma$ est constituée de 3 paramètres ayant chacun une signification différente : le paramètre α nous renseigne sur l'environnement d'usinage, le type d'atelier de production et le nombre de machines présentes dans le système, tandis que le champ β concerne les caractéristiques des problèmes d'ordonnancement et regroupe les contraintes imposées par l'environnement de production et par les ressources. Le champ γ concerne les critères de performance qui vont représenter les objectifs à atteindre pour le problème en question.

le champ α

Le champ α décrit les machines utilisées, le type de machines, leur nombre, et le type d'atelier, il est composé de plusieurs sous champs et est distingué par :
$\alpha = \alpha_1, \alpha_2$ tel que :

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

$\alpha_1 = 1$: une seule machines.

α_1 = P : machines parallèles identique.

α_1 = Q : machines parallèles uniformes.

α_1 = R : machines parallèles différentes.

α_1 = O : Open Shop.

α_1 = F : Flow Shop.

α_1 = J : Job Shop.

le champ β

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

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

$\beta_2 \in \{\emptyset, prec, tree, chain, \dots\}$: *indique la possibilité d'interruption des taches.*

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

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

$\beta_5 \in \{\emptyset, di\}$: *indique les dates échues.*

le champ γ

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

$\gamma \in \{C_{\max}, \Sigma\, C_i\}$

ou :

C $_{max}$: la durée totale d'exécution des taches.

$\Sigma\, C_i$: *la date de fin moyenne des taches.*

Exemple des notations

$\alpha 1 = J, a2$=4 / $\beta 2 = prec, b7$=5 / γ=Cmax est un problème de type job-shop où on a besoin de traiter 5 jobs sur4 machines, avec les contraintes de précédence entre les opérations.

1.3.6 Les ateliers de production

Les systèmes de production sont divers et variés et ont différentes organisations au niveau des ateliers de production. Ces organisations déterminent les problèmes d'ordonnancements. Un atelier est caractérisé par le nombre de machines qu'il contient et par son type, ainsi on trouve :

Une seule machine

Les problèmes d'ateliers à une seule machine en anglais appelé (single machine problem). C'est le plus simple, chaque tache Jj de durée de traitement Pj s'exécute sur une machine qui ne peut traiter plus qu'une tâche à la fois; c'est à dire ordonnancer sur une seule machine des jobs constitués d'une seule opération.

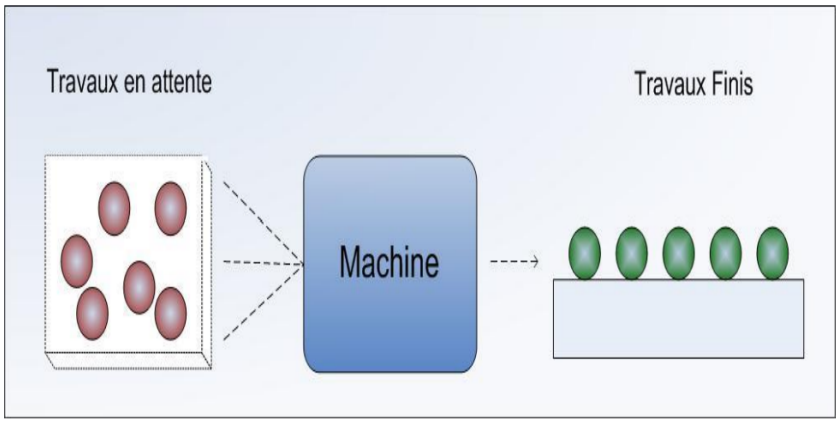


Figure 1.3: Représentation des machines uniques

Machines identiques en parallèle

Il y a m machines identiques en parallèle ; chaque tâche peut être exécutée indifféremment sur une des m machines, Celles-ci peuvent cependant avoir des vitesses différentes (à l'opposé des vitesses uniformes).

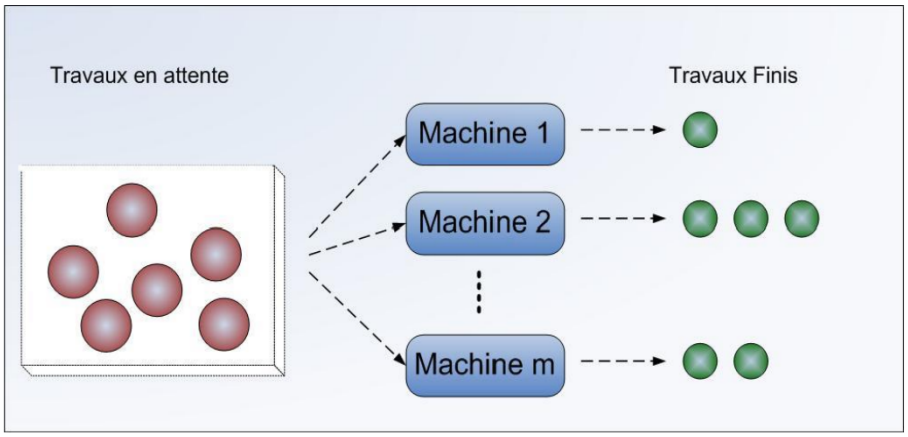


Figure 1.4: Représentation des machines parallèles

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

- Machines parallèles identiques : la durée d'exécution des opérations est la même sur toutes les machines.

- Machines parallèles uniformes : la durée d'exécution des opérations varie uniformément en fonction de la performance de la machine choisie.

- Machines parallèles indépendantes : les durées opératoires dépendent complètement des machines utilisées.

Les ateliers de type flow-shop

Chaque ordre de fabrication doit être traité par chacune des m machines en série et dans le même ordre. Toutes les tâches ont donc le même routage. En temps identiques, Dans les ateliers de type flow-shop hybride, une machine peut exister en plusieurs exemplaires identiques fonctionnant en parallèle.

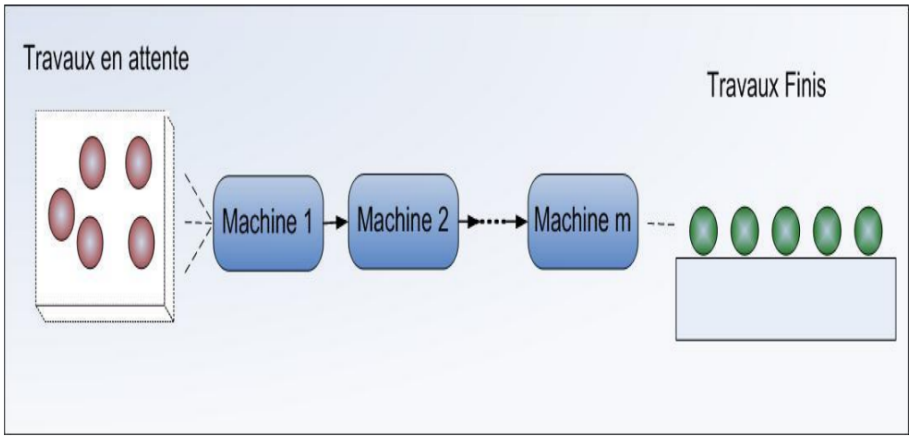


Figure 1.5: Représentation d'ateliers à cheminement unique(Flow Shop)

Les ateliers de type job shop

Chaque tâche possède son propre routage, Une tâche peut revenir une seconde fois sur la même machine. C'est le phénomène de recirculation.

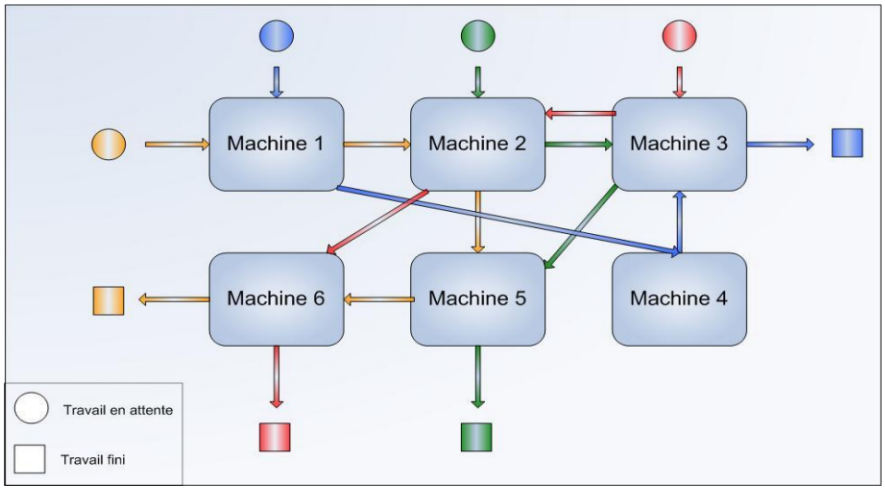


Figure 1.6: Représentation d'ateliers à cheminement multiple (Job Shop)

Les ateliers de type open-shop

C'est un type d'atelier moins contraint que le type Flow shop et le type job shop, puisque l'ordre des opérations n'est pas fixe à priori, Le problème d'ordonnancement consiste d'une part à déterminer le cheminement de chaque travail et d'autre part à ordonnancer les travaux en tenant compte des gammes trouvées, les opérations peuvent être effectuées dans n'importe quel ordre.

