



UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHÉMATIQUES 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 Décisionnel et Optimisation

Par : KACIMI EL HASSANI Mounya

SUJET

**Etude comparative des algorithmes génétiques
pour un problème d'ordonnancement d'atelier
M-Machines identiques en parallèles**

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

Phd. BOURAHLA Moustafa

Université de M'sila

Président

Dr. BOUNIF Mohamed Elhadi

Université de M'sila

Rapporteur

Dr. LOUNASS Bilal

Université de M'sila

Examineur

Promotion : 2018 /2019

Dédicaces

A mes très chers parents.

Remerciements

On remercie Dieu le tout puissant, qui nous a donné la force et la patience pour l'accomplissement de ce travail.

*Nous remercions en particulier Mr. **BOUNIF Mohamed Elhadí**, pour l'honneur qu'il nous a fait de bien vouloir nous encadrer, et pour les conseils donnés lors de la réalisation de ce travail.*

On adresse nos remerciements au membre de jury pour avoir accepté de nous prêté de leur attention et évaluer notre travail.

Table des Matières

Introduction générale.....	1
-----------------------------------	----------

CHAPITRE 1 LA PRODUCTION ET GESTION DE LA PRODUCTION

Introduction.....	4
1. La production.....	4
1.1 Définition.....	4
1.2 Objectifs associés.....	4
1.3 Décomposition du système de production.....	5
2. Gestion de la production.....	6
2.1 Définition.....	6
2.2 Rôle et l'importance	6
2.3. Décomposition hiérarchique de la gestion de production.....	7
3. Les problèmes des systèmes de production.....	8
3.1 Problème d'ordonnancement	8
3.2 Problème d'optimisation.....	9
Conclusion.....	10

CHAPITRE 2 ORDONNANCEMENT

Introduction.....	12
1. Problème d'ordonnancement.....	12
1.1 Notion de base et définition.....	12
1.2 Notion de tâche.....	12
1.3 Notion de ressource.....	13
1.4 Notion d'objectifs et de critères.....	14
1.5 Notion de contrainte.....	15
2. Les problèmes d'ordonnancement d'atelier.....	15
2.1 Classification des problèmes d'ordonnancement	16

3. Caractéristiques générales des ordonnancements.....	19
3.1. Ordonnements statique et dynamique	19
3.2. Ordonnements admissibles	19
3.3. Ordonnements actifs et semi-actifs	19
3.4. Ordonnement sans retard	19
3.5. Ordonnement préemptif et non préemptif	19
4. Ordonnement par l'approche proposée.....	20
5. Les méthodes de résolution du problème d'ordonnement.....	21
5.1 Méthodes exactes.....	22
5.2 Méthodes approchées.....	22
5.2.1 Les heuristiques.....	22
5.2.2 Les méta-heuristiques.....	23
Conclusion.....	24

CHAPITRE 3 ALGORITHMES GENETIQUES ET OPTIMISATION MULTIOBJECTIF

Introduction.....	26
1. Algorithmes génétiques mono-objectif.....	26
1.1 Principes de base des algorithmes génétiques	26
1.2 Codage et population initiale	27
1.3 La fonction d'évaluation.....	28
1.4 Opérateur de sélection	28
1.5 Opérateur de croisement	29
1.6 Opérateur de mutation	30
2. L'optimisation multi-objectif.....	31
2.1. Dominance au sens de Pareto	31
2.2. Pareto optimum.....	31
3. Optimisation multi-objectif par algorithmes génétiques.....	32
3.1. Méthodes agrégées	33

3.2. Méthodes non-agrégées non-Pareto	33
3.3. Méthodes Pareto	34
3.1. Non-dominated Sorting Genetic Algorithm II (NSGA II).....	34
3.2. Strength Pareto Evolutionary Algorithm II (SPEA II)	37
3.3. Pareto Archived Evolution Strategy (PAES).....	39
Conclusion.....	42
 CHAPITRE 4 ETUDE EXPERIMENTALE ET COMPARATIVE	
Introduction.....	44
1. Outils et environnement de développement.....	44
2. Quelque exemple sur l'implémentation.....	45
3. Comparaison.....	56
4. Quelques interfaces de l'application.....	58
Conclusion.....	59
Conclusion générale.....	60

Liste des Figures

Figure 1.1 Les sous-systèmes constituant le système de production	5
Figure 1.2 Les sous fonctions de l'ordonnancement dans l'atelier.....	9
Figure 2.1 Typologie des problèmes d'ordonnancement par les ressources.....	14
Figure 2.2 Les différents types des problèmes d'ordonnancement d'atelier.....	16
Figure 2.3 Ordonnancement sur machine unique.....	17
Figure 2.4 Modèle d'une Machine parallèle.....	17
Figure 2.5 Ateliers à cheminement unique (Flow-shop).....	17
Figure 2.6 Ateliers à cheminement multiple (Job-hop).....	18
Figure 3.1 : Exemples de codage par valeurs.....	28
Figure 3.2 : Croisement à un site.....	30
Figure 3.3 montre un croisement à deux points.....	30
Figure 3.4 Mutation d'un chromosome	30
Figure 3.5 montre le front de pareto.....	32
Figure 3.6 Vector Evaluated Genetic Algorithm (VEGA).....	34
Figure 3.7 le fonctionnement du NSGA II (exemple 3 fronts admis)	36
Figure 3.8 Distance de Crowding (les points noirs sont des Solutions appartenant au même front).....	37
Figure 3.8 Fonctionnement du SPEAII.....	38
Figure 3.9 Fonctionnement du PAES.....	41
Figure 4.1 Exemple de Diagramme de Gant obtenu par NSGAII de problème 4x7.....	46
Figure 4.2 Exemple de Diagramme de Gant obtenu par SPEAII de problème 4x7.....	47
Figure 4.3 Exemple de Diagramme de Gant obtenu par PAES de problème 4x7.....	48
Figure 4.4 Exemple de Diagramme de Gant obtenu par NSGA I de problème 2x8.....	49
Figure 4.5 Exemple de Diagramme de Gant obtenu par SPEA II de problème 2x8.....	50
Figure 4.6 Exemple de Diagramme de Gant obtenu par PAES de problème 2x8.....	51
Figure 4.7 Exemple de Diagramme de Gant obtenu par NSGAII de problème 3x9.....	53
Figure 4.8 Exemple de Diagramme de Gant obtenu par SPEA II de problème 3x9.....	54
Figure 4.9 Exemple de Diagramme de Gant obtenu par PAES de problème 3x9.....	55
Figure 4.10 Diagramme obtenu par Exemple 2.....	55
Figure 4.11 Diagramme obtenu par Exemple 4.....	56
Figure 4.12 Diagramme obtenu par Exemple 2.....	56
Figure 4.13 Diagramme obtenu par Exemple 4.....	56
Figure 4.14 Diagramme obtenu par Exemple 4.....	56
Figure 4.15 Interface principale de l'application.....	57
Figure 4.16 Exécution de premier exemple de NSGA II.....	57

Liste des Tables

Table 4.1 La table des tâches Exemple 1.....	45
Table 4.2 La table de la solution avec NSGA II Exemple 1.....	45
Table 4.3 La table de la solution avec SPEA II Exemple 1.....	46
Table 4.4 La table de la solution avec PAES Exemple 1.....	47
Table 4.5 La table des tâches Exemple 2.....	48
Table 4.6 La table de la solution avec NSGA II Exemple 2.....	49
Table 4.7 La table de la solution avec SPEA II Exemple 2.....	50
Table 4.8 La table de la solution avec PAES Exemple 2.....	51
Table 4.9 La table des tâches Exemple 3.....	52
Table 4.10 La table de la solution avec NSGA II Exemple 3.....	52
Table 4.11 La table de la solution avec SPEA II Exemple 3.....	53
Table 4.12 La table de la solution avec PAES Exemple 3.....	54

Liste des Algorithmes

Algorithme 3.1 L'algorithme de NSGA II.....	35
Algorithme 3.2 L'algorithme de SPEA II.....	37
Algorithme 3.3 L'algorithme de PAES.....	39

INTRODUCTION GENERALE

De nos jours, les industries sont dominées par une très forte concurrence en terme de production des produits personnalisés alliant qualité et prix, dans des délais restreints.

Dans ce marché concurrentiel -où la clientèle est très exigeante- est apparue la nécessité de concevoir et mettre en place des systèmes de production performants et passer obligatoirement par une gestion de production plus fiable. Dans ce contexte, Parmi les fonctions de la gestion de production, la résolution du problème d'ordonnancement.

L'ordonnancement s'identifie, généralement, à un problème d'optimisation combinatoire dans lequel la recherche de solutions optimales nécessite la mise au point des méthodes exactes qui requièrent un grand nombre de calculs exponentiels. De ce fait, des méthodes approchées ont été proposées pour résoudre ce problème en un temps raisonnable. Parmi ces méthodes, on a les méthodes dites "Méta-heuristiques" qui ont démontré leur efficacité dans de nombreuses applications telles que les Algorithmes Génétiques qui appartiennent aux méthodes évolutives.

Les Algorithmes génétiques sont une méthode de résolution des problèmes multi-objectifs qui consistent à optimiser plusieurs problèmes en même temps et trouver des compromis et non-pas une seule solution finale. Ces compromis représentent des choix qui dépendent du système étudié.

La multitude des modèles des algorithmes génétiques offre un choix, mais parfois ce choix est difficile car il existe plusieurs types de problèmes multi-objectifs. Notre travail consiste à départager entre quelques types d'algorithmes génétiques multicritères en menant une étude comparative sur des critères de qualité (la durée totale d'exécution «Cmax », la somme des retards « $\sum T$ »), et de coût (temps d'exécution, espace mémoire) dans un problème m-machines identiques en parallèles.

Dans le premier chapitre, on commence par une vue d'ensemble sur la production, la gestion de la production. Ainsi que les problèmes liés aux systèmes de production.

Dans Le deuxième chapitre, nous allons aborder les problèmes d'ordonnements incluant entre autre les différents éléments liés à ce dernier, la typologie des problèmes d'ordonnement ainsi que les différents types, d'ateliers ajouter à cela le développement des modèles mathématiques ainsi que les différents méthodes de résolutions.

Le chapitre trois, est consacré aux algorithmes génétiques ainsi que l'optimisation multi-objectif à travers les algorithmes génétiques.

Dans le quatrième chapitre et le dernier, nous allons présenter l'implémentation ainsi que la comparaison des algorithmes génétiques utilisées et plus particulièrement NSGA II, SPEA II, PAES. Enfin, nous clôturons par une conclusion.

CHAPITRE 1

LA PRODUCTION ET GESTION

DE LA PRODUCTION

Introduction

Dans nos jours de globalisation économique et de révolutions technologiques, la fonction de production devient de plus en plus stratégique, dans le mesure, où elle oblige à la maîtrise d'un environnement instable comme elle doit atteindre un objectif mixte de coût-qualité-délais, flexibilité, qui caractérise la meilleure réponse de l'entreprise au consommateur.

Dans ce chapitre introductif on présente les principaux traits de la production, on utilisera trois sections : la première vise à proposer une définition aussi complète que possible de ce qui est la production, la seconde s'intéressera à la gestion de production. Enfin dans la dernière section on va clore ce chapitre en démentant les problèmes des systèmes de production qui inclut le problème d'ordonnancement et le problème d'optimisation.

1 La production

1.1 Définition.

La production est le processus conduisant à la création de produits par l'utilisation et la transformation de ressources [1].

Le système de production (SP) représente l'ensemble du processus et des moyens de production que l'entreprise met en place afin de transformer des matières premières ou des produits semi finis, en produits finis prêts à être commercialisés[2].

Les systèmes de production industrielle se sont considérablement diversifiés et compliqués. En effet, ils peuvent se décomposer en plusieurs sous-systèmes, qui s'intègrent en vue d'assurer la pérennité et la compétitivité de l'entreprise [3].

Alors la production est la création des produits, le système de production est la séquence des opérations nécessaires pour réaliser des produits finis.

1.2 Objectifs associés

L'objectif principal des systèmes de production est de produire un bien économique. Cependant, la fonction de production se doit de satisfaire d'autres objectifs intermédiaires que sont :

- **En termes de quantités produites ;** Il faut que la fonction de production permette de satisfaire la demande des clients. Pour pouvoir la réaliser, l'entreprise doit mener des actions pour maintenir les capacités productives ou bien mettre au point des plans d'investissement en capacité.
- **En termes de qualité ;** Afin de satisfaire les besoins de la clientèle et s'assurer un niveau de compétitivité, il faudrait que les biens économiques produits soient de bonne

qualité.

- **En termes de coût** ; Il faut que l'entreprise garantisse sa compétitivité par des coûts de production les plus faibles possibles.
- **En termes de délais** ; Cela consiste à respecter des délais raisonnables, conformes avec le niveau de demande à laquelle doit faire face l'entreprise. Cela permettra d'éviter le stockage des biens finaux et le stockage des produits finis.
- **En termes de flexibilité** ; C'est-à-dire la capacité du Systèmes de Production à pouvoir s'adapter aux variations de la demande [2].

1.3 Décomposition du système de production

Classiquement, un système de production peut se décomposer en trois sous-systèmes : le système physique de production, le système de décision et le système d'information [4] [5].

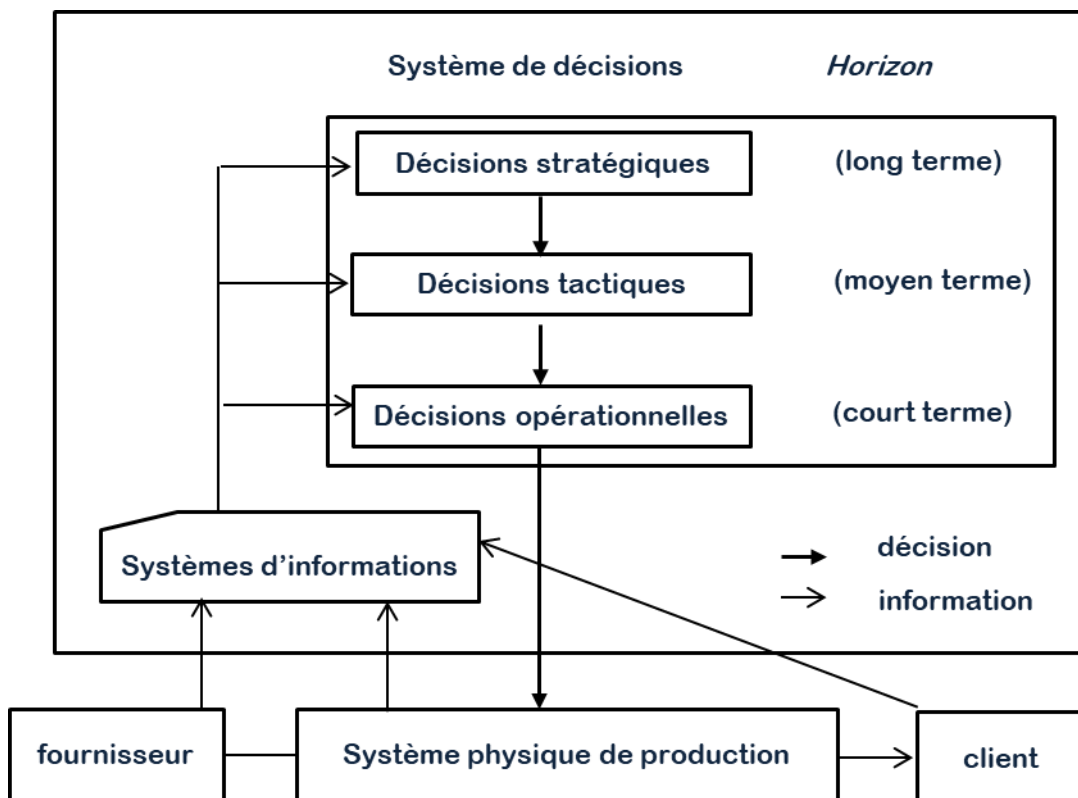


Figure 1.1 Les sous-systèmes constituant le système de production [5]

- **Le système physique de production (le système de fabrication)**: Transforme les matières premières ou composants 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...) [3].