1.3.7 Représentation des problèmes d'ordonnancement

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

Le diagramme de GANTT

Le but de ce diagramme est de représenter de manière claire et rapide la solution à un problème d'ordonnancement. Deux types de diagramme de Gantt sont utilisés : Gantt ressources et Gantt tâches (figure 1.7). Le diagramme de Gantt ressource, est composé d'une ligne horizontale pour chaque ressource (machine). Sur cette ligne, sont visualisées les périodes d'exécution des différentes opérations en séquence et les périodes de l'oisiveté de la ressource. [07] Le diagramme de Gantt tâches permet de visualiser les séquences des opérations des tâches, en représentant chaque tâche par une ligne sur laquelle sont visibles, les périodes d'exécution des opérations et les périodes ou la tâche est en attente des ressources.

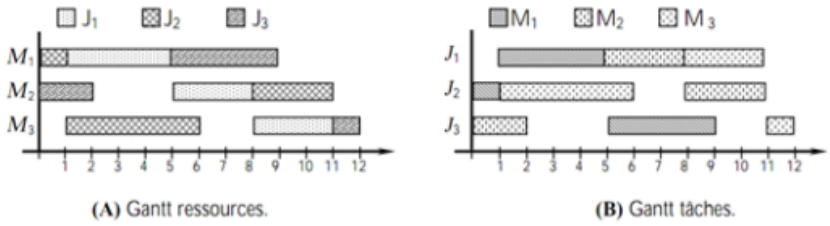


Figure 1.7: Diagramme de Gantt

Graphe Potentiel-Tâches

Cet outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à évènements discrets [08]. Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les contraintes par des arcs [09], comme le montre la figure 1.8. Ainsi, les arcs peuvent être de deux types :

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

• les arcs disjonctifs indiquant les contraintes de ressources [10], [11].

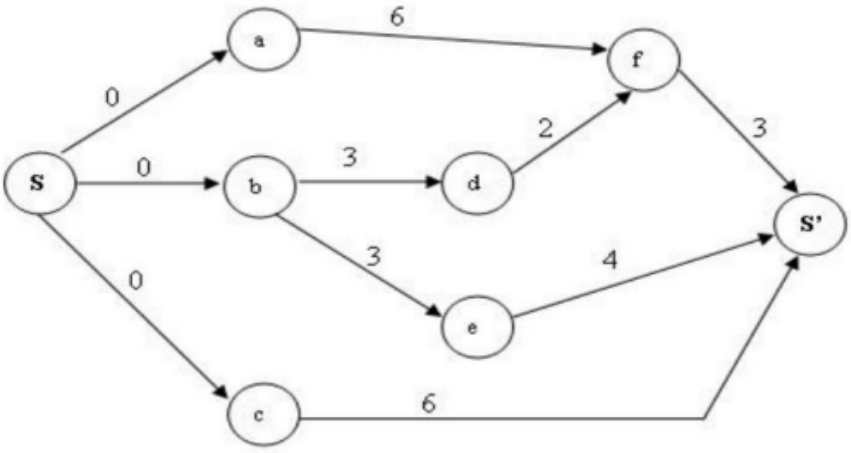


Figure 1.8: Graphe Potentiel-Tâches d'un ordonnancement

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

Method PERT (Program Evaluation and Research Task)

Cette représentation, semblable à la précédente, permet de représenter une tâche par un arc, auquel est associé un chiffre qui représente la durée de la tâche. Entre les arcs, figurent des cercles, appelés sommets ou événements, qui marquent l'aboutissement d'une ou de plusieurs tâches. Ces cercles sont numérotés afin de suivre l'ordre de succession des divers évènements. Les méthodes graphiques ont connu une très importante évolution surtout avec l'apparition des Réseaux de Pétri (RDP) [12], [13], qui permettent de traduire plusieurs notions fondamentales ayant un lien avec les problèmes d'ordonnancement, telles que : les conflits sur les ressources, les durées opératoires certaines, les durées opératoires aléatoires, les gammes, les disponibilités, les multiplicités et les capacités de ressources la répétitivité, etc ...

1.3.8 Méthodes de résolution

Au fil des années, de nombreuses méthodes de résolution de problèmes ont été proposées. Ainsi, une grande variété de concept et principe, de la stratégie et des performances ont été discernées. Cette variété et ces différences ont permis de regrouper les différentes méthodes de résolution de problèmes d'ordonnancement en deux classes principales : la classe de méthodes exactes et la classe des méthodes approchées.

Les Méthodes Exactes

Elles recherchent un ordonnancement optimal qui minimise ou maximise un des critères présentés ou une combinaison de plusieurs critères. Les techniques utilisées sont les méthodes par séparation et évaluation, la programmation dynamique, la déduction mathématique, la théorie des jeux, la théorie des graphes etc. Cette technique est coûteuse en temps de calcul.

Les Heuristiques

Les méthodes dites heuristiques sont des méthodes spécifiques à un problème particulier. Elles nécessitent des connaissances du domaine du problème traité. En fait, ce sont des règles empiriques qui se basent sur l'expérience et les résultats acquis afin d'améliorer les recherches futures

Les Méta-heuristiques

Les méthodes dites méta heuristiques sont des méthodes générales, des heuristiques polyvalentes applicables sur une grande gamme de problèmes. Elles peuvent construire une al-

ternative aux méthodes heuristiques lorsqu'on ne connaît pas l'heuristique spécifique à un problème donné. [14].

1.4 Conclusion

La préparation d'un bon ordonnancement est un processus très important qui influence directement la productivité du système, quel que soit le type de système de production, En effet un mauvais ordonnancement signifie une utilisation non adéquate de ressources de production et donc, une consommation physique (ressource) et temporelle (temps de production) élevée, qui se traduit par une performance basse du système de production. Les problèmes d'ordonnancement sont des problèmes combinatoires qui sont classé parmi les problèmes d'optimisation les plus difficiles. Le job shop est l'un des problèmes d'ordonnancement les plus complexe. Alors le deuxième chapitre permettra de présenter une vue globale sur ce problème et la théchnique de résolution .

Chapitre 2

Algorithmes génétiques pour la résolution de problème d'ordonnancement de Job Shop

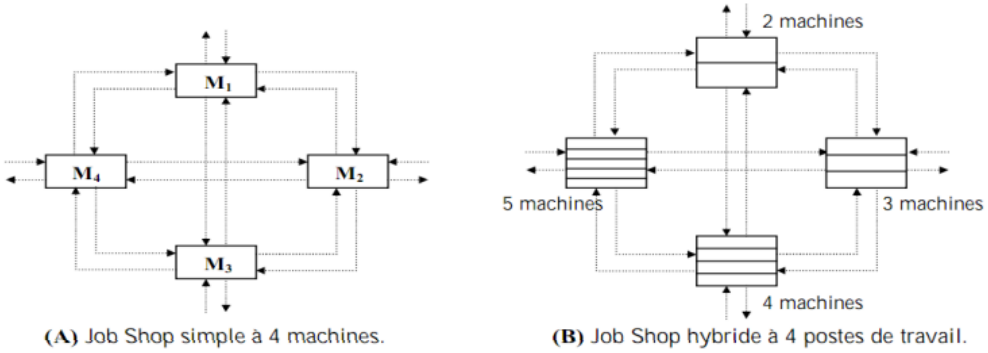
2.1 Introduction

Avant les problèmes de job shop sont considérés comme des problèmes simples dont les déplacements des pièces d'une machine à une autre n'étaient pas pris en compte dans le calcul de l'ordonnancement ou bien sont considérés intégrés dans la tâche elle-même. Or, cette hypothèse est souvent non justifiée en pratique. En effet, le transport des tâches doit être intégré aux données du problème. Ce présent chapitre est consacré à la présentation de ce type de problème ainsi que les algorithmes génétiques.

2.2 Le Problème d'ordonnancement Job-Shop

2.2.1 Atelier à cheminement multiple (job shop)

Dans un atelier de type job shop, chaque tâche a son propre cheminement. Les tâches ne s'exécuteront pas dans le même ordre sur toutes les machines. Cette forme d'organisation correspond généralement à une production par lot, notamment dans une unité de production, utilisant plusieurs moyens dans des séquences différentes produire divers produits.
Si pour chaque machine nous n'avons qu'un seul exemplaire, nous disons que l'organisation est un job shop simple. Au lieu de cela, si pour au moins une machine (poste de travail) nous avons Plus d'un exemplaire, ça s'appelle le Job Shop Hybride (Comme il est montré dans la figure 2.1).



(A) Job Shop simple à 4 machines. **(B)** Job Shop hybride à 4 postes de travail.

Figure 2.1: Exemple de job shop simple et hybride.

2.2.2 Types de job shop

Job shop Classique

Le problème d'ordonnancement de l'atelier de travail est l'un des problèmes d'ordonnancement Les plus étudiées dans la littérature, les variantes du problème du job shop sont Beaucoup, et il est impossible de trouver son expression unique dans la littérature. Nous introduisons ici la formulation la plus générale possible du problème de job shop simple. Le problème d'ordonnancement de job shop implique l'exécution d'un ensemble de n job sur un ensemble de m machines. La recherche atteint certains objectifs. Chaque job Ji consiste en une séquence de ni opérations qui seront exécutées sur différentes machines selon un ordre préalablement défini. De plus, il existe généralement un ensemble de contraintes sur les machines et les tâches, telles que :

- Ces machines sont indépendantes les unes des autres. - Une machine ne peut effectuer qu'une seule opération à la fois.

- La machine est disponible lors de la planification.

- Les jobs sont indépendants les uns des autres. En particulier, il n'y a pas d'ordre Le travail passe avant tout.
La figure (2.2) représente un problème type de job shop classique composé de 3 jobs et 4machines, les gammes opératoires sont les suivantes :

J1 :M1-M2-M4.J2 :M2-M3-M1.J3 :M4-M3-M2.

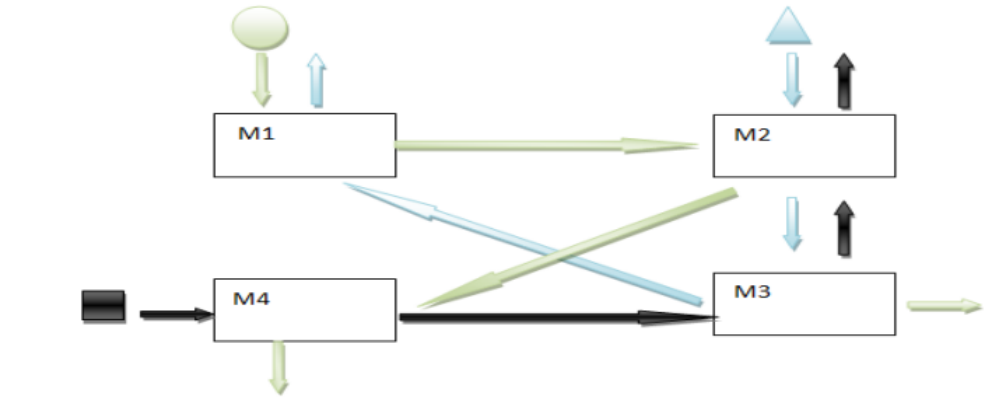


Figure 2.2: job shop classique à 3 jobs et 4 machines.

Job shop flexible

Le job shop flexible est une extension du modèle de job shop classique. Sa spécificité essentielle est que chaque opération peut être réalisée sur plusieurs machines. Dans ce modèle, les machines effectuant la même opération sont regroupées sur un même étage.
Ainsi, en raison de la polyvalence de ces machines, il offre une plus grande flexibilité par rapport aux job shop. Cependant, en raison de la nécessité de détermination d'allocation appropriée avant de déterminer l'ordre dans lequel effectuer diverses opérations sur la machine. Un exemple de ce problème à m étages et trois machines maximum par étage, est donné dans la Figure(2.3). [15]

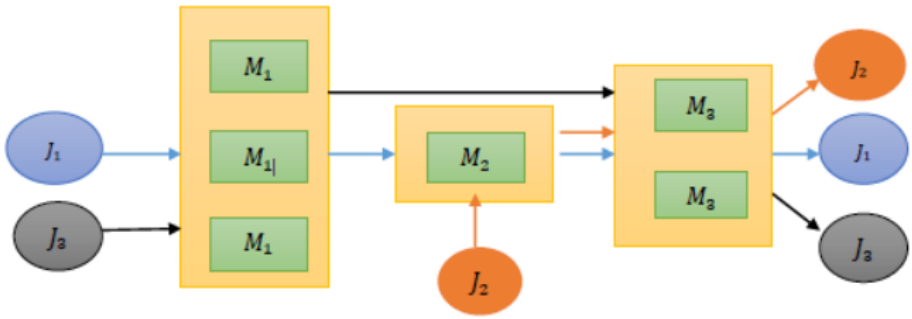


Figure 2.3: Représentation d'un système de type Job-shop flexible à deux étages.

Une mise en garde importante aux problèmes de type job shop hybride est que le nombre de machines peut varier d'un étage à l'autre et que les performances de la machine ne sont pas nécessairement les mêmes pour toutes les machines au même étage. Ces propriétés permettent de classer les systèmes à machines parallèles en trois groupes, à savoir :

- Machines identiques : Le temps d'exécution de la tâche est le même sur toutes les machines.

- Machines uniformes : Le temps d'exécution des tâches est similaire sur toutes les machines.

- Machine indépendante : Le temps d'exécution d'une opération dépend de la machine sur laquelle elle est exécutée.

2.2.3 Complexité

Les problèmes de job shop sont en général NP-complets [24], même si l'atelier est simple. En effet, Lenstra et Rinnooy Kan [25] ont montré que les ateliers possédant plus de trois machines, ou un nombre de tâches supérieur ou égal à trois, sont NP-difficiles même si la préemption est permise. De même, pour les problèmes à deux machines, dès qu'il y a recirculation, ils deviennent fortement NP-difficiles. En revanche, si la fonction objectif est la minimisation du makespan, alors il existe quelques cas particuliers résolubles en temps polynomiaux :

- un atelier composé de deux machines dont chaque tâche comprend au plus deux opérations, Jackson [26]

- un atelier comprenant deux tâches sur m machines, Akers [27], Brucker [28],

- un atelier à deux machines dont les tâches ont des opérations unitaires, Kubiak étal. [29],

- un atelier à deux machines dont les tâches ont des opérations unitaires avec des dates de disponibilité, Timkovsky [30],

- un atelier à deux machines dont le nombre de tâches est fixe, Brucker [31].

Malheureusement, même de petites modifications à ces ateliers les font basculer dans la classe des problèmes NP-difficiles. Par exemple, l'ajout d'une troisième machine même pour des ateliers ayant des tâches unitaires devient NP-difficile. Le Tableau 2.1 résume les principales complexités en fonction des caractéristiques de l'atelier.

Table 2.1: Récapitulatif des principales complexités d'un job shop

Type d'atelier	Complexité	Référence
J2 | $n_{\text{i}} \leq 2$ | $C_{\max}$	O(nlogn)	Jackson, 1956 [36]
J2 | $p_{\text{ij}} = 1$ | $C_{\max}$	O(nlog(nr)	Kubiak, 1995 [43]
J2 | $p_{\text{ij}} = 1; r_{\text{i}}$ | $C_{\max}$	O(n ^{2})	Timkovsky, 1997 [81]
J2 | $n = k$ | $C_{\max}$	O(r^{2k})	Brucker, 1994 [13]
J2| $p_{\text{ij}} \in \{1, 2\}$ | $C_{\max}$	NP-difficile	Lenstra et Rinnooy Kan, 1979 [51]
J2| $chains; p_{\text{ij}} = 1$ | $C_{\max}$	NP-difficile	Timkovsky, 1985 [79]
J2| $n = 3; prmtn$ | $C_{\max}$	NP-difficile	Brucker et a/., 1996 [16]
J3 | $p_{\text{ij}} = 1$ | $C_{\max}$	NP-difficile	Lenstra et Rinnooy Kan, 1979 [16]
J3 | $n = 3$ | $C_{\max}$	NP-difficile	Sotskov et Shakhlevich, 1995 [76]

2.2.4 Résolution d'un problème d'ordonnancement d'un type job shop par les algorithmes génétiques

Présentation du problème

Dans le problème d'ordonnancement de type job shop, les jobs finis doivent être traités par des machines finies. Chaque travail consiste en une séquence prédéterminée d'opérations de tâche, dont chacune doit être traitée sans préemption pendant une période de temps donnée sur une machine donnée. Les tâches d'un même travail ne peuvent pas être traitées simultanément et chaque travail doit visiter chaque machine exactement une fois.
Chaque opération ne peut pas être commencée tant que le traitement n'est pas terminé, si l'opération précédente est toujours en cours de traitement. Un planning est une affectation d'opérations à des time slots sur une machine. Le Makespan est le temps d'exécution maximal des job dont l'objectif est de trouver une solution qui le minimise .

Formulation du problème

Nous considérons le cas flexible où les étapes peuvent être sauté

- Chaque travail est une chaîne d'opérations et chaque opération doit être traitée sur une machine donnée pendant un temps donné.

- La tâche consiste à trouver que le temps d'exécution de la toute dernière opération est minimal.

- L'ordre de la chaîne de chaque tâche doit être maintenu et chaque machine ne peut traiter qu'une seule tâche à la fois.

- Aucun job ne peut être préemptée.

- une fois qu'une opération commence, elle doit être terminée .

- deux opérations d'un job ne peuvent pas être traitées en même temps.

- Pas plus d'un job ne peut être traité sur une machine en même temps.

- Le même niveau de priorité à chaque operation.

- il n'y a pas de configuration et de temps d'inactivité.

- il n'y a pas de temps de pause.

- toutes les machines sont disponibles à t = 0 .

Les définitions et notations supplémentaires suivantes aideront à formuler le problème :

a - Notations

(1) i : Nombres de machines.

(2) j_i : *Nombre d'opérations de la machine i* .

(3) p_{ij} : *Temps de traitement de l'operation j sur la machine i* .