- **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 :** à l'interface entre les systèmes de décision et de production; à l'intérieur du système de décision, pour la gestion des informations utilisées lors de prises de décisions; et à l'intérieur du système physique de production. Son rôle est de collecter, stocker et transmettre des informations de différents types [3].

2 Gestion de la production

2.1 Définition

La gestion de la production est « la fonction qui permet de réaliser les opérations de production en respectant les conditions de qualité, délai, coût qui résultent des objectifs de l'entreprise » [6].

La gestion de la production est la mise en application des méthodes et techniques dans le but d'accomplir la transformation des matières en produits finis. Elle se résume en la combinaison de ressources, parmi lesquelles les moyens matériels (les machines), les moyens humains (le personnel par qualification) et les matières (matières premières, matières consommables) dans un planning pour but d'assurer la fabrication du produit en qualité et en quantité définie [7].

2.2 Rôle et l'importance

Le rôle de la gestion de production est, d'organiser et de piloter le fonctionnement des processus physiques mis en œuvre dans l'entreprise, afin d'assurer une meilleure utilisation des moyens humains, physiques et technologiques disponibles, et de satisfaire au mieux l'objectif global de production défini en terme de quantités à fabriquer avec une qualité demandée, et des délais à respecter[7].

Les contraintes rencontrées sont de divers ordres [7] :

- **Financières** (produire à un coût optimal), coût des matières et consommables, coût de stockage des encours et des produits semi-ouvrés, coût de gestion des magasins, coût des heures de travail supplémentaires, coût des arrêts faisant partie intégrante du coût de revient, maîtriser ces derniers est aussi une garantie pour la commercialisation des produits finis,
- **Temporelles** (produire dans les délais, assurer une livraison juste à temps), éviter les ruptures de stocks, éviter le gonflement des stocks de produits finis. Car cela a une incidence directe sur la satisfaction de la clientèle (pertes de commandes) ou sur le coût de revient du

produit finis dû aux couts supplémentaires du stockage.

- **Mécaniques** (maintenance préventives et gestion des temps d'arrêt), anticiper sur les pannes et prévoir des solutions alternatives en cas d'arrêt d'une machine,
- **Qualité** (produire avec le moins de défaut possible, le moins de déchets), un produit de bonne qualité participe à la fidélisation de la clientèle, véhicule l'image de marque de l'entreprise.
- **Planification** : définir un plan de production, définir les gammes opératoires et d'ordonnancer les opérations, et enfin de gérer la répartition des tâches durant tout le processus de fabrication.

2.3. Décomposition hiérarchique de la gestion de production

On distingue trois niveaux hiérarchiques de la gestion de production (Figure 1.1) : stratégique, tactique et opérationnel [6].

- **Le niveau stratégique**

Ce niveau trace la politique à long terme de l'entreprise (à un horizon de plus de deux ans). Cette politique porte essentiellement sur la gestion des ressources durables. Afin que celle-ci soient en mesure d'assurer la pérennité de l'entreprise [6].

- **Le niveau tactique**

Ce niveau relie les deux niveaux stratégique et opérationnel, il porte sur les décisions à moyen terme. Le but est d'assurer une production satisfaisante à la demande en minimisant les couts, tout en respectant le plan tracé par le niveau stratégique de l'entreprise [6].

- **Le niveau opérationnel**

Ce niveau porte sur les décisions à court terme. Il s'agit d'une gestion quotidienne. L'objectif visé est de répondre aux contraintes établies par les configurations et les politiques de planification choisies aux niveaux stratégiques et tactiques, comme par exemple: Ordonnancement et pilotage en temps réel des systèmes de production [6].

3 Les problèmes des systèmes de production

L'objectif des systèmes de production (SP) est de satisfaire le client et pour y arriver il est impératif de mieux gérer la production, ce qui implique un ordonnancement pouvant aboutir à des échéances acceptables. De ce fait la complexité des SP peut se résumer en deux principaux problèmes [28]: problème d'ordonnancement et problème d'optimisation.

3.1 Problème d'ordonnancement

La gestion de production décompose les décisions en trois niveaux : Stratégique, tactique et opérationnel (pilotage et suivi quotidien des flux de matières et du travail). Cette hiérarchisation se traduit par une échelle de responsabilité (direction, cadre, agent ...) [3].

Dans la mesure où des événements aléatoires telles que des pannes ou des commandes imprévues, qui peuvent survenir à tout moment, il est nécessaire de recalculer fréquemment l'ordonnancement, il est difficile de résoudre le problème d'ordonnancement au niveau supérieur (stratégique). Donc, il est résolu au niveau inférieur (opérationnel) [3].

L'ordonnancement traduit l'ensemble des décisions de fabrication définies par le programme directeur de production en instructions d'exécution détaillées destinées au lancement au contrôle et au pilotage à court terme de l'activité des postes de travail [7].

A l'issue de la fonction ordonnancement, on obtient un calendrier qui assure une affectation optimale des tâches sur les ressources disponibles, en précisant la durée et la date d'exécution de chacune d'elles, tout en respectant certaines contraintes et en optimisant un ou plusieurs fonctions objectifs [7].

La fonction ordonnancement se décompose en trois sous fonctions [7] :

- L'élaboration des ordres de fabrication (OF) : c'est-à-dire transformation des informations du programme directeur de production (suggestion de fabrication) en ordres de fabrication.
- L'élaboration du planning d'atelier : ceci dit, 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 : ont appliquées (voir Figure 1.2):

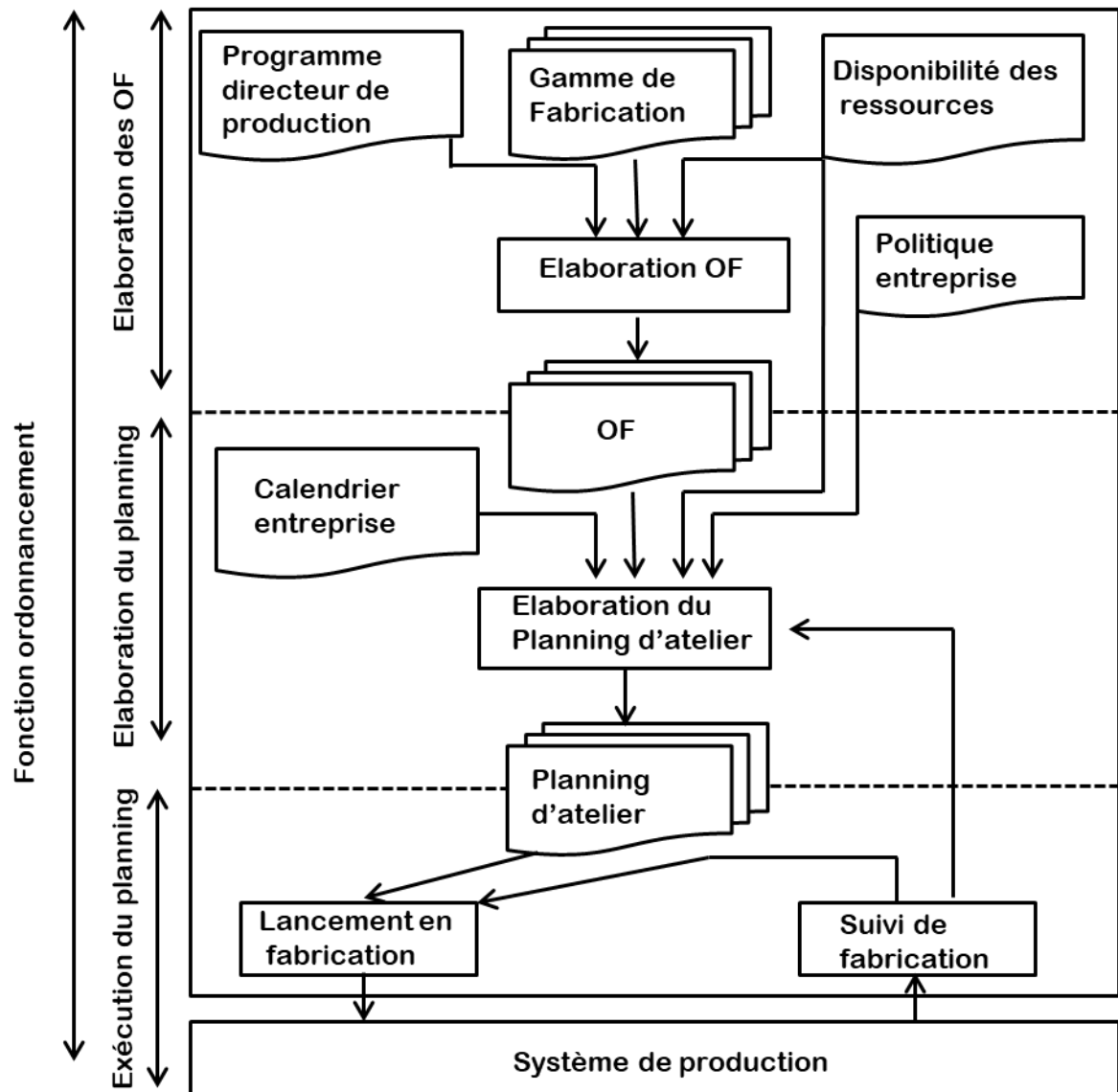


Figure 1.2 Les sous fonctions de l'ordonnancement dans l'atelier [7].

3.2 Problème d'optimisation

Optimiser c'est tout d'abord mesurer avant d'agir sur les points critiques. Pour un système de production il faut mettre en place des indicateurs liés à la productivité en fonction des données, aux arrêts machines dont les résultats révéleront les véritables gisements de productivité.

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. L'approche par optimisation exige de satisfaire certains critères d'évaluations numériques. On cherchera donc à minimiser ou maximiser certains critères [8].

Conclusion

Dans ce chapitre nous avons résumé les différents notions liés aux systèmes de production on a mentionné ses différentes composantes et objectifs, Le deuxième point exposé dans ce chapitre, est la gestion de production et faire valoir l'importance du problème d'ordonnancement au niveau opérationnel .Enfin on a présenté les problèmes des systèmes de production.

Dans le chapitre suivant, nous présentons la notion de l'ordonnancement, le modelé mathématique et les différents modèles de résolutions des problèmes d'ordonnancement.

CHAPITRE 2

ORDONNANCEMENT

Introduction

L'ordonnancement est un secteur de la recherche opérationnelle et de la gestion de la production qui consiste à attribuer plusieurs tâches à des moyens de production dans le but de réaliser des produits (jobs) tout en respectant les contraintes de production et en optimisant certains critères pour améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison.

Dans ce chapitre Nous présentons tout d'abord ce qu'est un problème d'ordonnancement en évoquant notamment les différents éléments qui le déterminent, ensuite nous allons représenter les différents types des problèmes d'ordonnancement et ses caractéristiques générales. la quatrième section sert à présenter l'ordonnancement par l'approche proposée. Enfin dans la dernière section on va clôturer ce chapitre en expliquant les différentes méthodes de résolution du problème d'ordonnancement.

1 Problème d'ordonnancement

1.1 Notion de base et définition

L'ordonnancement se situe exactement dans la phase planification. Il réalise le suivi opérationnel du projet : gestion de ressources, suivi de l'avancement, lancement des activités. Techniquement, ordonnancer un projet consiste à programmer dans le temps l'exécution des tâches, tout en respectant les contraintes et de manière à optimiser les critères de performance retenus [9].

D'une façon plus discrète « On parle de problème d'ordonnancement lorsqu'on doit déterminer les dates de début et les dates de fin des tâches, alors qu'on réserve le terme de problème de séquençement au cas où l'on cherche seulement à fixer un ordre relatif entre les tâches qui peuvent être en conflit pour l'utilisation des ressources. Un ordonnancement induit nécessairement un ensemble unique de relations de séquençement [9]. Dans un problème d'ordonnancement interviennent deux notions fondamentales: les tâches et les ressources. Une ressource est un moyen, technique ou humain, dont la disponibilité limitée ou non est connue a priori. Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource [7].

1.2 Notion de tâche

Une tâche i est une entité élémentaire de travail localisée dans le temps par une date de début s_i ou de fin c_i , dont la réalisation nécessite une durée $p_i = c_i - s_i$, et qui consomme des moyens k . [10]. 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.

1.3 Notion de ressource

Une ressource k est donc, un moyen humain ou technique qui est utilisé dans la réalisation d'une tâche. Elle est disponible en quantité limitée. La disponibilité est généralement exprimée par une capacité propre à chaque ressource k notée Q_k ($Q_k \geq 1$) [9].

On distingue deux types de ressources: les ressources renouvelables et les ressources consommables [9] :

- **Une ressource est consommable** si, après avoir été allouée à une tâche, elle n'est plus disponible pour les tâches suivantes. Le cas pour l'argent, la matière première, etc.
- **Une ressource est renouvelable** si, après avoir été allouée à une tâche, elle redevient disponible après la fin de cette tâche pour les tâches suivantes. C'est le cas pour les machines, les fichiers, le personnel, etc.

On distingue par ailleurs, principalement dans le cas de ressources renouvelables, les ressources disjonctives (ou non partageables) qui ne peuvent exécuter qu'une tâche à la fois (machine, robot, etc.) et les ressources cumulatives (ou partageables) qui peuvent être utilisées par plusieurs tâches en même temps (équipe d'ouvriers, poste de travail, etc.) [9].

Les problèmes d'ordonnancement à ressources disjonctives couvrent une classe importante d'applications qu'on appelle les problèmes d'atelier ou de machines [9].

La nature des ressources prises en considération permet de dresser une typologie des problèmes d'ordonnancement (Figure 2.1) [9].

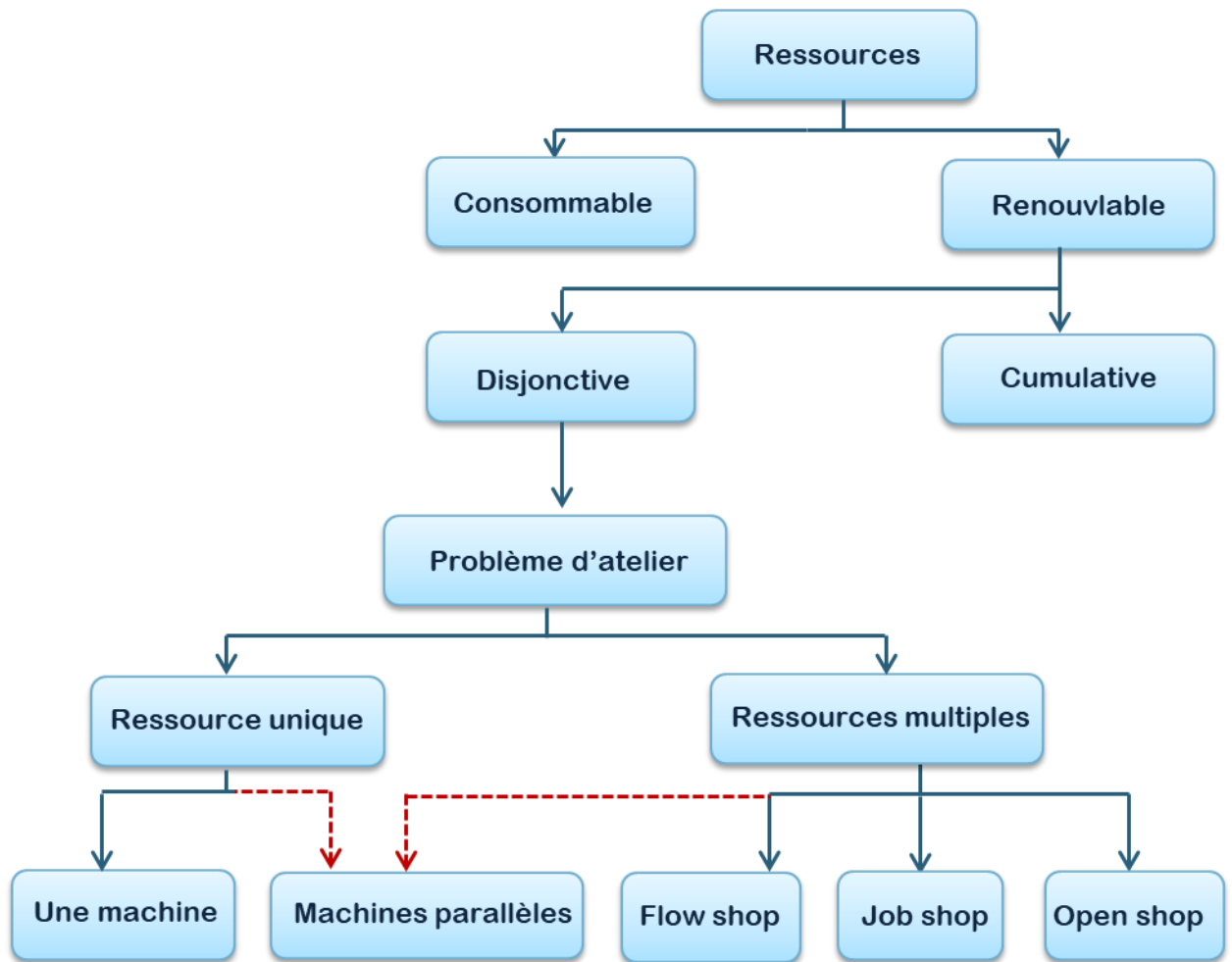


Figure 2.1 Typologie des problèmes d'ordonnancement par les ressources [9].

1.4 Notion d'objectifs et de critères

Chaque ordonnancement est guidé par un ou plusieurs objectifs qu'il doit chercher leur optimisation. Les objectifs que doit satisfaire un ordonnancement sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement donné [4]:

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

L'ordonnancement est la programmation dans le temps de l'exécution d'une série de tâches (ou activités, opérations) sur un ensemble de ressources physiques (humaines et techniques),

en cherchant à optimiser certains critères, financiers ou technologiques, et en respectant les contraintes de fabrication et d'organisation.

1.5 Notion de contrainte

Une contrainte est une condition que doit respecter le plan de travail. Elle limite le degré de liberté d'exécution de processus [11]. Cette condition est liée aux opérations entre elles, aux ressources, etc.

- **Les contraintes de localisation temporelles** Elles représentent les bornes de l'intervalle de temps durant lequel une tâche doit être traitée. L'intervalle est représenté par la date de début d'exécution et la date de fin d'exécution [29].

$$ri \leq si \text{ et } ci \leq di$$

- **Les contraintes de succession ou de précédence** Elles prennent en compte la succession des opérations de la gamme opératoire. Les contraintes indiquent que le début d'exécution d'une tâche j doit être après la date de fin d'exécution d'une autre opération i qui la précède, d'où [29]:

$$Ti \geq si + pi$$

- **Les contraintes de ressources** Ces contraintes de limitation de ressources renouvelables expriment la nature de la quantité des moyens utilisés par les tâches. Il y a deux types de contraintes suivant la nature des ressources [29]:
 - **Contraintes disjonctives** : Elles obligent à réaliser toute paire de tâches sur des intervalles de temps disjoints. Soit une tâche i , elle s'exécute soit avant j , soit après j .
 - **Contraintes cumulatives** : Elles impliquant la limitation du nombre de tâches à réaliser en parallèle.

2 Les problèmes d'ordonnancement d'atelier

Dans un problème d'atelier, une pièce doit être usinée ou assemblée sur différentes machines. Chaque machine est une ressource disjonctive, c'est-à-dire qu'elle ne peut exécuter qu'une tâche à la fois, et les tâches sont liées exclusivement par des contraintes d'enchaînement. Plus précisément, les tâches sont regroupées en n entités appelées travaux ou lots. Chaque lot est constitué de m tâches à exécuter sur m machines distinctes, et dans le cas des problèmes d'atelier, une tâche est une opération, une ressource est une machine et chaque opération nécessite pour sa réalisation une machine. Dans le modèle de base de l'ordonnancement d'atelier, l'atelier est constitué de m machines, n travaux (jobs), disponibles à la date 0, doivent être réalisés, un travail i est constitué de n_i opérations, l'opération j du

travail i est notée (i,j) avec $(i, 1) < (i, 2) < \dots < (i, n_i)$ si le travail i possède une gamme ($A < B$ signifie A précède B). Une opération (i,j) utilise la machine $m_{i,j}$ pendant toute sa durée $p_{i,j}$ et ne peut être interrompue [12].

Un atelier se définit par le nombre de machines qu'il contient et par son type. Une classification peut exister selon le nombre des machines et l'ordre d'utilisation des machines, pour réaliser un travail (par exemple fabrication d'un produit qui dépend de la nature de l'atelier) [12].

Selon les caractéristiques des machines on peut distinguer plusieurs types des problèmes [12].

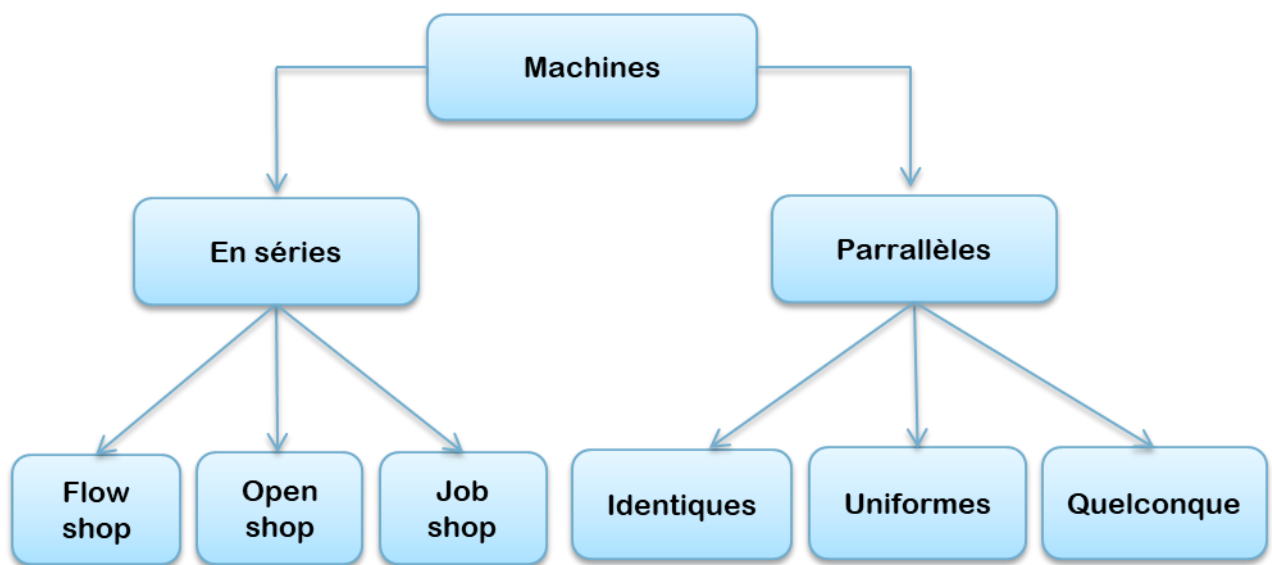


Figure 2.2 Les différents types des problèmes d'ordonnancement d'atelier [12].

2.1 Classification des problèmes d'ordonnancement

Dans la littérature, il existe parmi les notations utilisées pour classifier les problèmes d'ordonnancement la notation de Lawler [13]. Cette notation a trois champs $\alpha | \beta | \gamma$. Dans ce qui suit, nous allons énumérer tout ce que décrit chaque champ de cette notation.

- **Champ α** : Le champ α représente les environnements machines possibles, que sont [14] :
 - **Une machine ($\alpha = 1$)** : Il y a une seule machine dans le système. Chaque tâche est constituée d'une seule opération. Dans les problèmes à une machine, il faut déterminer une séquence des tâches optimisant le critère de performance. Les résultats obtenus pour les problèmes à une machine sont souvent utilisés pour résoudre des problèmes plus généraux.

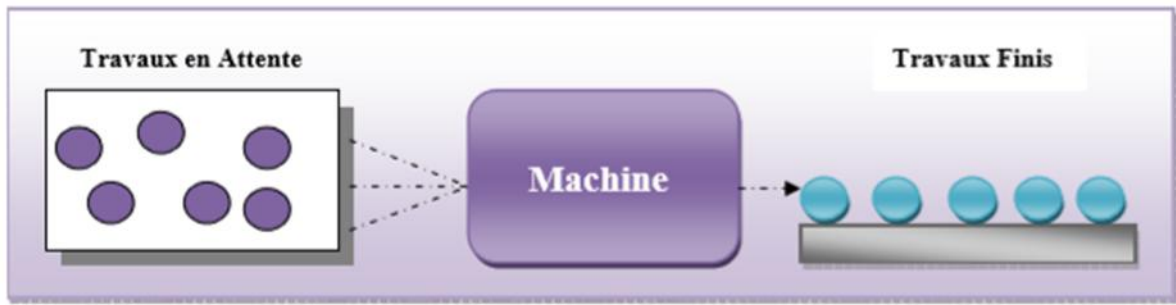


Figure 2.3 Ordonnancement sur machine unique [21].

- **Machines parallèles** : Le système est composé de m machines parallèles. Chaque tâche est constituée d'une seule opération et elle peut être traitée sur n'importe laquelle des m machines. Dans les problèmes à machines parallèles, il y a deux types de décisions à prendre: affecter les tâches aux machines et décider de l'ordre des tâches sur une même machine [14] .

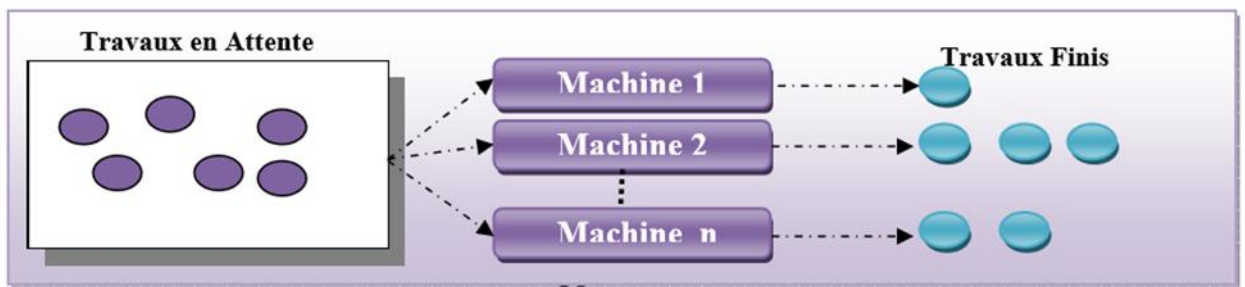


Figure 2.4 Modèle d'une Machine parallèle [21].

- **Flow shop ($\alpha = F_m$)** : Un atelier à cheminement unique est un atelier où le processus d'élaboration de produits est dit « linéaire », c'est-à-dire lorsque les étapes de transformation sont identiques pour tous les produits fabriqués. Si on trouve plusieurs exemplaires identiques et parallèles de la même machine, l'atelier devient Flow-Shop Hybride [11].

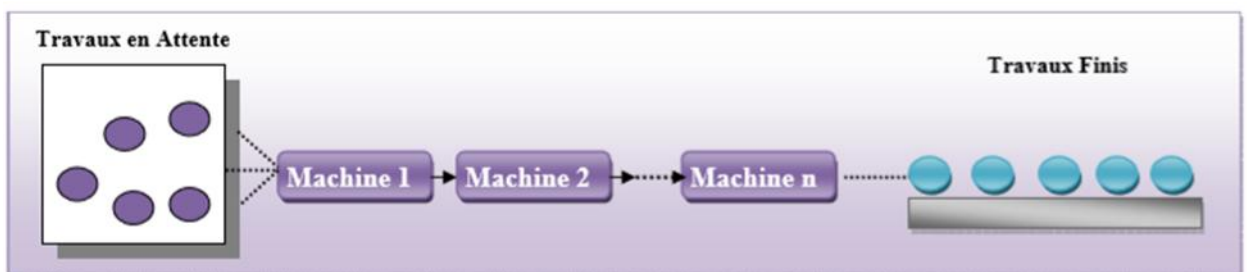


Figure 2.5 Ateliers à cheminement unique (Flow-shop) [21].

- **Job shop ($\alpha = J_m$)** : Dans un Job shop à m machines, chaque tâche a un cheminement

prédéterminé à suivre. Elle peut visiter certaines machines plus d'une fois et elle peut ne pas visiter certaines machines du tout. L'ordre de visite des machines pour une tâche dépend de sa gamme (l'ordre des opérations qui la composent). Les durées des opérations sur les différentes machines font parties des données du problème [14].

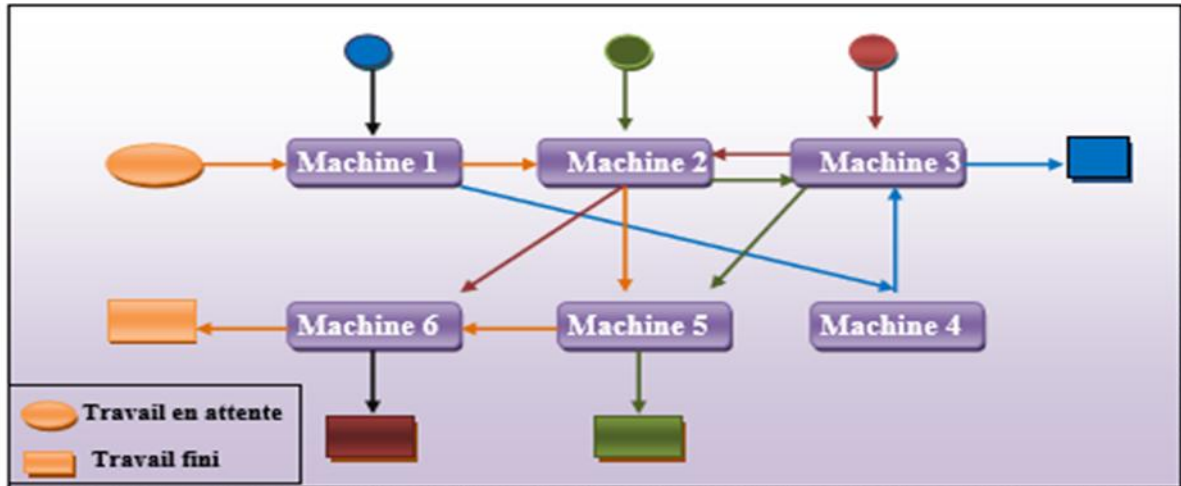


Figure 2.6 Ateliers à cheminement multiple (Job-hop) [21].