(4) t_{ij} : *Temps de début de l'opération j sur la machine i.*

(5) t_j : *Temps d'achèvement de l'opération j .*

(6) $X_{ijk} = \begin{cases} 1 \, si \, l'opration j \, prcde \, l'opration, \\ 0 \qquad\qquad sinon \end{cases}$

(7) $Z_{ij} = \begin{cases} 1 \, si \, l'opration \, j \, est \, alloue \, sur \, la \, machine \\ 0 \qquad\qquad sinon \end{cases}$

(8) Cmax : makespan

b - Critères à minimiser

L'objectif de notre travail est de trouver un planning qui minimise le Makespan (le temps d'exécution maximal des jobs).

c -Fonction fitness à optimiser

Max C_i

$S.t$

$C_i = \sum_{j=0}^{nbopration} P_{ij} + t_{ij}.$

Algorithmes génétiques

Les algorithmes génétiques font partie des algorithmes évolutionnaires dont son nom est dérivé de l'analogie avec le mécanisme d'évolution des espèces biologiques. Un algorithme évolutionniste typique se compose de trois éléments de base : une population composée de plusieurs individus représentant des solutions potentielles (configurations) à un problème donné, un mécanisme d'évaluation de l'adéquation de chaque individu de la population à son environnement extérieur, un mécanisme d'évolution composé d'opérateurs de mécanisme permet d'éliminer certains individus et d'en générer de nouveaux à partir d'individus sélectionnés[16].

a - Présentation des algorithmes génétiques

Le développement d'algorithmes génétiques a parcouru un long chemin au cours des trois dernières décennies depuis l'idée originale de Holland. L'algorithme génétique est une technique de recherche basée sur le mécanisme de Sélection naturelle et génétique. Il s'est montré très efficace pour résoudre des problèmes d'optimisation complexes comme le problème du voyageur de commerce [17] , [18] ou les problèmes d'ordonnancement [19], [20] [21] , là où

Les méthodes traditionnelles, telles que les algorithmes branch and bound, ont montré leurs limites.

b - Principe de base des AG

Indépendamment de la problématique traitée, les algorithmes génétiques sont basés sur six principes :

1- Choisir le codage des solutions;

2- Générer une population initiale de taille fixe N, formée d'un ensemble fini de solutions, dite generation initiale;

3- Définir une function d'évaluation (fitness) permettant d'évaluer une solution et la comparer aux autres;

4- Choisir les solutions par un mécanisme de selection qui choisit pour un éventuel couplage;

5- Générer de nouvelles solutions à l'aide des opérateurs génétiques en utilisant :

• Opérateur de coisement : il manipule la structure des chromosomes des parents afin de produire des individus meilleurs ou différents. Cet opérateur est effectué selon une probabilité Px .

• Opérateur de mutation : il évite d'établir des populations uniformes incapables d'évoluer. Il consiste à modifier la valeurs des genes de chromosomes selon une probabilité Pm .

6- Etablir un compromis entre les solutions produites (progénitures) et les solutions productrices (les parents) en utilisant un mécanisme d'insertion. En d'autres termes, et suite à des informations précises, decider ce qui doit rester et ce qui doit disparaitre. Tout ceci, en souvgardant à chaque generation une taille de la population N fixe.

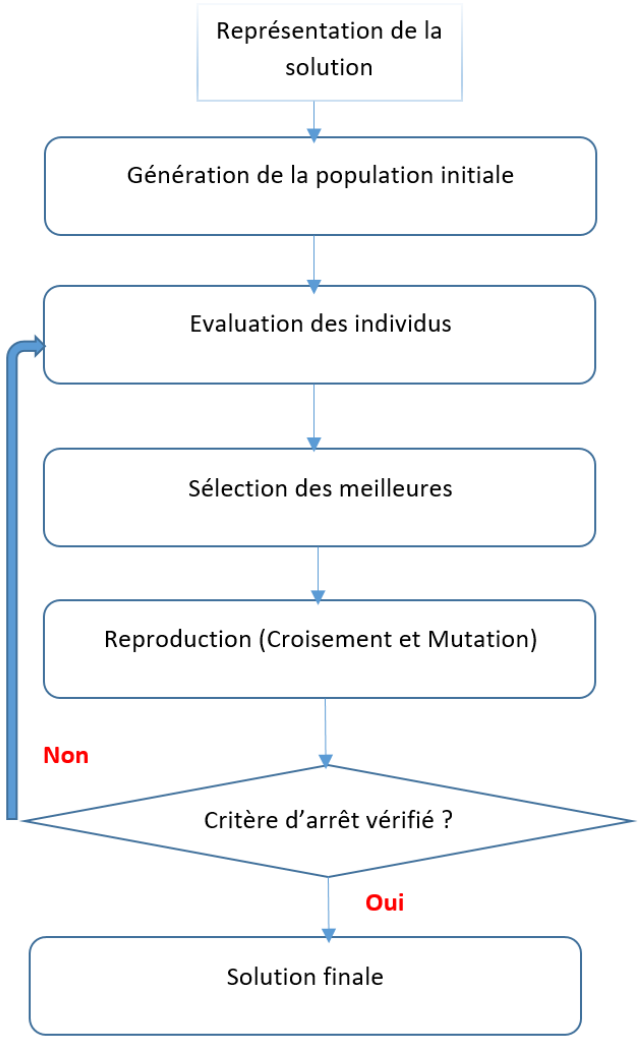


Figure 2.4: Principe de base des algorithmes génétiques

c - Fonctionnement d'un algorithme génétique

Dans la recherche d'une solution robuste et optimale, l'algorithme génétique se distingue des méthodes classiques qui peuvent s'exprimer selon quatre axes principaux :

- Les algorithmes génétiques utilisent des encodages de paramètres plutôt que les paramètres eux-mêmes.

- ils traitent un grand nombre de points au lieu d'un seul point.

- ils n'utilisent que la valeur de la fonction étudiée, pas sa dérivée ou d'autres connais-

sances auxiliaires.

- Ils utilisent des règles de transition probabilistes et non déterministes [22] .

Par conséquent, la première étape de l'application d'un algorithme génétique à un problème particulier consiste à convertir une solution réalisable à ce problème en une structure appelée chromosome. Pour trouver la meilleure solution à un problème donné, les algorithmes génétiques standard commencent par générer de manière aléatoire un ensemble de solutions (chromosomes), appelé la population initiale, et évoluent au fil des générations vers des solutions différentes mais meilleures. À chaque génération, la fonction objective ou fitness détermine quels chromosomes sont sélectionnés pour la reproduction. L'algorithme sélectionne ainsi le meilleur chromosome (avec la meilleure fonction de fitness). Des opérateurs génétiques, tels que le croisement et la mutation, sont appliqués à ces chromosomes. Les nouveaux chromosomes (descendants) qui en résultent constituent la prochaine génération. Ces itérations se poursuivent jusqu'à ce que le critère d'arrêt soit satisfait.

d - Codage des algorithmes génétiques

Le codage des solutions au problème d'optimisation obtenu en utilisant l'algorithme génétique consiste à modéliser chaque solution par chromosomes, c'est-à-dire par séquences de gènes. Initialement, le codage utilisé par l'algorithme génétique est Représenté comme une chaîne de bits contenant toutes les informations nécessaires pour décrire un point dans l'espace d'état. L'avantage de ce codage est que de simples opérateurs de croisement et de mutation peuvent être créés. Aussi en utilisant ce type d'encodage, le premier résultat La convergence théorique a été obtenue [23]. Plus généralement, un gène peut être un entier, un nombre réel, un caractère, ou toute autre extension ou collection de ces entités École élémentaire [20]. Le choix adéquat du codage, ou la description de la solution, est une tâche importante qui peut garantir le succès de l'application des algorithmes génétiques [24].

e - Opérateurs des algorithmes génétiques

Les opérateurs utilisés par les algorithmes génétiques font que la population de plusieurs générations se diversifie et explore l'espace d'état qui est représenté par l'espace de la solution. L'opérateur de croisement reconstruit les gènes des individus dans la population car l'opérateur de mutaion vise à assurer l'exploration de l'espace national. Les opérateurs suivants proposent ces différents opérateurs pour résoudre la solution au problème.

1 - Opérateur de selection

La sélection permet de déterminer les meilleurs individus de la population et d'éliminer partiellement les mauvais. Néanmoins, ce n'est pas parce qu'un individu est bon qu'il survit nécessairement et ce n'est pas parce qu'il est mauvais qu'il disparaît.

2 - Opérateur de croisement

L'intention de l' opérateur de croisement est de diversifier la population en manipulant la structure des chromosomes. L'opérateur de croisement à un point proposé consiste à choisir deux parents et un point de croisement. Le père 1 (père 2) reçoit les chromosomes du père 2 (les chromosomes du père 1), qui suivent le point de croisement, comme le montre la figure 2.5 Ensuite, mettez à jour pour calculer la fonction de fitness.

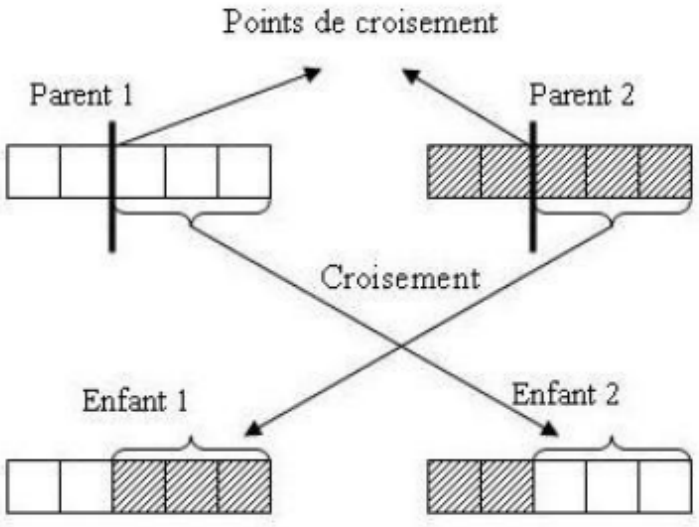


Figure 2.5: Fonctionnement de l'opérateur de croisement

3 - Opérateur de mutation

Les opérateurs de mutation impliquent la modification aléatoire de la valeur de certains gènes dans un chromosome, comme le montre la figure 2.6. Sans elle, la population restera

uniforme et n'évoluera peut-être pas.