- **Open shop ($\alpha = O_m$)** : Dans un open shop avec m machines, chaque tâche a besoin d'être traitée exactement une fois sur chacune des machines (comme dans un flow shop). Sauf que l'ordre de passage des opérations constituant la tâche n'est pas fixé a priori (cheminement libre). Contrairement aux autres modèles, l'open shop n'est pas courant dans les ateliers et il n'est pas beaucoup étudié dans la littérature [14].
- **Champ β** : Ce champ spécifie les caractéristiques des tâches et les contraintes d'ordonnancement. Il peut contenir plusieurs paramètres (9 au total), de β_1 , β_2 jusqu'à β_9 [14] :
 - **Préemptions (pmtn)** : Si la préemption est permise, pmtn est inclus dans le champ β . Sinon, il n'est pas inclus.
 - **Splitting ou morcellement (spl)** : Les tâches peuvent être découpées en morceaux et exécutées en parallèle sur plusieurs machines.
 - **Sans attente ou No-Wait (nwt)** : les tâches ne sont pas autorisées à attendre entre deux machines successives.
 - **Contraintes de précédence (prec)** : prec est spécifié dans le champ β quand certaines tâches doivent être achevées avant que le traitement d'autres tâches soit commencé.
 - **Date de disponibilité ou Release dates (rj)** : Si ce symbole n'est pas présent dans le champ β alors le traitement des tâches peut commencer à tout moment.
 - **Restrictions sur le nombre de tâches (nbr)** : Si ce symbole est présent dans le champ β

alors le nombre de tâches est restreint ; par exemple $nbr = 5$ veut dire qu'il y a au plus 5 tâches à traiter.

- **Restrictions sur le nombre d'opérations des tâches (n_j)** : Ce sous champ est seulement applicable au job shop. Par exemple $n_j = 4$ veut dire que chaque tâche est limitée à au plus quatre opérations. Si ce symbole n'est pas présent, alors le nombre d'opérations n'est pas restreint.
 - **Restrictions sur les durées des tâches (p_j)** : Si ce symbole est présent, alors le temps de traitement de chaque tâche est restreint à une valeur p spécifiée.
 - **Les délais (d_j)** : Si ce symbole est présent, alors il est souhaitable que chaque tâche J_j soit terminée avant son délai fixé a priori.
- **Champ γ** : Le champ γ représente la fonction objectif à optimiser [14].

3 Caractéristiques générales des ordonnancements

3.1. Ordonnements statique et dynamique

Si l'ensemble des informations nécessaires à la résolution d'un ordonnancement est fixé au départ (ensembles des tâches, des contraintes, des ressources, etc.), on est alors devant un problème d'ordonnancement statique. Dans le cas contraire si une ou plusieurs informations nécessaires à la résolution d'un ordonnancement n'est pas fixé au départ on est devant un problème d'ordonnancement dynamique [9].

3.2. Ordonnements admissibles

Un ordonnancement est dit admissible s'il respecte toutes les contraintes du problème (dates de début, dates de fin, précédence, contraintes de ressources, etc.) [9].

3.3. Ordonnements actifs et semi-actifs

Dans un ordonnancement actif, aucune tâche ne peut commencer au plus tôt et qui entraîne l'ordre relatif entre au moins deux tâches. Dans l'ordonnancement semi actif, on ne peut pas avancer une tâche sans modifier la séquence sur la ressource [9].

3.4. Ordonnement sans retard

Dans un ordonnancement sans retard, on doit exécuter la tâche qui est en attente à condition que la ressource soit disponible [9].

3.5. Ordonnement préemptif et non préemptif

Selon les problèmes, les tâches peuvent être exécutées sans interruption, c'est-à-dire si on commence l'exécution d'une tâche elle n'est pas interrompu jusqu'à son achèvement. On parle alors d'un ordonnancement préemptif. Par contre, si les tâches sont exécutées par

morceaux, l'ordonnancement est appelé non préemptif [9].

4 Ordonnancement par l'approche proposée

On considère que l'atelier de type m machines identiques en parallèles.

Dans notre problème, la préemption n'est pas autorisée, c'est-à-dire que chaque tâche j s'exécute durant p_j unités de temps consécutives sur la même machine. Après les études des modèles mathématiques, les données du problème et le modèle qui s'expriment de la manière suivante [15] :

- n : le nombre de tâches
- m : le nombre de machines
- j : l'indice de la tâche, où $j=1, \dots, n$
- k : l'indice de la machine, où $k=1, \dots, m$
- r : la position de la tâche dans une machine, où $r=1, \dots, n_k$
- r_{jk} : date de début
- d_{jk} : date de fin de la tâche j
- s_{ij} : temps de préparation de la tâche j effectuée immédiatement après la tâche i sur une machine
- p_{jk} : temps opératoire de la tâche j
- C_{jk} : date de fin d'exécution de la tâche j
- T_{jk} : retard réel de la tâche j
- C_{max} : makespan
- n_k : nombre de tâches affectées à la machine k

$$\text{Minimiser } (C_{max} \sum T) \quad (1)$$

Sous contraintes :

$$\sum_{j=1}^n X_{jkr} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (2)$$

$$\sum_{k=1}^m \sum_{r=1}^{n_k} X_{jkr} \quad j=1,2,\dots,n \quad (3)$$

$$P_{[kr]} = \sum_{r=1}^{n_k} X_{jkr} P_{jk} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (4)$$

$$S_{[kr]} = \sum_{i=1}^n \sum_{j=1}^n X_{jkr} Y_{ij} S_{ij} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (5)$$

$$r_{[kr]} = \sum_{j=1}^n X_{jkr} r_{jk} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (6)$$

$$d_{[kr]} = \sum_{j=1}^n X_{jkr} d_{jk} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (7)$$

$$C_{[kr]} = \max(C_{[kr-1]} + s_{[kr]}, r_{[kr]}) + P_{[kr]} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (8)$$

$$T_{[kr]} = \max(C_{[kr]} - d_{[kr]}, 0) \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (9)$$

$$C_{\max} = \max_{k=1}^m \max_{r=1}^{n_k} C_{[kr]} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (10)$$

$$\sum T_{jk} = \sum_{k=1}^m \sum_{r=1}^{n_k} T_{[kr]} \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad (11)$$

$$X_{jkr} = 0 \text{ or } 1 \quad k=1,2,\dots,m \quad r=1,2,\dots,n_k \quad j=1,2,\dots,n \quad (12)$$

$$Y_{ij} = 0 \text{ or } 1 \quad i=1,2,\dots,n \quad j=1,2,\dots,n \quad (13)$$

La fonction (1) exprime directement les objectifs de notre problème, i.e., la minimisation du makespan et la minimisation de la somme des retards.

Les contraintes (2) et (3) sont des contraintes d'unicité des tâches sur la machine k et à la position r. Elles assurent qu'il y a seulement une tâche sur la machine k et à la position r, et que chaque tâche est affectée seulement une fois sur ces machines.

La contrainte (4) calcule la durée d'opération de la tâche qui est en position r sur la machine k.

La contrainte (5) définit le temps de préparation de la tâche qui est en position r sur la machine k.

Les contraintes (6) - (9) concernent respectivement la date de début, la date de fin, la date de fin d'exécution et le retard réel de la tâche qui est en position r sur la machine k.

Les contraintes (10) et (11) représentent le calcul du makespan et de la somme des retards.

La variable binaire X_{jkr} est égale à 1, si la tâche j est ordonnée en position r sur la machine k et 0 sinon. La variable binaire Y_{ij} contrôle la position relative de deux tâches i et j. Si la tâche i précède immédiatement la tâche j, alors Y_{ij} est égale à 1, sinon elle est égale à 0. Par contre, si j est la première tâche (j=1), alors Y_{ij} est égale à 1, car aucune tâche ne précède j.

5. Les méthodes de résolution du problème d'ordonnancement

Delors qu'il y a un ensemble de tâches et un ensemble de ressources, il est nécessaire de programmer les tâches et d'affecter les ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), en respectant un ensemble de contraintes. La principale difficulté à laquelle est confronté un décideur, en présence d'un problème d'optimisation est celui du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable. Les différentes méthodes de résolution

développées peuvent être classées en deux catégories : les méthodes exactes et les méthodes approchées [12].

5.1 Méthodes exactes

Une méthode exacte permet de trouver une solution optimale à un problème donné. Ces méthodes sont généralement utilisées pour résoudre des problèmes de petite taille. Dans ce cas, le nombre de combinaisons possibles est suffisamment faible pour pouvoir explorer l'espace de solutions en un temps raisonnable. On distingue trois sous-classes de méthodes exactes [16] : la procédure de séparation et d'évaluation (Branch and Bound), la programmation dynamique et la programmation linéaire.

5.2 Méthodes approchées

Est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles. D'un autre côté les algorithmes d'optimisation tels que les algorithmes de recuit simulé, les algorithmes tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires. Les méthodes approchées englobent deux classes : les heuristiques et les méta-heuristiques [16].

5.2.1 Les heuristiques

Les heuristiques sont des méthodes empiriques basées sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de ces méthodes est d'intégrer des stratégies de décision pour construire une solution proche de l'optimum, tout en essayant de l'obtenir en un temps de calcul raisonnable [6]. On distingue :

- **FIFO** (First In First Out) : la première tâche qui vient est la première tâche ordonnancée,
- **SPT** (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu,
- **LPT** (Longest Processing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu,
- **EDD** (Earliest Due Date) : la tâche ayant le date due la plus petite est la plus prioritaire,
- **SRPT** (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs,

- **ST** (Slack Time) : à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative.

5.2.2 Les Méta-heuristiques

Les méta-heuristiques sont des stratégies de recherche itératives, destinées à l'exploitation de l'espace de solutions par l'utilisation de différentes techniques. Ces méthodes n'essayent pas d'examiner toutes les solutions possibles, mais de trouver dans un temps raisonnable une solution satisfaisante par investigation intelligente de l'espace. L'ensemble des méta-heuristiques proposées dans la littérature sont partagées en deux catégories : des méta-heuristiques à base de solution unique, des méta-heuristiques à base de population de solution [16].

- **Les Méta-heuristiques à base de solution unique**

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

De nombreuses méthodes à base de solution unique ont été proposées dans la littérature, le recuit simulé, la recherche tabou. [16].

- **Les Méta-heuristiques à base de population de solutions**

Les méta-heuristiques à base de population de solutions débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure (l'optimum global) qui représentera la solution du problème traité. L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité.

Une grande variété de méta-heuristiques basées sur une population de solutions a été proposée dans la littérature, algorithmes génétiques, et les algorithmes à base d'intelligence par essais : l'algorithme d'optimisation par essaim de particules, l'algorithme de colonies de fourmis [16].

Conclusion

Dans ce chapitre nous avons résumé les différents notions liés à l'ordonnancement on a mentionné ses différentes composantes et Caractéristiques, Le deuxième point exposé dans ce chapitre sont les modèles mathématiques de problème m-machines identiques en parallèles. Enfin on a présenté les méthodes de résolution des problèmes d'ordonnancement.

Dans le chapitre suivant nous allons apprendre les algorithmes génétiques et l'optimisation multi-objectif.

CHAPITRE 3

ALGORITHMES GENETIQUES ET

OPTIMISATION MULTI-OBJECTIF

Introduction

Le présent chapitre est dédié à la description détaillée de l'algorithme d'optimisation. En particulier, nous allons présenter Les principes de base de l'optimisation multi-objectif, ainsi que les principaux algorithmes génétiques multi-objectifs (AGMOs) et une description plus détaillée des trois modèles implémenté ; le Nondominated Sorting Genetic Algorithm (NSGA II), le Strength Pareto Evolutionary Algorithm II (SPEA II) et Pareto Archived Evolution Strategy (PAES), afin de les comparer.

1. Algorithmes génétiques mono-objectif

1.1 Principes de base des algorithmes génétiques

Les Algorithmes Génétiques (AG) ont connu un succès grandissant dans les domaines de la recherche opérationnelle ces dernières années. Ils sont utilisés dans la résolution des problèmes d'optimisation complexes [15].

A travers une transposition artificielle des concepts de base de la génétique et des lois de survie énoncés par Darwin, les AGs tentent de simuler le processus d'évolution des espèces dans leur milieu naturel. Ces AGs furent développés par J.Holland [17], puis leurs fondements furent exposés par Goldberg [15].

La génétique représente un individu identifié par un ensemble de données (chromosomes). La naissance d'un individu enfant, est obtenue à partir d'un mélange aléatoire de chromosomes de deux individus, lui conférant une empreinte génétique héritée des parents [15].

L'évolution normale d'une espèce, d'une génération à l'autre, est alors une simple recombinaison obtenue par croisement aléatoire des chromosomes d'individus appariés. La mutation génétique est caractérisée dans le code génétique de l'enfant par l'apparition de nouveaux gènes inexistants chez les parents. Les changements dans la morphologie des espèces sont expliqués par l'apparition de " mutants" ce qui est un phénomène exceptionnel et ceci pour être mieux adapter au milieu naturel [15].

Un algorithme génétique idem que le phénomène génétique. Une population de taille N , constituée de N individus donne N solutions pour un problème d'optimisation. La force de chaque individu est déterminée par une procédure d'évaluation. Viennent ensuite une phase de sélection (en sélectionnant les meilleurs individus qui ont plus de chance de demeurer dans la nouvelle population), une phase de recombinaison (opérateurs artificiels de croisement) et enfin une phase de mutation. Ces trois phases permettent de générer une nouvelle population d'individus, qui ont de bonnes chances d'être plus forts que ceux de la génération

précédente[15].

La qualité des individus de la population augmente de génération en génération et après un certain nombre d'itérations. Ces individus constituent une population forte et représentent des solutions quasi - optimales à tout problème posé [15].

Les concepts de bases d'un AG sont alors comme suit [15] :

- **Individu:** Une structure fondamentale permettant le codage d'une solution candidate du problème.
- **Population:** Un ensemble d'individus d'une même génération.
- **Fitness:** Un indicateur qui calcule la valeur de la fonction de cout pour chaque individu.
- **Sélection:** le choix des éléments les plus adaptés pour la formation de la nouvelle génération.
- **Croisement et mutation:** phase de reproduction, dans laquelle une nouvelle population est construite à partir des individus sélectionnés, via des opérateurs de croisement et de mutation.

Les phases suivantes représentent le fonctionnement d'un A.G [15] :

1. **Initialisation** : une population initiale de taille N chromosomes est tirée aléatoirement.
 2. **Evaluation** : chaque chromosome est décodé puis évalué.
 3. **Reproduction:** création d'une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée.
 4. **Opérateurs génétiques:** croisement et mutation de certains chromosomes au sein de la nouvelle population.
 5. **Retour** à la phase 2 tant que la condition d'arrêt du problème n'est pas satisfaite.
- Afin d'obtenir des solutions de qualité, les opérations AG sont exécutées plusieurs fois.
La description de ces opérations est présentée dans les sous-sections suivantes.

1.2 Codage et population initiale :

Les algorithmes génétiques sont appliqués sur une population d'individus, chacun de ces derniers est codé par un chromosome ou génotype. Donc, une population est présentée par un ensemble de chromosomes. Le processus de codage d'individus consiste à trouver une structure de données pour les individus. Il existe principalement trois types de codage [18][3]:

- **Le codage binaire**

Les premiers algorithmes génétiques ont utilisé le codage binaire. Un chromosome est

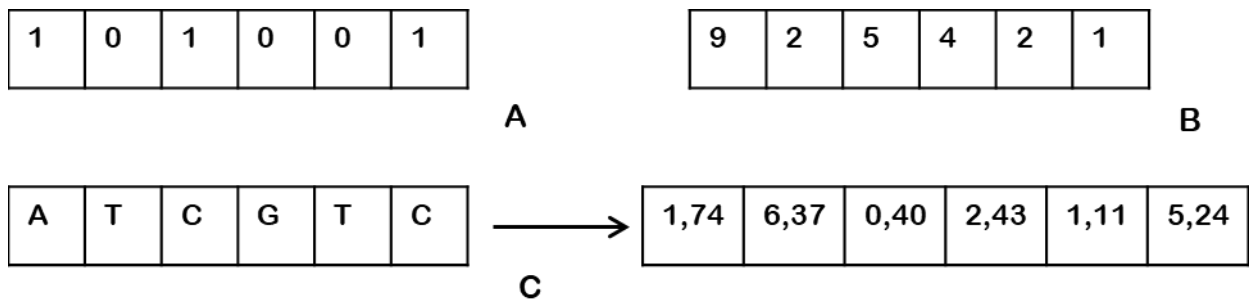
représenté par une chaîne binaire (chaîne de bits) précisant l'information nécessaire à la description d'un individu [6].