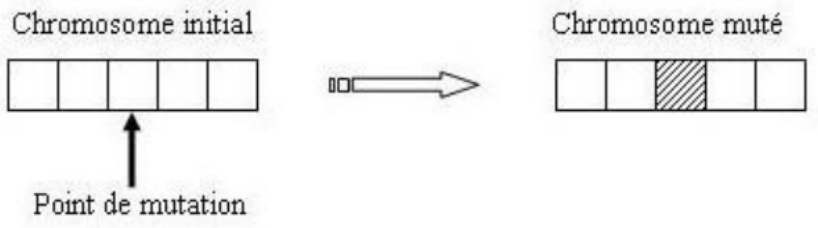


Figure 2.6: Fonctionnement de l'opérateur de mutation

Remarque

Pour pouvoir appliquer ces trois opérateurs, on a besoin de définir principalement trois paramètres : le nombre d'individus dans la population n, les taux de croisement Pc et de mutation Pm. Trouver de bonnes valeurs de ces paramètres est un problème parfois délicat. La valeur de n dépend fortement du problème, alors que des taux de 5% de mutation et 70% de croisement sont souvent utilisés.

Les effets des opérateurs génétiques cités plus haut sur l'évolution d'une population donnée s'expliquent en adoptant un des concepts centraux de l'analyse des algorithme génétiques qui est le concept de schéma.

f - Les avantages des algorithms génétiques

Par rapport aux algorithms classiques d'optimisation, l'algorithme génétique présente plusieurs points forts comme :

- Le fait d'utiliser seulement l'évaluation de la function objectif sans se soucier de sa nature. En effet, nous n'avons besoin d'aucune propriété particulière sur la function à optimiser(continuité, dérivabilité,convexité,etc), ce qui lui donne plus de souplesse et un large domaine d'application;

- Génération d'une forme de parallélisme en travaillant sur plusieurs points en meme temps (population de taille N) au lieu d'un seul itéré dans les algorithms classiques;

- L'utilisation des règles de transition probabilistes (probabilité de croisement et de mutation), contrairement aux algorithms determinists ou la transition entre deux iterations successives est impose par la structure et la nature de l'algorithme. Cette utilization permet dans certaines situations aux algorithmes génétiques d'éviter des optimums locaux et de se diriger vers un optimum global .

2.3 Conclusion

Plusieurs travaux de recherche ont été effectués pour trouver des solutions aux problèmes d'atelier. ils ont appliqué des méthodes exactes et des méthodes approchées dont l'objectif est de rapprocher plus à la réalité. Dans le chapitre suivant une proposition sera donnée sur la resolution de ce problème.

Chapitre 3

Implémentation et Expérimentation

3.1 Introduction

Dans ce chapitre une présentation de la démarche de résolution est faite . L'objectif du travail est de trouver, en utilisant les algorithms génétiques, les meilleures solutions possibles pour un ordonnancement de type job shop, tout en minimisant le temps total d'exécution (Makespan). Ensuit une description des technologies utilisées pour l'implémentation et quelques capture d'écrans de l'application seront donnés .

3.2 Description du système étudié

On désigne sous le terme job-shop JxM , une instance du problème de job-shop simple constituée de J produits et M machines. Un job-shop est dit simple si chaque produit est constitué d'une seul gamme de fabrication, et si chaque opération de la gamme ne peut être effectuée que sur une seule machine. Le coût global de production est lié au temps nécessaire à la fabrication des différents produits, l'objectif consiste à réduire la durée globale de fabrication, appelée « makespan ».

3.3 Démarche de resolution

Dans cette partie une présentation de la démarche de résolution ainsi que l'algorithme pour résoudre le problème seront donné.

3.3.1 Description de la solution

Pour encadrer la problématique une proposition d'un ensemble d'hypothèses concernant le fonctionnement de système sera donné.

- Les produits sont toujours disponibles à l'instant t=0.

- Chaque machine possède une file d'attend avec une capacité illimitée.

- Pas de panne de machines.

- Pas de recirculation (un produit ne peut exécuter sur une machine qu'une seule fois)

- trouver le Cmax minimum.

L'algorithme de résolution

L'espace des solutions est exploré grâce au mécanisme appliqué des opérateurs génétiques, à savoir le croisement et la mutation sur une population initiale définie aléatoirement. Elles s'étalent sur plusieurs générations dont le nombre est fixé par un critère d'arrêt, laissant espérer une solution proche de la solution optimale, voire optimale. Les étapes décrites ci-après, expliquent en détail le fonctionnement des différents opérateurs de l'algorithme génétique appliqué aux problèmes d'ordonnancement de type job shop.

- Génération de la population initiale.

- Sélection.

- Reproduction (croisement et mutation).

- Remplacement par la nouvelle population.

Algorithm 1 Algorithme génétique

```
Générer population initiale;
for NombreDeGenaration do
  for NombreDePopulation do
    Générer CreerChromosom();
    Générer Population.ajouter(Chromosom);
    Générer CrossOver;
    Générer Mutation;
    Générer SupprimerMauvaisCHromosom;
  end for
end for
```

A. Génération de la Population Initiale

Plusieurs méthodes existent pour définir le mécanisme de la Génération d'une population initiale représentant un point de départ pour la constitution des générations futures : sélection aléatoire, heuristique ou une combinaison d'heuristiques et de solutions aléatoires.

B. Sélection

Cette a phase consiste à choisir parmi les N individus i de la population courante les plus forts individus à partir desquels la génération suivante sera créée. Soit pour un ordonnancement i réalisable, la valeur de la fonction objectif « Makespan » est calculée. à chaque solution une fonction d'évaluation pour calculer sa force d'adaptation Fi est associe, dans ce cas le problème à pour but de minimiser la fonction objectif min f (i).

Algorithm 2 Créer Chromosom

```
mélanger toutes les opérations;
t=0;
while !toute les d'opérations terminée do
  selectedOperations();
  for chaque opération do
    if opération Non Précédée  then
      if l'opération n'est pas la première then
        if l'opération précédente n'est pas encore terminée then
          selectetionnerOperation;
        end if
      end if
    else
      selectionnerOperation;
    end if
  end for
  for toute les operations sélectionnées do
    if !la machine est ocuupée then
      Exécuter opération;
    end if
  end for
  t++;
end while
makspane=t-2;
```

C. Croisement (crossover)

L'opérateur de croisement est le plus important dans GA, ce qui permet une exploration efficace de l'espace de recherche. Il est appliqué à deux parents distincts parent1 et parent2 sélectionnés dans la population sélectionnée, génèrent deux nouvelles solutions enfant1 et enfant2 en combinant les propriétés des parents 1 et 2.

Algorithm 3 Croisement

```
Sélectionner aléatoirement deux parents parent1 et parent2;
First.half.child=First.half.parent1 ;
Second.half.child=Second.half.parent2;
```

Exemple de Croisement

Considérez les 2 parents suivants, qui ont été sélectionnés pour le croisement :

Table 3.1: parent 1

J_3O_2	J_2O_3	J_3O_3	J_1O_2	J_2O_2	J_1O_3	J_1O_1	J_3O_1	J_2O_1

Table 3.2: parent 2

J_1O_2	J_2O_2	J_1O_1	J_2O_3	J_1O_3	J_3O_3	J_3O_1	J_2O_1	J_3O_2

Si le taux de mélange est de 0, 5, environ la moitié des gènes proviendront de Le parent 1 et l'autre moitié proviendront du parent 2. Vous trouverez ci-dessous un ensemble possible de descendant après croisement uniforme:

Table 3.3: Selection aléatoire d'un fils

J_1O_1	J_3O_2	J_3O_1	J_2O_1	J_3O_3	J_1O_3	J_2O_3	J_2O_2	J_1O_2

Pour assurer la non duplication des jobs après le croisement les doublons seront remplacés au hasard par des tâches non placés.

Table 3.4: Exemple de changement

J_1O_1	J_3O_2	J_3O_1	J_2O_1	J_3O_3	J_1O_3	J_2O_3	J_2O_2	J_1O_2
Swap (0ET6)	Swap (1ET0)	Swap (2ET7)	Swap (3ET8)	Swap (4ET5)	Swap (5ET5)	Swap (6ET2)	Swap (7ET6)	Swap (8ET7)

Ce qui donne le fils suivant, comme illustré dans la table 3.5.

Table 3.5: Fils obtenue après le croisement

J_3O_2	J_2O_3	J_1O_1	J_1O_2	J_1O_3	J_3O_3	J_3O_1	J_2O_1	J_2O_2

D. Mutation

Le deuxième opérateur génétique important est la mutation, qui se classe au deuxième rang en termes d'importance par rapport au croisement. Les mutations sont des perturbations introduites dans les parties constitutives d'un individu afin de préserver la diversité et d'élargir le champ d'exploration. L'approche suivie dans ce travail consiste à sélectionner au hasard un individu, puis une sélection au hasard de deux opérations qui consistent à permuter l'ordre entre elles.

Algorithm 4 Mutation

Sélectionner aléatoirement un individu i;
Échanger (Operations, Random$_i ndex1, Random_i ndex2$);

E. Remplacement

Cette phase permet de ne retenir que les meilleurs éléments entre enfants et parents.

Algorithm 5 Remplacement

Rangement de la population des parents. ;
Rangement de la population enfants;
Remplacement;
Choisir les N meilleurs individus parmi les enfants et les Parents;

F. Critère d'arrêt

Le processus est stoppé au bout d'un nombre fixé de génération, ou lorsque les individus ont convergé vers une ou plusieurs solutions satisfaisantes, dans cet étude la première démarche est utilisé. Les meilleurs individus de la population sont alors retenus comme solutions au problème.

3.4 EXPERIMENTATION

Un système de production d'atelier est souvent défini comme étant un système de production complexe. Plusieurs machines collaborent à la réalisation d'un ensemble d'opérations comme l'assemblage, l'usinage, le chargement, le déchargement et le stockage d'un produit. pour simulé le fonctionnement des algorithmes génétiques sur le problème job shop le nombre des machines sera fixé à 3 et le nobmre des jobs pour chaque exemple ce change dont pour chaque job le nombre des opérations est fixé suivants le nombre de machine utilisées où dans chaque job il doit réalisé au moins une opération sur chaque machine.

3.4.1 Teste n°1

le problème JSP 3x3 représente l'exécution de trois jobs sur trois machines, pour chaque job j trois opérations sont associes, les opérations sont numérotés par des entiers, les durées opératoires et les machines utilisées pour exécuter les opérations sont fournies dans la table 3.6. Les paramètres des algorithmes génétiques utilisés sont les suivants :
- La taille de population initiale (Nbr-pop) = 20
- Probabilité de croisement (Pc) = 0.5
- Nombre de génération (NB_g) $= 50$

Table 3.6: Les durées opératoires et les affectations machines des operations.

	J1		J2		J3	
	Opérations	Durée	Opérations	Durée	Opérations	Durée
M1	O1	6	O2	14	O3	9
M2	O3	12	O1	16	O2	18
M3	O2	9	O3	5	O1	22

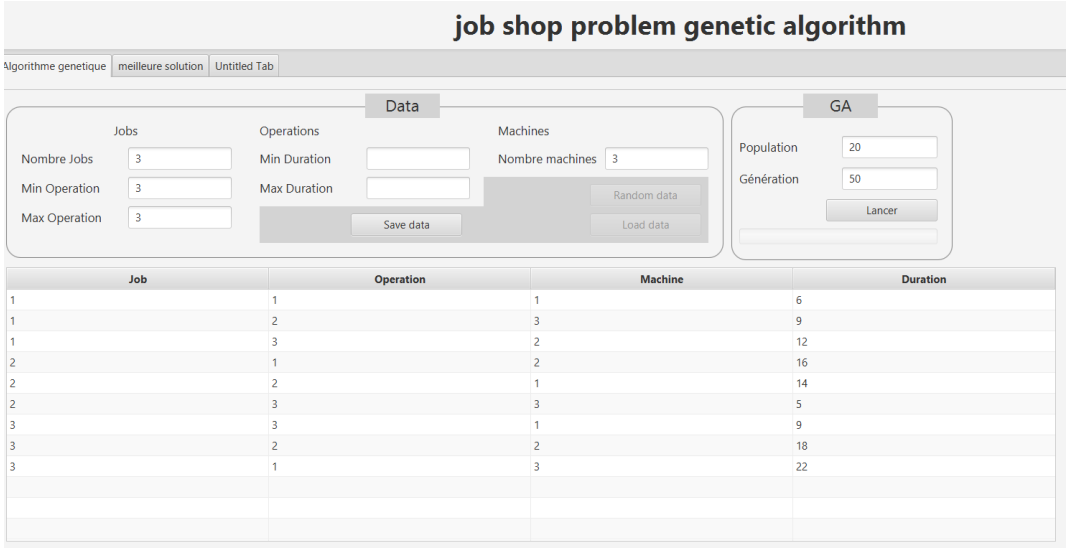


Figure 3.1: Exemple de jsp 3x3

Le diagramme de Gantt correspond à ce teste présenté dans la figure suivante représente le meilleur minimum obtenu après application des algorithmes génétiques sur le problème 3 x 3. ainsi, une répartition de trois jobs avec un Cmax de 54 est obtenue.

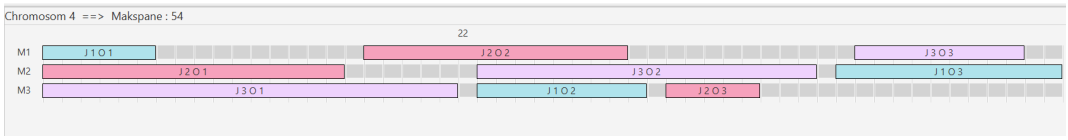


Figure 3.2: Diagramme de Gantt

les testes effectués consiste à augmenté à CHaque fois le nombre de jobs qui varie en gardant le nombre de machines fixer à trois , le nombre d'opérations est fixé par job suivant le nombre de machines .

Table 3.7: Les résultats des testes

Type de problème étudier	Nombres d'opérations par job	Nombre de génération	Valeur du makespan (temps)	Temps d'execution (seconde)
3x3	(3,3,3)	50	54	249165 ms
5x3	(3,3,3,3,3)	50	141	285992 ms
7x3	(3,3,3,3,3,3,3)	50	141	288254ms

3.4.2 Discussion des résultats

Dans cet échange, un algorithme génétique est utilisé pour résoudre le problème d'ordonnancement de type jo shop, qui appartient à la classe des problèmes NP-complets dans la littérature et est considéré comme l'un des problèmes d'optimisation combinatoire les plus difficiles. makespan représente l'objectif de minimisation. Le choix de cette méta-heuristique repose sur l'étude de grands problèmes, ce qui permet la construction de solutions de haute qualité avec un temps de calcul raisonnable. Les tests à effectuer consistent en un nombre fixe de machines, pour chaque jobe le nombre d'opérations est déterminé en fonction du nombre de machines utilisées, où chaque jobe doit exécuter au moins une opération sur chaque machine. Le nombre de solutions globales à chaque problème n'est lié qu'au nombre de jobs et au nombre de machines utilisées, dans cet étude seulement le nombre de jobs par test fait varier avec ce que est considéré être un nombre fixe de trois machines, l'espace de recherche peut vite devenir très grand. Les résultats obtenus sont très satisfaisants en termes de paramètres temporels et de contraintes de priorité, par contre l'optimalité est toujours maintenue approchée de la solution.

3.5 Implémentation

3.5.1 Outils de développement

NetBeans IDE

NetBeans IDE est un environnement de développement intégré (EDI), permet également de supporter différents autres langages, comme java, C, C++, JavaScript, XML, Groovy, PHP et HTML de faon native ainsi que bien d'autres (comme Python ou Ruby) par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refacturant, éditeur graphique d'interfaces et de pages Web). NetBeans

constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plateforme. La Figure 3.3 présente fenêtre de programmation Sur Netbeans :

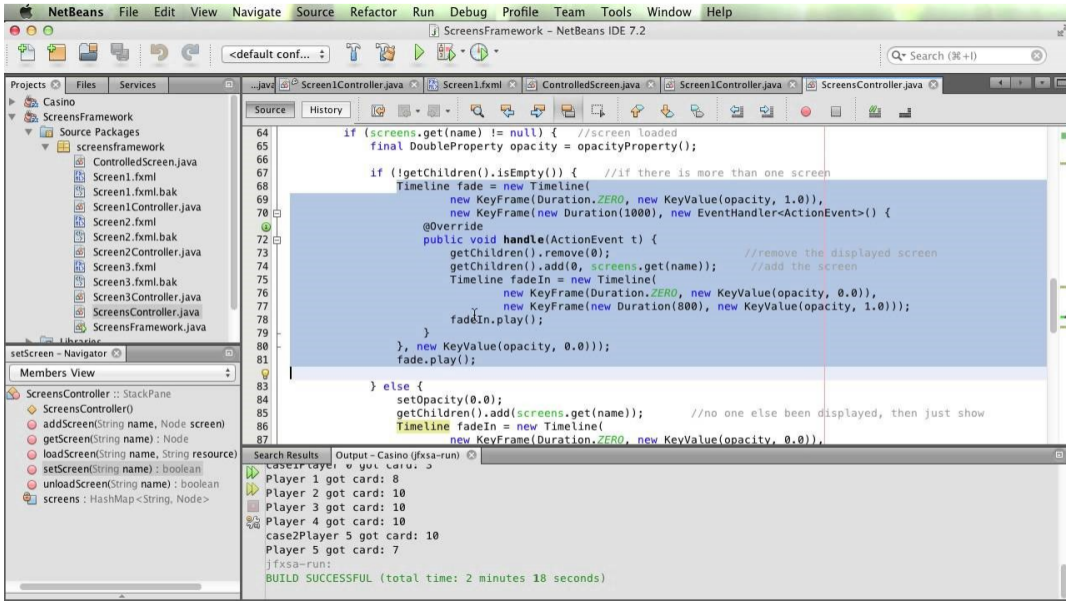


Figure 3.3: fenêtre de programmation Sur Netbeans.