- **Le codage par valeurs entières**

Chaque gène est codé par une valeur entière. Donc, un chromosome est représenté sous forme d'une chaîne d'entiers. Ce codage est bien adapté à l'optimisation des problèmes industriels réels [6].

- **Le codage par valeur**

Dans ce type de codage, chaque gène est codé par une valeur qui appartient à un ensemble fini ou infini. Ces valeurs sont liées au problème à résoudre [6].



(A) : codage binaire,

(B) : codage par permutation de valeurs entières,

(C) : codage par valeur.

Figure 3.1 : Exemples de codage par valeurs [6]

1.3 La fonction d'évaluation

La fitness (fonction d'évaluation) évaluera l'efficacité des individus et permettra la sélection des candidats aptes à la reproduction. Trois opérateurs primaires feront évoluer la population [15] :

- La sélection.
- Le croisement.
- La mutation.

Employés dans cet ordre, ils créeront une nouvelle population d'individus.

1.4 Opérateur de sélection

Cet opérateur permet la survie, la reproduction ou la mort aux individus d'une population et sélectionne les parents reproducteurs parmi tous les individus. Les individus sélectionnés répondent au mieux au problème étudié et ayant une fitness au-dessus de la moyenne, le reste des individus sera pénalisé par cette sélection [15].

La sélection par roulette biaisée (roulette wheel selection) de Goldberg est la plus utilisée des méthodes de reproduction connues. A partir de cette méthode, chaque chromosome sera copié dans une nouvelle population proportionnellement à sa fitness. La probabilité avec laquelle un individu chi de fitness $f(chi)$, sera sélectionné dans une population de taille N est déterminée généralement de la façon suivante [15] :

$$p(chi) = \frac{f(chi)}{\sum_{j=1}^N f(chi)}$$

Le tournoi (tournament selection) est l'une des méthodes les plus connues. Il choisit entre deux individus sélectionnés au hasard l'individu le plus important et le reproduit dans une nouvelle population. La même opération sera répétée jusqu'à avoir une population complète [15].

Aussi importante soit elle la phase de sélection ne produit pas de nouveaux individus car ce rôle appartient aux opérateurs de croisement et de mutation [15].

1.5 Opérateur de croisement

La création de nouveaux individus est faite par l'opérateur de croisement, selon un processus fort simple. Un échange d'information se fait par le biais d'une combinaison entre les chromosomes après avoir été sélectionnés.

D'abord, un couple d'individus est tiré de la nouvelle population elle-même issue de la reproduction. Ensuite un ou plusieurs sites de croisement sont choisis au hasard. (chiffre entre 1 et $l-1$ ou l est la longueur de la chaîne binaire).

Enfin, le croisement se fait selon une probabilité p_c (entre 0.5 et 0.9). On garde, sans changement, les sous-chaînes contenant les bits compris entre 1 et p_c , et on échange les sous-chaînes contenant les bits compris entre $p_c + 1$ et l [15].

Parmi les méthodes de croisement les plus utilisées on peut souligner deux opérateurs: le croisement à un point et le croisement multipoints

- **Croisement à un point** : Pour chaque couple de chromosomes, on choisit aléatoirement un point de croisement et à partir duquel, on échange les ensembles d'allèles entre les deux parents comme indiqué dans la figure suivante [15] :

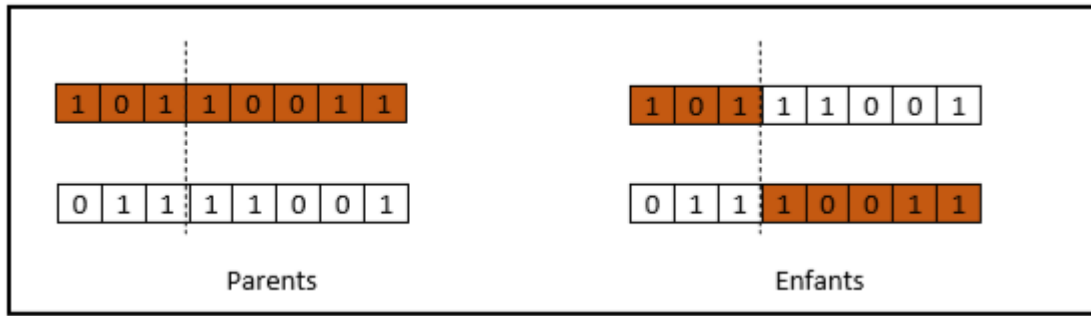


Figure 3.2 : Croisement à un site [20].

- **Croisement multipoints** : Dans ce cas, on sélectionne plusieurs points de croisement et on échange les différentes parties d'allèles entourées par ces points entre les parents [15].

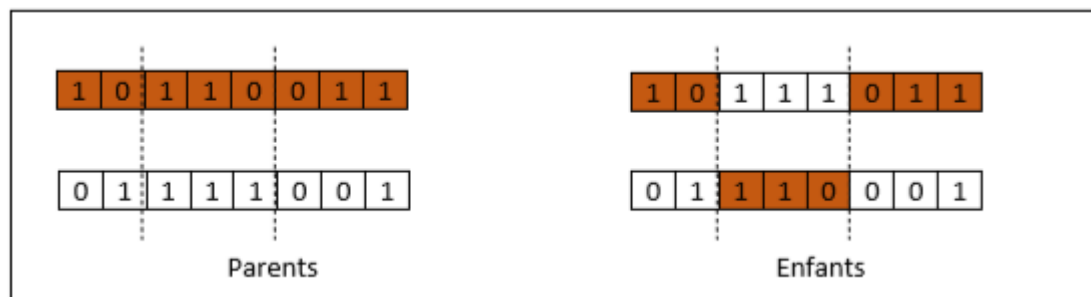


Figure 3.3 montre un croisement à deux points [20].

1.6 Opérateur de mutation

Une mutation est faite sur les individus de la population née du processus de croisement. Les chromosomes des fils biologiques, de cette mutation, sont distincts des chromosomes des parents biologiques. Le rôle de l'opérateur est de modifier la valeur d'un gène dans un chromosome en fonction du facteur de mutation. Cet opérateur applique le principe de variation de la théorie de Darwin et permet, dans le même contexte, d'éviter une convergence prématurée de l'algorithme vers un maximum local [15].



Figure 3.4 Mutation d'un chromosome [20] .

Avec ces trois opérateurs d'évolution, nous pouvons appliquer les algorithmes génétiques.

2. L'optimisation multi-objectif

L'optimisation multi-objectif consiste à trouver un vecteur de décisions qui satisfait les contraintes et optimise le vecteur objectif dont les éléments représentent les fonctions objectif, ces dernières forment une description mathématique du critère de performance, ces fonctions sont généralement en conflit. Le terme « optimiser » veut dire trouver une solution de telle façon que toutes les fonctions objectives renvoient des valeurs acceptables [20].

Un problème de minimisation est décrit comme ci-dessous [19] :

$$\text{minimise} = (f_1(x), \dots, f_n(x))$$

$$\text{Avec} \quad x = (x_1, \dots, x_m) \in X$$

$$y = (y_1, \dots, y_n) \in Y$$

Où x est appelé vecteur de décision constitué de m variables de décision, dans un AGs, il représente un individu (solution potentielle). Et y vecteur objectif de n objectifs. X représente l'espace de décision, et Y l'espace des objectifs.

L'optimisation multi objectif traite plusieurs fonctions objectifs en même temps, pour cela, elle utilise la notion de compromis optimaux. Les solutions sont départagées en se basant sur la notion de dominance au sens de Pareto.

2.1. Dominance au sens de Pareto

Dans un problème de minimisation, un vecteur de décision $A \in X$ domine un autre vecteur de décision $B \in X$ si et seulement si [15] :

$$f_i(A) \leq f_i(B) \text{ , } \forall i \in \{1, \dots, n\}$$

$$\exists j \in \{1, \dots, n\}, f_j(A) < f_j(B)$$

1- La solution A n'est pas pire que B dans tous les objectifs.

2- A est meilleure que B dans au moins un seul objectif.

2.2. Pareto optimum:

Le mathématicien italien Vilfredo Pareto au XIX^{ème} siècle, a formulé un autre concept du problème multi objectif appelé plu tard optimum de Pareto. Selon lui, il y a un équilibre entre les critères du problème multi-objectif qui empêche d'améliorer d'un d'eux sans détériorer au moins un des autres critères. Un point x est appelé Pareto-optimal lorsque aucun autre point de l'espace de recherche X ne le domine, ces points sont également appelés solutions *non inférieures* ou *non dominées* [15].

On appelle front de Pareto la représentation des solutions non dominées dans l'espace d'objectif. La figure suivante illustre l'espace objectif contenant les images des solutions non-dominées, et les solutions dominées dans l'espace objectif, ainsi que le front de Pareto pour un problème.

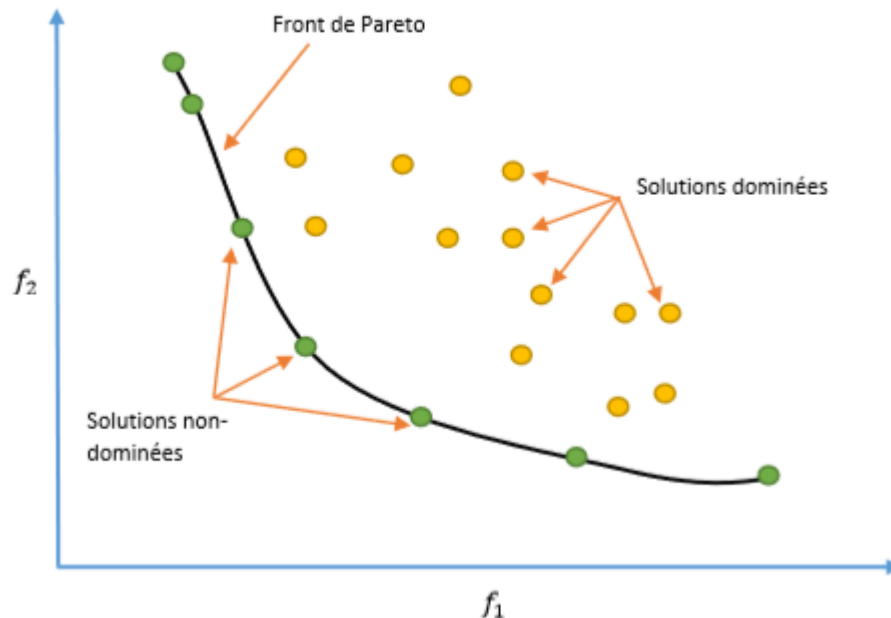


Figure 3.5 montre le front de pareto [20].

Dans la partie suivante, nous allons faire une description en l'état de l'art des méthodes d'optimisation par algorithmes génétiques et après on citera quelques exemples de quelques algorithmes génétiques.

3. Optimisation multi-objectif par algorithmes génétiques

Les algorithmes génétiques sont une méthode efficace dans la résolution des problèmes d'optimisation multi-objectif, ils sont répartis en trois classes [22] :

- Ceux qui transforment le problème multi-objectif en un problème mono-objectif en utilisant une fonction objectif équivalente Les AGs de cette classe sont appelés : **méthodes agrégées**.
- **Les méthodes non-agrégées non-Pareto** utilisent des sous-populations pour optimiser chaque objectif indépendamment.
- **Les méthodes Pareto** consistent à déterminer un ensemble de solutions optimales, elles favorisent les solutions non-dominées dans leurs processus de sélection en utilisant la notion de dominance au sens de Pareto.

3.1. Méthodes agrégées :

Par la combinaison des composants f_i du vecteur objectif du problème en une seule fonction scalaire f , on transforme le problème multi objectif en un problème mono-objectif. Dans la pratique différentes façon de construire la fonction f existent, dont la plus connu et la plus utilisée est l'addition de tous les objectifs, en attribuant à chacun d'eux un poids, représente l'importance de l'objectif aux yeux du décideur [15].

L'avantage de cette méthode réside dans sa simplicité et sa grande efficacité si on se limite à la recherche d'une solution optimale relative au vecteur des poids associés aux différents critères. Par contre l'inconvénient de cette méthode réside dans sa difficulté de fixer les valeurs des poids appropriés [15].

3.2. Méthodes non-agrégées non-Pareto

Dans ce type d'approche, la population est utilisée pour diversifier la recherche, mais le principe de Pareto n'est pas directement incorporé dans le processus de sélection. Le modèle le plus représentatif de cette méthode est le Vector Evaluated Genetic Algorithm (VEGA)[20].

Vector evaluated genetic algorithm (VEGA)

En 1985 Schaffer propose une extension d'un algorithme génétique simple pour la résolution d'un problème multi-objectif. Cette méthode est appelée Vector Evaluated Genetic Algorithm. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection.

L'idée est simple. Si nous avons K objectifs et une population de N individus, une sélection de N/K individus est effectuée pour chaque objectif. Ainsi K sous-populations vont être créées, chacune d'entre elles contenant les N/K meilleurs individus pour un objectif particulier. Les K sous-populations sont ensuite mélangées afin d'obtenir une nouvelle population de taille N . Le processus se termine par l'application des opérateurs génétiques de modification (croisement et mutation) [23].

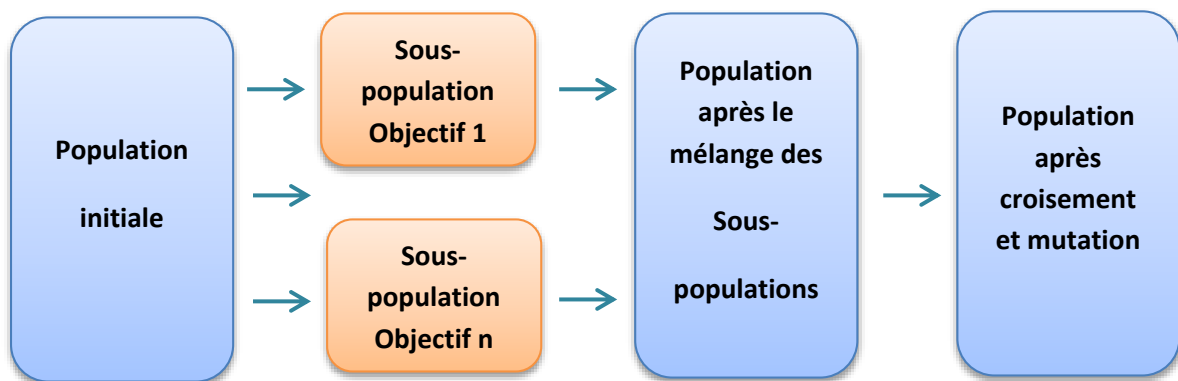


Figure 3.6 Vector Evaluated Genetic Algorithm (VEGA) [20].