3.5.2 Langage de programmation (Java)

Java est un langage de programmation et une plateforme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et des sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. Java est rapide, sécurisé et fiable. Des ordinateurs portables aux centres de données, des consoles de jeux aux super ordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente sur tous les fronts. C'est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C. Ses caractéristiques ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisée pour le développement d'applications d'entreprises et mobiles.

Environnement Java

Java est un langage interprété, ce qui signifie qu'un programme compilé n'est pas directement exécutable par le système d'exploitation mais il doit être interprété par un autre programme, qu'on appelle interpréteur. Figure 3.4 présente l'architecture exécutable Code java :

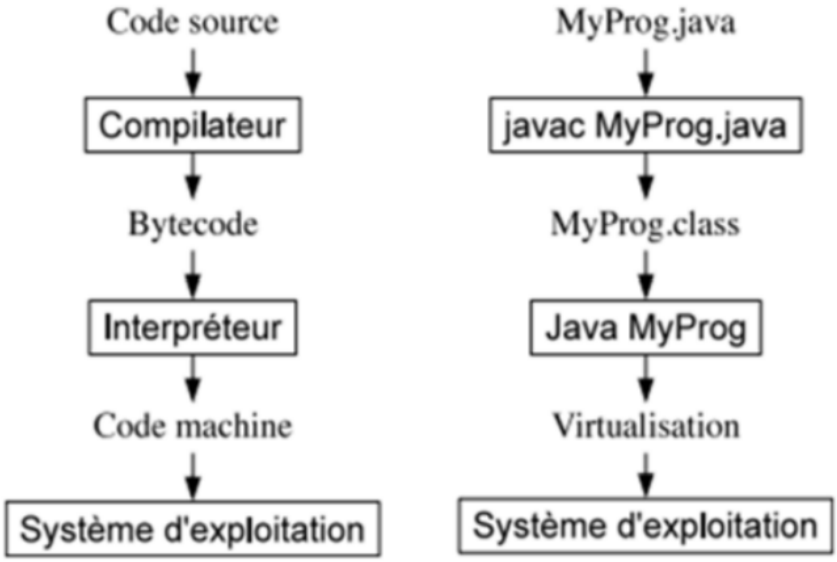


Figure 3.4: architecture exécutable Code java.

3.5.3 JavaFX

JavaFX Intégration

JavaFX est une bibliothèque graphique intégrée dans le JRE et le JDK de Java. Oracle la décrit comme « The Rich Client Platform », c'est-à-dire qu'elle permet de réaliser des interfaces graphiques évoluées et modernes grâce à de nombreuses fonctionnalités, telles que les animations, les effets, la 3D, l'audio, la vidéo, etc. Elle a de plus l'avantage d'être dans le langage Java, qui permet de réaliser des architectures avec des paradigmes objet, et aussi de pouvoir utiliser le typage statique. Dans ce premier tutoriel, nous allons voir ensemble un rapide historique de la bibliothèque pour ensuite découvrir les fondamentaux qui sont les classes « Stage », « Scene », « Application » et le « threading » associé. Cette présentation ne fait pas dans le bling-bling, même si JavaFX est doué pour cela, en préférant se focaliser sur les concepts primordiaux d'une telle bibliothèque. Bien comprendre ces basiques vous aidera bien à commencer pour ensuite pouvoir faire des interfaces de qualité et peut-être spectaculaires

Figure 3.5 Illustre un projet Java FX Main :

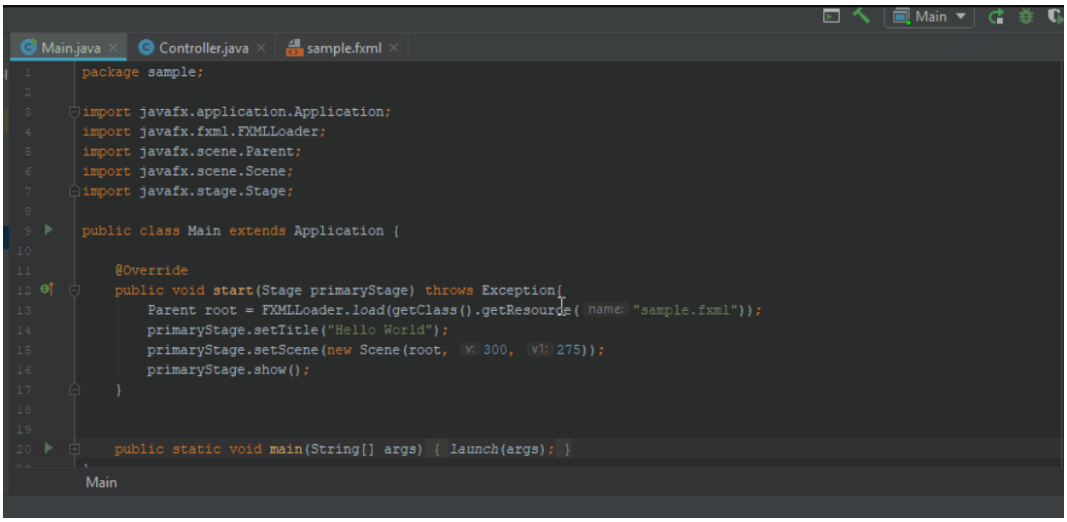


Figure 3.5: projet Java FX Main.

3.5.4 Scene Builder

JavaFXSceneBuilder (Scene Builder) vous permet de concevoir rapidement des interfaces utilisateur d'application JavaFX en faisant glisser un composant de l'interface utilisateur d'une bibliothèque de composants de l'interface utilisateur et en le déposant dans une zone d'affichage du contenu. Le code FXML de la mise en page de l'interface utilisateur que vous créez dans l'outil est automatiquement généré en arrièreplan. Scene Builder peut être utilisé comme un outil de conception autonome, mais il peut également être utilisé avec des IDE Java pour que vous puissiez utiliser l'IDE pour écrire, construire et exécuter le code source du contrôleur que vous utilisez avec l'interface utilisateur de votre application. Bien que SceneBuilder soit plus étroitement intégré à l'EDINetBeans, il est également intégré aux autres EDI Java décrits dans ce document. L'intégration vous permet d'ouvrir un document FXML à l'aide de Scene Builder, d'exécuter les exemples Scene Builder et de générer un modèle pour le fichier source du contrôleur.

3.6 Présentation de L'application

L'application développé en se basent sur les algorithmes génétiques lors de la conception est la suivante :

- Interface d'accueil

La figure 3.6 montre La page d'accueil de l'application

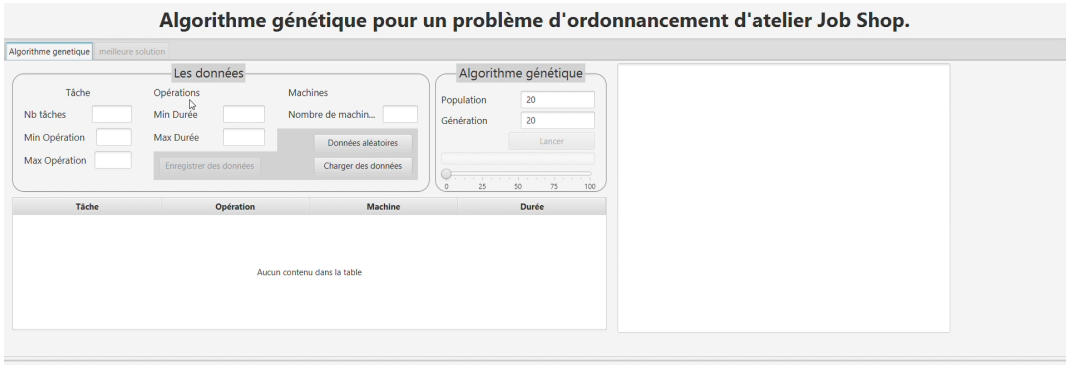


Figure 3.6: Page d'accueil de l'application

- Exécution d'un problème Job Shop

La figure 3.7 montre les résultats et le diagrammes de Gantt obtenues , la valeur de Makspane aprés l'exécution d'un exemple job shop.

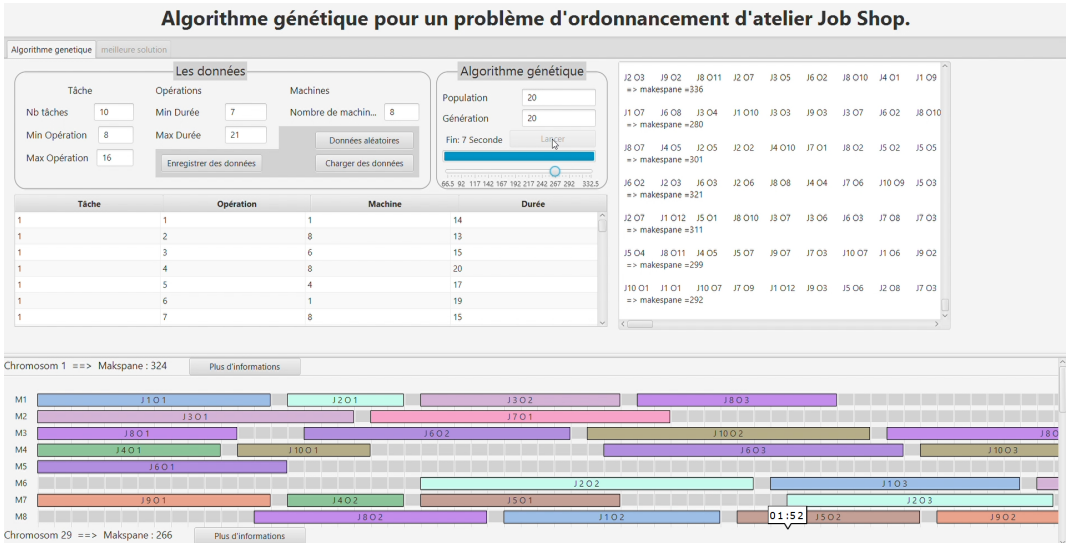


Figure 3.7: Résultats d'exécution d'un problème Job Shop

3.7 Conclusion

Dans ce chapitre, l'apport des algorithmes génétiques c'y mis en évidence de façon concrète et en particulier des méthodes qui se sont développées pour aider les responsables d'atelier en leur donnant non seulement un ensemble d'ordononancement admissible très proche de l'optimum selon le ou les critères choisis par le chef d'atelier. suite aux résultats exposé dans

ce chapitre ,il s'y montré, que les approches étaient capables d'améliorer les résultats trouvés par d'autres méthodes et de s'approcher très efficacement de l'optimum, ceci avec des temps de calcul très courts. Dans tous les cas présentés une très bonne solution à été trouvée au bout de quelques secondes et même dans le cas le plus délicat un bon résultat est trouvé en moins d'une minute. Cette solution est plus souple, plus rapide car elle permet d'intégrer les contraintes de précédence dans la configuration même du chromosome, elle permet aussi de se détacher des autres méthodes car la première population peut etre crée d'une façon totalement aléatoire ce qui permet de faire démarrer l'algorithme.

Conclusion Générale

Dans ce mémoire une étude de la contribution des algorithmes à stratégie d'évolution est donnée pour résoudre conjointement les problèmes d'affectation et d'ordonnancement des ateliers. L'objectif dans ce travail est la résolution approchée du problème d'ordonnancement d'atelier de type job-shop . Le critère considéré est la minimisation de la durée totale de l'ordonnancement (le Cmax ou Makespan).

Les problèmes d'ordonnancement sont extrêmement difficiles à résoudre parce qu'ils appartiennent à la classe dite NP-difficile, ils demandent un espace de recherche hautement combinatoire, de traitement particulièrement complexe. C'est pourquoi les méthodes exactes telles que la branch and bound et la programmation dynamique demandent un temps d'exécution long et/ou des formulations mathématiques complexes, particulièrement quand la taille du problème est importante. Pour palier à ce type de problème, une approche basée sur le principe des algorithmes génétiques est utilisée. La revue de la littérature présentée au chapitre 2, met en évidence les difficultés liées à leur utilisation pour résoudre les problèmes posés. Il apparaît clairement que leur utiliation est conditionnée par certaines modifications à apporter notamment dans la représentation de la solution , ainsi que dans les différents opérateurs génétiques (le croisement et la mutation), d'où le nom des algorithmes à stratégie d'évolution.

Dans ce mémoire , tout d'abord, l'étude de l'ordonnancement dans sa globalité est faite, à savoir ses différentes composantes (tâches, ressources, contraintes, critères), ses différents types d'ateliers (job-shop, flow-shop et open-shop) et les méthodes d'optimisation le plus souvent utilisées pour sa résolution, allant des méthodes exactes (programmation linéaire, programmation dynamique) aux méthodes approchées .Par la suite, l'intérêt s'est porté sur l'étude de la méthode des algorithmes génétiques et son utilisation pour la résolution de problème d'ordonnancement de type job-shop . Pour cela, les différentes types relatives à ce type d' ordonnancement sont présentées et les contraintes pouvant y survenir détaillées, la durée totale de l'ordonnancement (le Cmax ou Makespan) étant le critère à minimiser.

Les résultats obtenus par l'algorithme génétique utilisé, relatifs au problèmes étudiés, ont confirmé la capacité des algorithmes génétiques à s'approcher de la solution optimale. Dans le dernier chapitre, une presentation de problème et la déscrition de solution proposé sont données. Ce chapitre est décomposé en trois parties. Après la presentation de problème, s'est présentée la méthode de resolution, ensuit une application par l'algorithme génétique utilisé de cette solution à des exemples de tailles et de flexibilités différentes, les résultats obtenus montrent l'aboutissement à des solutions admissibles et meilleures .

Bibliography

[1] V Giard. Gestion de la production, 2 ème édition. *Economica, Paris*, 1988.

[2] Georges Javel and Joel Le Bert. *L'Organisation et la Gestion de Production*, volume 138. Masson, 1993.

[3] Mebarek Kebabla. *Utilisation des stratégies métaheuristiques pour l'ordonnancement des ateliers de type Job Shop*. PhD thesis, Batna, Université El Hadj Lakhdar. Faculté des sciences de l'ingénieur, 2008.

[4] Fatma Tangour. *Ordonnancement dynamique dans les industries agroalimentaires*. PhD thesis, Ecole Centrale de Lille, 2007.

[5] BL MacCarthy and Jiyin Liu. A new classification scheme for flexible manufacturing systems. *The International Journal of Production Research*, 31(2):299–309, 1993.

[6] Franck Fontanili. *Intégration d'outils de simulation et d'optimisation pour le pilotage d'une ligne d'assemblage multiproduit à transfert asynchrone*. PhD thesis, Université Paris 13, 1999.

[7] J Carlier, Ph Chretienne, and Claude Girault. Modelling scheduling problems with timed petri nets. In *Advances in Petri Nets 1984*, pages 62–82. Springer, 1985.

[8] Pierre Lopez. *Approche par contraintes des problèmes d'ordonnancement et d'affectation: structures temporelles et mécanismes de propagation*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2003.

[9] Jacques Carlier and Philippe Chrétienne. *Problèmes d'ordonnancement: modélisation, complexité, algorithmes*. Masson, 1988.

[10] C CAUX, G FLEURY, M GOURGAND, and P KELLERT. Rairo. recherche opérationnelle. 1995.

[11] Anant Singh Jain and Sheik Meeran. Deterministic job-shop scheduling: Past, present and future. *European journal of operational research*, 113(2):390–434, 1999.

[12] Ibrahim H Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations research*, 63:511–623, 1996.

[13] François Blondel. *Gestion de la production: Comprendre les logiques de gestion industrielle pour agir Ed. 4*. Dunod, 2005.

[14] Imen DRISS. *Analyse d'un système job shop aspect ordonnancement*. PhD thesis, Université de Batna 2, 2016.

[15] Jin-Kao Hao, Philippe Galinier, and Michel Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 13(2):283–324, 1999.

[16] Prasanna Jog, Jung Y Suh, and Dirk Van Gucht. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 110–115, 1989.

[17] Timothy Starkweather, S McDaniel, Keith E Mathias, L Darrell Whitley, and C Whitley. A comparison of genetic sequencing operators. In *ICGA*, pages 69–76, 1991.

[18] Imed Kacem. *Ordonnancement multicritère des job-shops flexibles: formulation, bornes inférieures et approche évolutionniste coopérative*. PhD thesis, Lille 1, 2003.

[19] Khaled Mesghouni. *Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production*. PhD thesis, Lille 1, 1999.

[20] Hisao Ishibuchi, Naohisa Yamamoto, Tadahiko Murata, and Hideo Tanaka. Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems. *Fuzzy Sets and systems*, 67(1):81–100, 1994.

[21] Thomas Vallée and Murat Yildizoğlu. Présentation des algorithmes génétiques et de leurs applications en économie. *Revue d'économie politique*, pages 711–745, 2004.

[22] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the 1996 ACM symposium on Applied Computing*, pages 289–298, 1996.

[23] Emna Gargouri. *Ordonnancement coopératif en industrie agroalimentaire*. PhD thesis, Lille 1, 2003.

[24] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[25] Jan K Lenstra and AHG Rinnooy Kan. Computational complexity of discrete optimization problems. In *Annals of discrete mathematics*, volume 4, pages 121–140. Elsevier, 1979.

[26] James R Jackson et al. An extension of johnson's results on job idt scheduling. *Naval Research Logistics Quarterly*, 3(3):201–203, 1956.

[27] Sheldon B Akers Jr. A graphical approach to production scheduling problems. *Operations Research*, 4(2):244–245, 1956.

[28] Peter Brucker. An efficient algorithm for the job-shop problem with two jobs. *Computing*, 40(4):353–359, 1988.

[29] Wieslaw Kubiak, Suresh Sethi, and Chelliah Sriskandarajah. An efficient algorithm for a job shop problem. *Annals of Operations Research*, 57(1):203–216, 1995.

[30] Vadim G Timkovsky. A polynomial-time algorithm for the two-machine unit-time release-date job-shop schedule-length problem. *Discrete Applied Mathematics*, 77(2):185–200, 1997.

[31] Peter Brucker. A polynomial algorithm for the two machine job-shop scheduling problem with a fixed number of jobs. *Operations-Research-Spektrum*, 16(1):5–7, 1994.