3. Méthodes Pareto

Cette catégorie de méthodes introduit et prend en charge le concept d'optimalité de Pareto dans leurs mécanismes de sélection, on va citer quelques-unes :

3.1. Non-dominated Sorting Genetic Algorithm II (NSGA II)

Comme son nom l'indique, le Non-dominated Sorting Genetic Algorithm II trie (rapidement) la population à base de non-dominance [20], voici l'algorithme de cette méthode :

<i>NSGA (N,g)</i>	
<i>P0=initialisation-de-la-population(N)</i>	
<i>Evaluation des objectifs</i>	
<i>Tri-non-dominance(P0)</i>	
<i>Générer une population d'enfants Q0</i>	
<i>Sélection par tournoi binaire</i>	
<i>mutation et recombinaison</i>	
<i>Pour t=1 jusqu'à g</i>	
<i>Rt=Pt ∪ Qt</i>	Combiner les 2 populations
<i>{ F1,F2,..Fk }=tri-non-dominance(Rt)</i>	Trier Rt
<i>Pt+1= ∅</i>	Initialiser Pt+1
<i>i=1</i>	Initialiser le compteur
<i>Tant que N- Pt + 1 ≥ Fi </i>	Tant que Pt+1 est incomplète
<i>Pt+1=Pt+1 ∪ Fi</i>	inclure les fronts Fi dans Pt+1
<i>i=i+1</i>	incrémenter i de 1
<i>Fin tant que</i>	
<i>Si N- Pt + 1 < Fi & N - Pt + 1 ≠ 0</i>	Si dernier front Fi est trop large
<i>assignement-distance-crowding(Fi)</i>	attribuer les distances
<i>tirer(Fi,<n)</i>	opérateur de crowding
<i>Pt+1=Pt+1 ∪ Fi [1:N- Pt + 1]</i>	inclure une partie de Fi ∪ Pt+1
<i>Sinon</i>	si Pt+1 est complète
<i>Qt+1=créer-nouv-pop(Pt+1)</i>	créer une nouvelle population
<i>t+1</i>	Prochaine génération

Algorithme 3.1 L'algorithme de NSGA II [20]

Explication : D'abord, une population de parents (P_0) de taille (N) est générée, elle est tirée à base de non dominance, chaque solution se fait attribuer une fitness factice égale à son niveau de non-dominance (veut dire non-dominée donc meilleure fitness). Après l'application des opérations génétiques sur (P_0), une population enfant Q_0 de taille N est créée. Après, cette première procédure, une boucle commence. Tant que le critère d'arrêt n'est pas vérifié, les populations, parents et enfants sont mélangées pour produire une population ($R_t = P_t \cup Q_t$), puis tirées à base de non dominance pour identifier plusieurs fronts F_1, F_2 , etc.[20] Les meilleurs individus vont se retrouver dans le ou les premiers fronts. Une nouvelle population parent (P_{t+1}) est formée en ajoutant les fronts au complet (premier front F_1 , second front F_2 , etc.) tant que ceux-ci ne dépassent pas N . Si le nombre d'individus présents dans (P_{t+1}) est inférieur à (N), une procédure de crowding est appliquée sur le premier front suivant, (F_i), non inclus dans (P_{t+1}) pour insérer les ($N - |P_t + 1|$) meilleurs individus de (F_i) qui manquent dans la population (P_{t+1}). Les individus de ce front sont utilisés pour calculer la distance de crowding entre deux solutions voisines. Une fois que les individus de la

population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1}) est créée par sélection, croisement et mutation [15].

L'organigramme ci-dessous illustre le fonctionnement du NSGA II :

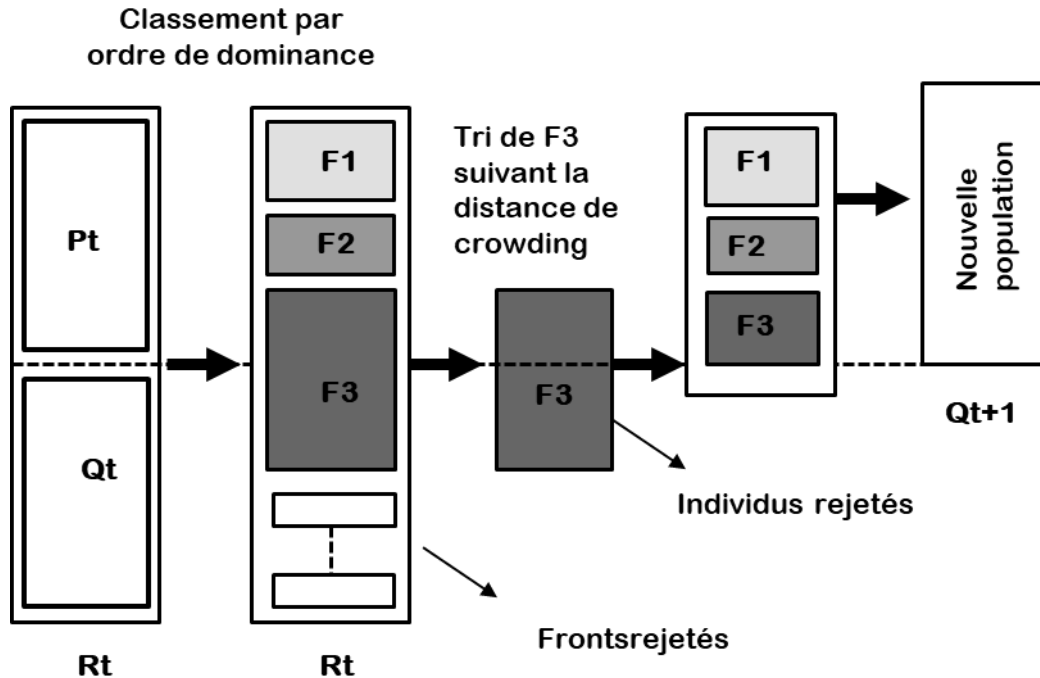


Figure 3.7 le fonctionnement du NSGA II (exemple 3 fronts admis) [20].

Calcul de la distance de crowding (distance de surpeuplement)

Une procédure de crowding, basée sur un calcul de distance (distance de crowding) qui ne nécessite aucun paramétrage. La distance de crowding d'une solution particulière i se calcule en fonction du périmètre de l'hypercube ayant comme sommets les points les plus proches de i sur chaque objectif. Sur la figure 3.8, est représenté l'hypercube en deux dimensions associé au point i . Le calcul de la distance de crowding nécessite, avant tout, le tri des solutions selon chaque objectif, dans un ordre ascendant. Ensuite, pour chaque objectif, les individus possédant des valeurs limites se voient associés une distance infinie. Pour les autres solutions intermédiaires, on calcule une distance de crowding égale à la différence normalisée des valeurs de fonctions objectifs de deux solutions adjacentes. Ce calcul est réalisé pour chaque objectif. La distance de crowding d'une solution est obtenue en sommant les distances correspondantes à chaque objectif [23].

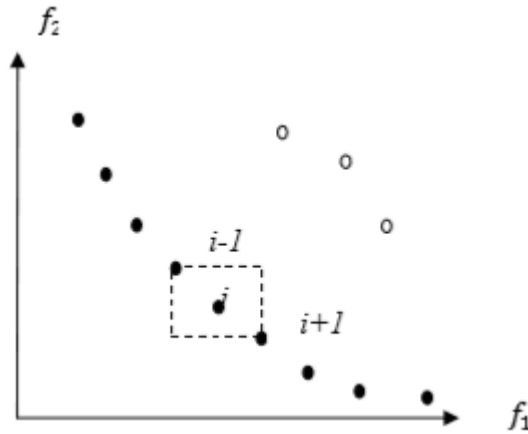


Figure 3.8 Distance de Crowding (les points noirs sont des Solutions appartenant au même front [23]).

Entre deux solutions de fronts différents, on préfère la solution avec le plus petit front. Pour deux solutions qui appartiennent au même front, on préfère l'individu possédant la plus grande valeur de distance de crowding [15].

3.2. Strength Pareto Evolutionary Algorithm II (SPEA II)

Le SPEA II a été conçu de telle sorte à corriger et à mettre à jours certaines techniques par rapport à la première version afin d'améliorer ses performances [20], voici l'algorithme de cette méthode :

<i>SPEA – II(N, g)</i>	
$P_0 = \text{générer} - \text{population} - \text{initiale}(N)$	
$\bar{P}_0 = \emptyset$	Initialiser l'archive
$t = 0$	Initialiser le compteur de générations
<i>Pour $t = 1$ jusqu'à g</i>	
<i>Attribuer – fitness($P_t \cup \bar{P}_t$)</i>	Attribuer les fitness archive + population
<i>Sélection – environnementale($\bar{P}_t$)</i>	Actualiser l'archive
<i>$S = \text{Sélection} - \text{Parentale}(\bar{P}_{t+1}, N)$</i>	Sélectionner les parents (tournoi)
<i>$P_{t+1} = \text{Opérateurs} - \text{génétiques}(S)$</i>	Croisement, mutation pour donner P_{t+1}
<i>$t = t + 1$</i>	Prochaine génération

Algorithme 3.2 L'algorithme de SPEAII [20]

Explication : D'abord, une population P_0 est aléatoirement générée, l'archive $\bar{P}_0$ est initialisé, la première étape consiste à attribuer des valeurs de fitness à tous les individus de l'archive et de la population ($P_t \cup \bar{P}_t$). Puis, à l'aide de la sélection environnementale l'archive est actualisé afin de contenir les meilleures solutions possibles, puis une sélection avec tournoi est effectuée sur l'archive $\bar{P}_t$ [20].

Enfin, les opérateurs génétiques (croisement, mutation) sont appliqués sur la population résultante donnant une nouvelle population P_{t+1} , ces opérations sont répétées tant que le critère d'arrêt n'est pas satisfait [20].

Le diagramme suivant illustre le fonctionnement de l'algorithme SPEAII :

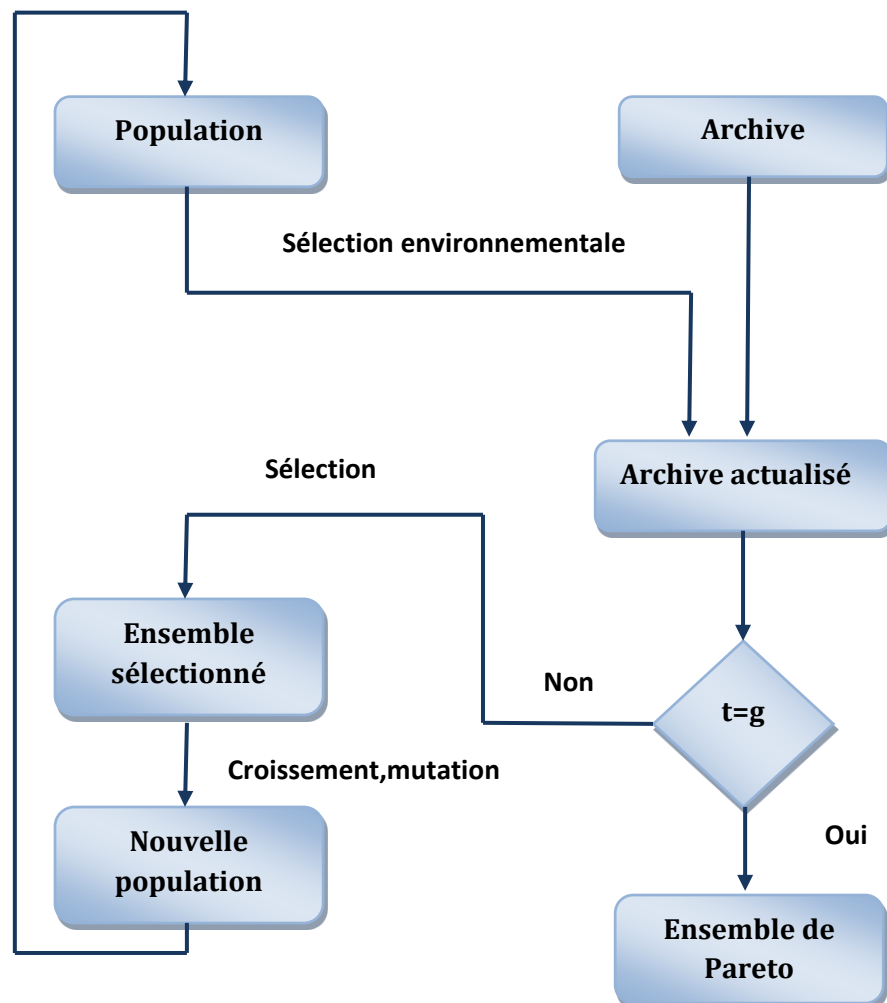


Figure 3.8 Fonctionnement du SPEAII [20]

- **La sélection environnementale** : En l'addition à la population principale, un archive contenant les solutions non-dominées. Un membre d'archive est supprimé seulement si une solution qui le domine vient d'être repérée dans la population où bien au cas de surcharge de l'archive [20].

3.3. Pareto Archived Evolution Strategy (PAES):

Cette méthode a été développée initialement comme une méthode de recherche locale. Les premiers travaux de Knowles et Corne ont montré que cette méthode simple objective fournissait des résultats supérieurs aux méthodes de recherche basées sur une population. Par conséquent, les auteurs ont adapté cette méthode aux problèmes multi-objectifs [23]. Voici l'algorithme de cette méthode :

Répéter

Générer une solution initiale aléatoire c et l'ajouter au archive

Muter c pour produire m et évaluer m

***si** c domine m jeter m*

sinon

***si** m domine c **Alors** remplace c par m et ajoute m à l'archive*

sinon

***si** m est dominé par un membre de l'archive: jeter m*

***sinon** appliquer le test ($c, m, \text{archive}$) pour déterminer quelle solution devient la nouvelle solution actuelle et s'il faut ajouter m à l'archive*

fin si

fin si

fin si

***jusqu'à** le critère d'arrêt est vérifié*

Algorithme 3.3 L'algorithme de PAES [25] .

Explication : D'abord, un parent génère par mutation une solution candidate. La solution candidate est comparée à la solution mère. Si la solution candidate domine le parent, la solution candidate est acceptée en tant que parent suivant et l'itération continue. Si le parent domine la solution candidat, celle-ci est rejetée et la nouvelle solution mutée (une nouvelle solution candidate) est générée. Si la solution candidate et le parent ne se dominant pas, un ensemble de comparaison d'individus non dominées précédemment trouvées est considérée [25].

Pour maintenir la diversité de la population au long du front de Pareto, un archive de solutions non dominées est considérée. Une nouvelle solution candidate générée est comparée à l'archive pour vérifier si elle domine un membre de l'archive. Si tel est le cas, la solution candidate entre dans l'archive et est acceptée en tant que nouveau parent. Les solutions dominées sont éliminées de l'archive. Si la solution candidate ne domine aucun membre de l'archive, la solution parent et la solution candidate sont vérifiées pour leur proximité avec la solution de l'archive [25]. Si la solution candidate réside dans la région la moins peuplée de l'espace des paramètres parmi les membres de l'archive, il est accepté en tant que parent et une copie est ajoutée à l'archive

Le diagramme suivant illustre le fonctionnement de l'algorithme PAES :

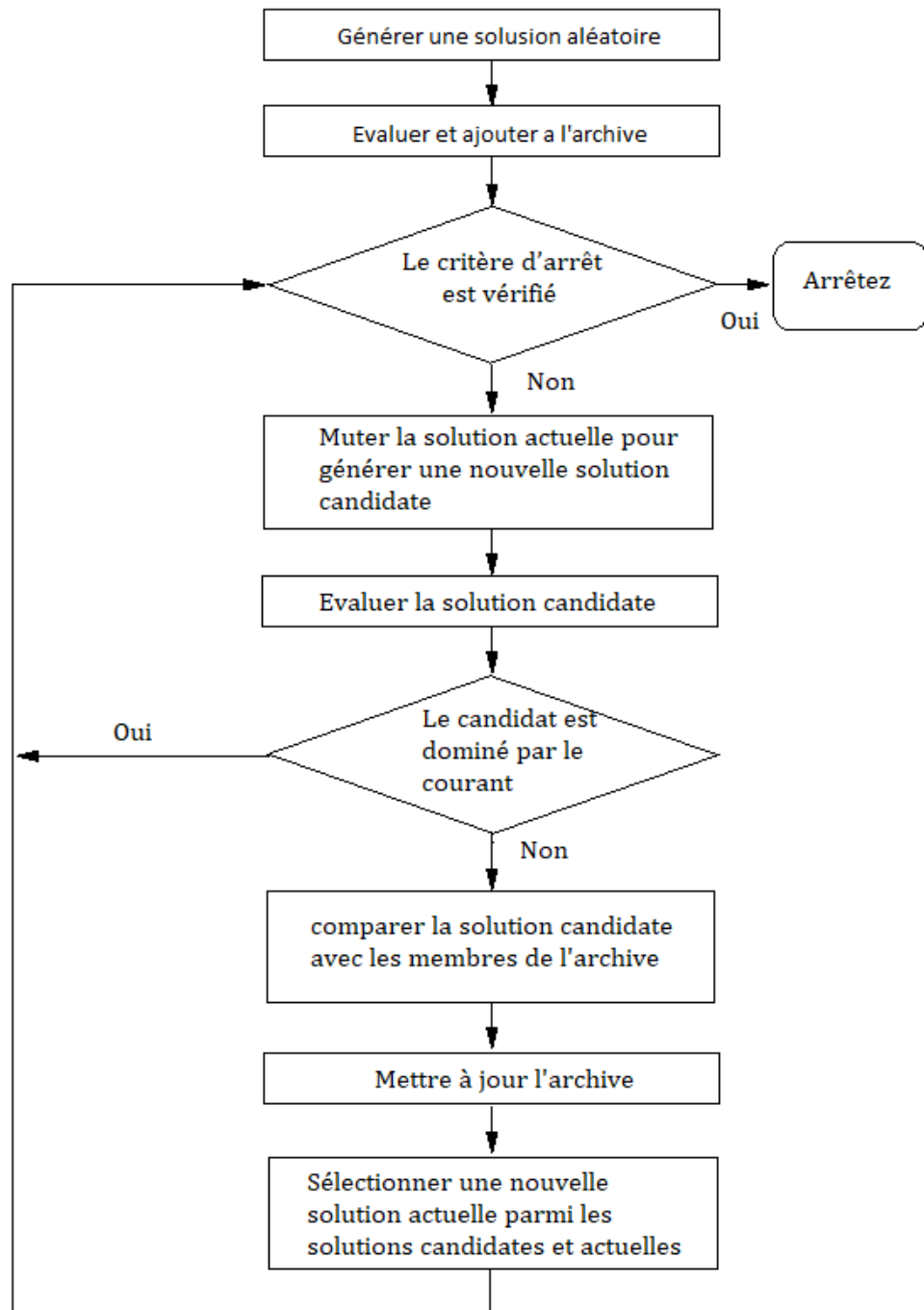


Figure 3.9 Fonctionnement de PAES [27].

Conclusion

Ce troisième chapitre a été consacré à l'optimisation multi-objectif et aux algorithmes génétiques multi-objectifs (AGMOs). Nous avons sélectionné trois modèles de références : le NSGA II, le SPEA II et le PAES. On distingue ceux qui sont basés sur le principe de Pareto, ceux qui travaillent sur une population, et ceux qui n'opèrent qu'avec un seul individu à la fois.

Dans le chapitre suivant nous allons apprendre l'étude comparative entre ces trois modèles afin de résoudre le problème m-machines identiques en parallèles.

CHAPITRE 4
ETUDE EXPERIMENTALE
ET COMPARATIVE

Introduction

Ce chapitre présente une étude comparative entre les trois algorithmes génétiques étudiés dans le chapitre précédent, et ce, dans un problème M-machines identiques en parallèles en vue de minimiser la durée total d'exécution et la somme des retards. La comparaison entre les modèles étudiées se fera selon certains critères que nous allons présenter dans ce chapitre.

1. Outils et environnement de développement

Avant de commencer l'implémentation de notre travail, nous allons tout d'abord spécifier les outils utilisés qui nous ont semblé être un bon choix vu les avantages qu'ils offrent.

Langage JAVA

Notre choix du langage java a été guidé par les avantages qu'offre la programmation orientée objet. Java utilise des processus qui augmentent les performances des entrées/sorties, facilitent l'internationalisation. Il examine le programme au fil de l'exécution et libère automatiquement la mémoire. Cette fonctionnalité diminue les risques de panne du programme et évite la possibilité de mal utiliser la mémoire.

Netbeans

Notre travail est écrit en Netbeans version 8.2 sous Windows 10. Le choix de la programmation sous Windows a été pris à cause de l'interface graphique qu'offre cet environnement. Le choix de Netbeans est fondamental car c'est un logiciel permettant principalement le développement en java. Il fournit un environnement standard de développement pour créer des interfaces très puissantes.

2. Quelque exemple sur l'implémentation

Exemple 01 : Dans un problème de type M-machines identiques en parallèles compose de 4 machines et 7 jobs tels que les données sont :

Tâches	1	2	3	4	5	6	7
P (temps de traitement)	4	3	4	5	2	6	3
r (temps de disponibilité)	0	2	2	4	6	8	10
d (temps de livraison)	8	8	12	12	14	18	20

Table 4.1 La table des tâches Exemple 1.

Avec notre implémentation nous avons la solution suivante :

- l'ordre de l'ordonnancement obtenu après l'application de NSGA II :

Tâches	1	2	3	4	5	6	7
n (indice de machine)	1	4	1	2	1	3	4
K (indice de position sur la machine)	1	1	2	1	3	1	2

Table 4.2 La table de la solution avec NSGA II Exemple 1.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 4 & 4 & 2 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 0 & 2 & 6 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 10 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 12 & 14 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 20 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 4 & 8 & 10 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 13 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Alors que Pr c'est la matrice de temps opératoire de la tâche j , R c'est la matrice de date de début, D c'est la matrice date de fin la tâche j , C c'est la matrice de makespan, T c'est la matrice retard réel de la tâche j .

Diagramme de Gantt :

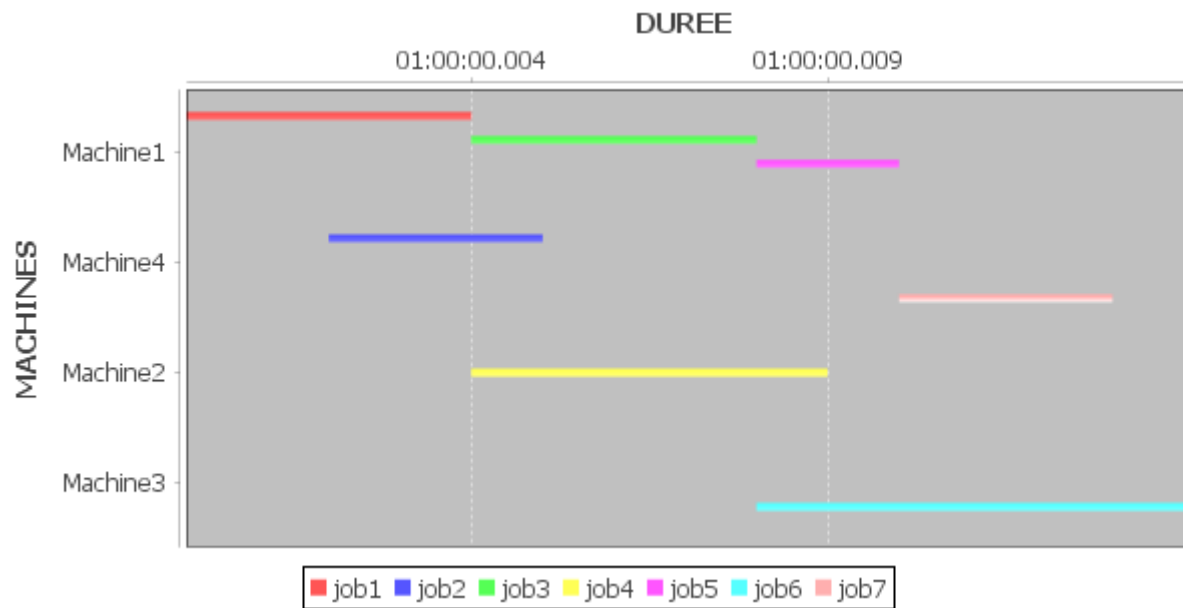


Figure 4.1 Exemple de Diagramme de Gant obtenu par NSGA II de problème 4x7.

Alors on a $C_{max}=14$, $\sum T=0$, le temps d'exécution est 1216 ms , l'espace mémoire utiliser est 48 mb.

- l'ordre de l'ordonnancement obtenu après l'application de SPEA II :

Tâches	1	2	3	4	5	6	7
n (indice de machine)	4	3	2	3	2	2	1
K (indice de position sur la machine)	1	1	1	2	2	3	1

Table 4.3 La table de la solution avec SPEA II Exemple 1.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 2 & 6 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 6 & 8 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 14 & 18 & 0 & 0 & 0 & 0 \\ 8 & 12 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 8 & 14 & 0 & 0 & 0 & 0 \\ 5 & 10 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

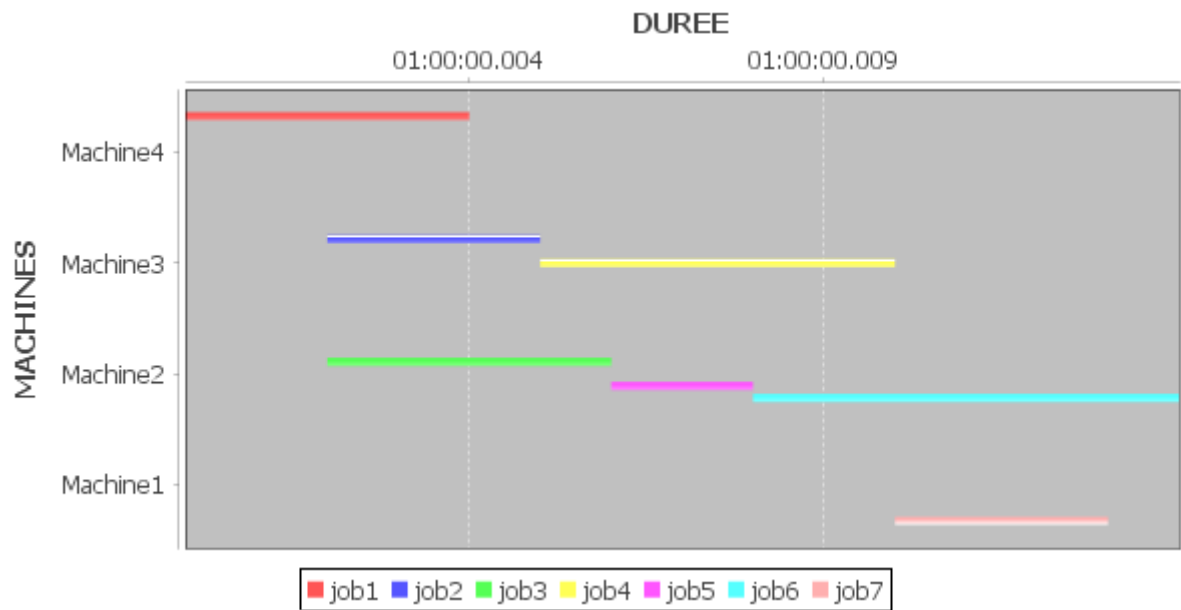


Figure 4.2 Exemple de Diagramme de Gant obtenu par SPEA II de problème 4x7.

Alors on a $C_{max}=14$, $\sum T=0$, le temps d'exécution est 21603 ms , l'espace mémoire utiliser est 201 mb.

▪ l'ordre de l'ordonnancement obtenu après l'application de PAES :

Tâches	1	2	3	4	5	6	7
n (indice de machine)	4	1	1	2	4	3	1
K (indice de position sur la machine)	1	1	2	1	2	1	3

Table 4.4 La table de la solution avec PAES Exemple 1.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 3 & 4 & 3 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 2 & 10 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 12 & 20 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 14 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 5 & 9 & 13 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

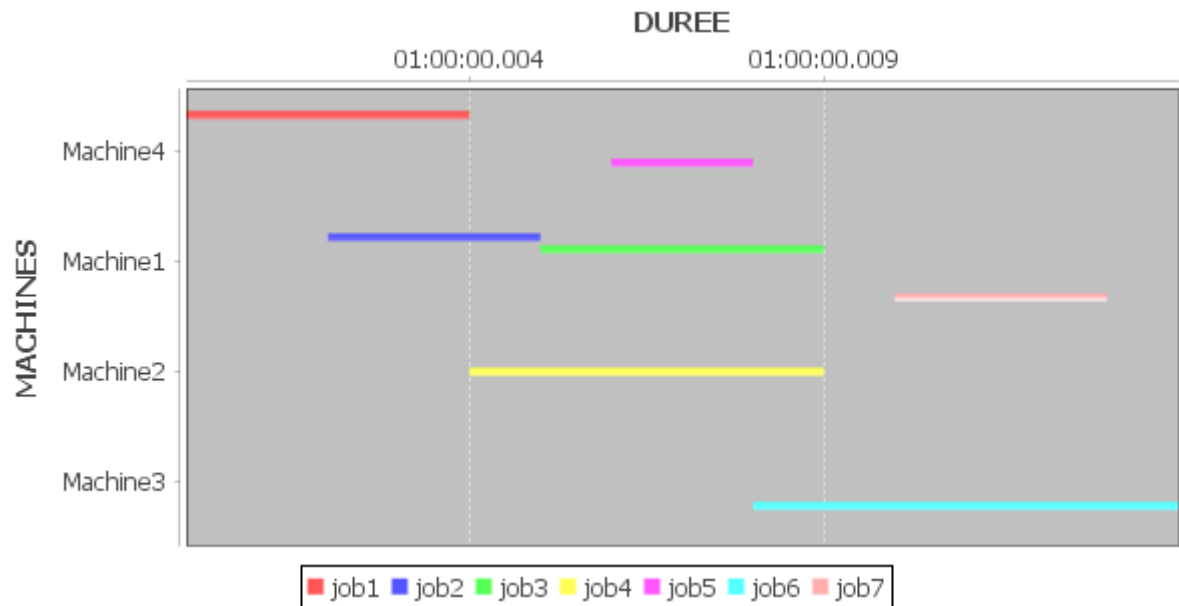


Figure 4.3 Exemple de Diagramme de Gantt obtenu par PAES de problème 4x7.

Alors on a $C_{max}=14$, $\sum T=0$, le temps d'exécution est 375 ms , l'espace mémoire utiliser est 85 mb.

Exemple 02 : Dans un problème de type M-machines identiques en parallèles compose de 2 machines et 8 jobs tels que les données sont :

Tâches	1	2	3	4	5	6	7	8
P (temps de traitement)	4	3	4	5	4	2	6	7
r (temps de disponibilité)	0	2	2	4	2	4	3	4
d (temps de livraison)	8	8	12	12	8	9	7	10

Table 4.5 La table des tâches Exemple 2.

Avec notre implémentation nous avons la solution suivante :

- l'ordre de l'ordonnancement obtenu après l'application de NSGA II :

Tâches	1	2	3	4	5	6	7	8
n (indice de machine)	1	2	1	2	1	2	1	2
K (indice de position sur la machine)	1	1	3	3	2	2	4	4

Table 4.6 La table de la solution avec NSGA II Exemple 2.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 4 & 4 & 4 & 6 & 0 & 0 & 0 & 0 \\ 3 & 2 & 5 & 7 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 0 & 2 & 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 4 & 4 & 4 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 8 & 12 & 7 & 0 & 0 & 0 & 0 \\ 8 & 9 & 12 & 10 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 4 & 8 & 12 & 18 & 0 & 0 & 0 & 0 \\ 5 & 7 & 12 & 19 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

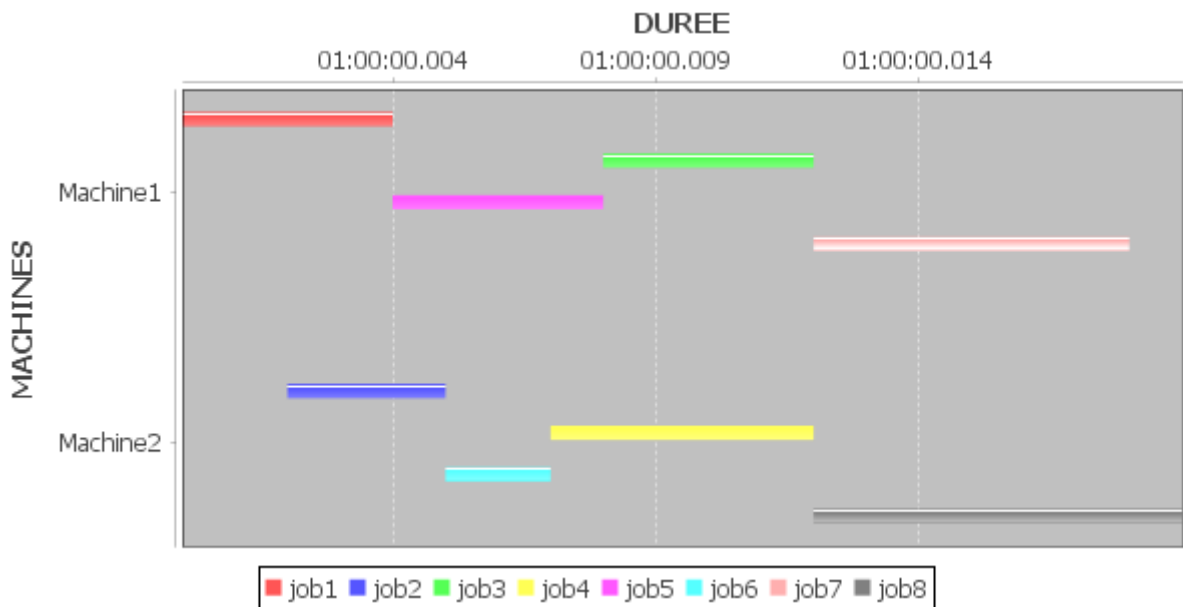


Figure 4.4 Exemple de Diagramme de Gant obtenu par NSGA II de problème 2x8.

Alors on a $C_{max}=19$, $\sum T=20$, le temps d'exécution est 1283 ms , l'espace mémoire utiliser est 61 mb.

▪ l'ordre de l'ordonnancement obtenu après l'application de SPEA II :

Tâches	1	2	3	4	5	6	7	8
n (indice de machine)	2	1	2	1	2	1	2	1
K (indice de position sur la machine)	1	1	3	3	2	2	4	4

Table 4.7 La table de la solution avec SPEA II Exemple 2.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 3 & 2 & 5 & 7 & 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 6 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 4 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 3 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 9 & 12 & 10 & 0 & 0 & 0 & 0 \\ 8 & 8 & 12 & 7 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 5 & 7 & 12 & 19 & 0 & 0 & 0 & 0 \\ 4 & 8 & 12 & 18 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

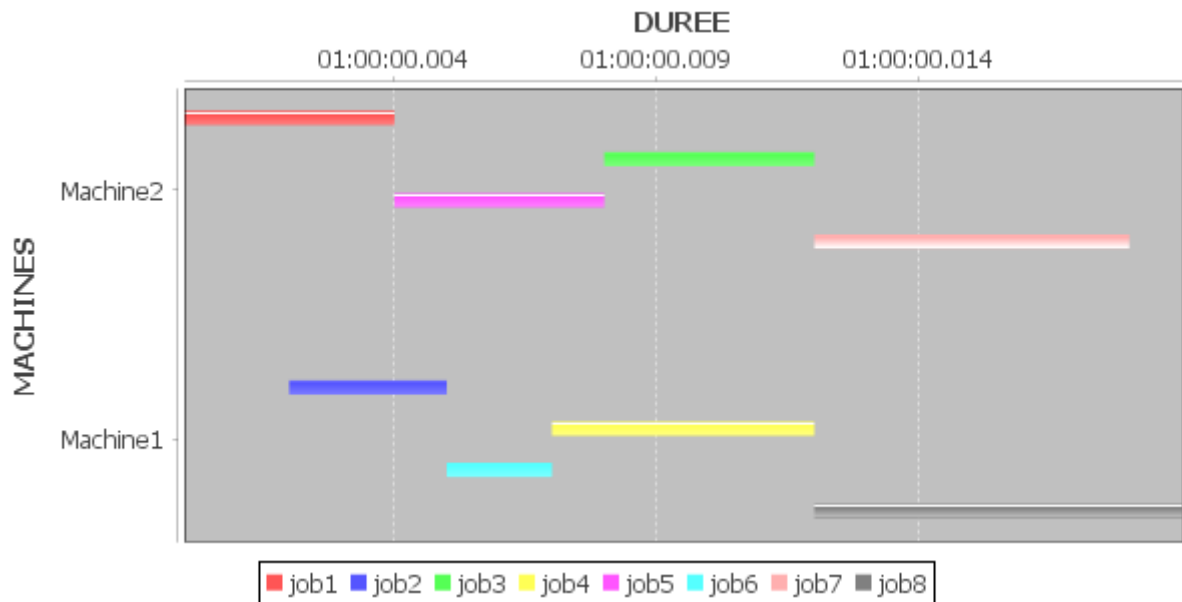


Figure 4.5 Exemple de Diagramme de Gant obtenu par SPEA II de problème 2x8.

Alors on a $C_{max}=19$, $\sum T=20$, le temps d'exécution est 14594 ms , l'espace mémoire utiliser est 133 mb.

▪ l'ordre de l'ordonnancement obtenu après l'application de PAES :

Tâches	1	2	3	4	5	6	7	8
n (indice de machine)	2	2	1	2	1	1	2	1
K (indice de position sur la machine)	1	2	3	3	1	2	4	4

Table 4.8 La table de la solution avec PAES Exemple 2.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 4 & 2 & 4 & 7 & 0 & 0 & 0 & 0 \\ 4 & 3 & 5 & 6 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 4 & 2 & 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 3 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 9 & 12 & 10 & 0 & 0 & 0 & 0 \\ 8 & 8 & 12 & 7 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 6 & 8 & 12 & 19 & 0 & 0 & 0 & 0 \\ 4 & 7 & 12 & 18 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

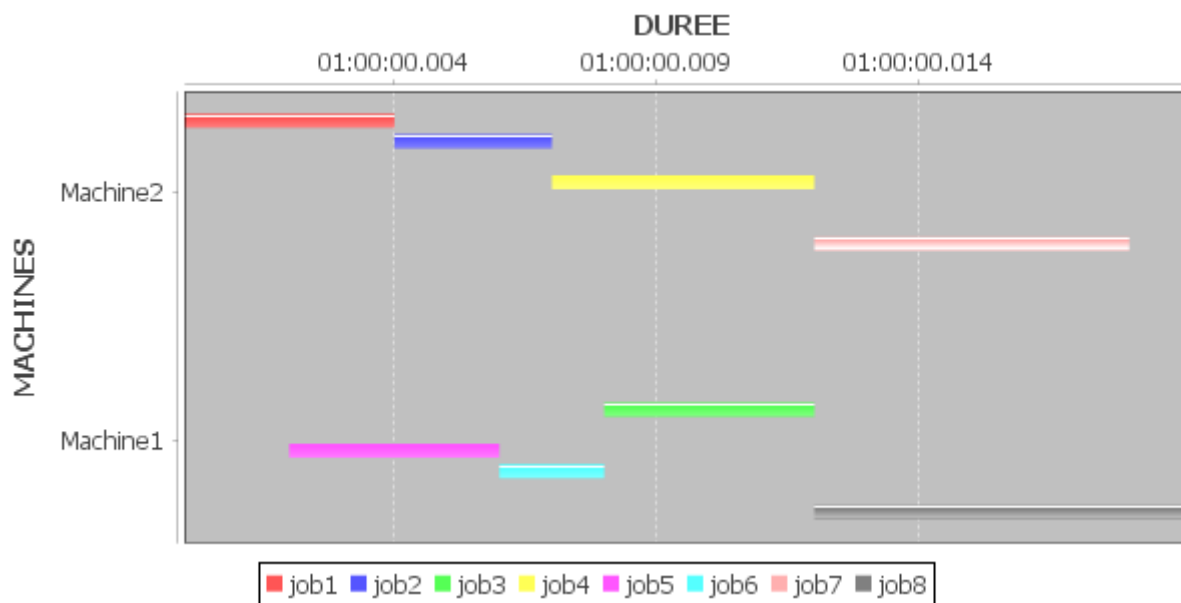


Figure 4.6 Exemple de Diagramme de Gant obtenu par PAES de problème 2x8.

Alors on a $C_{max}=19$, $\sum T=20$, le temps d'exécution est 506 ms , l'espace mémoire utiliser est 77mb.

Exemple 03 : Dans un problème de type M-machines identiques en parallèles compose de 3 machines et 9 jobs tels que les données sont :

Tâches	1	2	3	4	5	6	7	8	9
P (temps de traitement)	4	3	4	5	2	6	3	6	4
r (temps de disponibilité)	0	2	2	4	6	8	10	8	9
d (temps de livraison)	8	8	12	12	14	18	20	18	18

Table 4.9 La table des tâches Exemple 3.

Avec notre implémentation nous avons la solution suivante :

- l'ordre de l'ordonnancement obtenu après l'application de NSGA II :

Tâches	1	2	3	4	5	6	7	8	9
n (indice de machine)	2	1	2	3	1	1	3	2	3
K (indice de position sur la machine)	1	1	2	1	2	3	3	3	2

Table 4.10 La table de la solution avec NSGA II Exemple 3.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 3 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 9 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 14 & 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 12 & 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 18 & 20 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 5 & 8 & 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 13 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

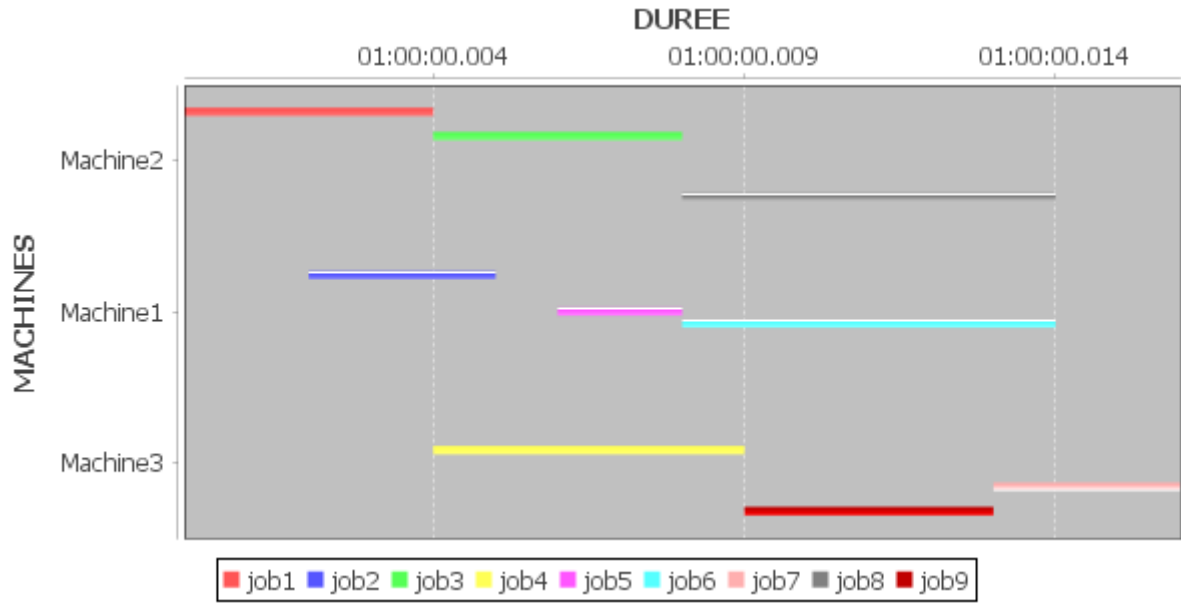


Figure 4.7 Exemple de Diagramme de Gant obtenu par NSGA II de problème 3x9.

Alors on a $C_{max}=16$, $\sum T=0$, le temps d'exécution est 1307 ms , l'espace mémoire utiliser est 45mb

▪ l'ordre de l'ordonnancement obtenu après l'application de SPEA II :

Tâches	1	2	3	4	5	6	7	8	9
n (indice de machine)	2	1	2	3	1	3	2	1	2
K (indice de position sur la machine)	1	1	2	1	2	2	4	3	3

Table 4.11 La table de la solution avec SPEA II Exemple 3.

$$Pr = \begin{pmatrix} 3 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 3 & 0 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 6 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 9 & 10 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 8 & 14 & 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 12 & 18 & 20 & 0 & 0 & 0 & 0 & 0 \\ 12 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 5 & 8 & 14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 13 & 16 & 0 & 0 & 0 & 0 & 0 \\ 9 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

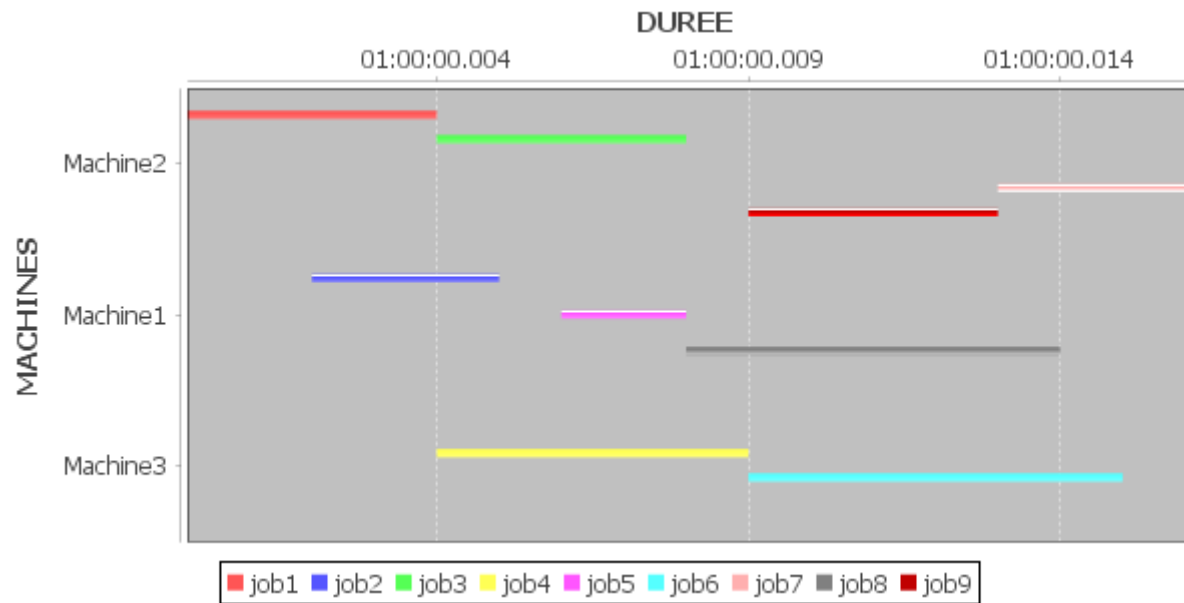


Figure 4.8 Exemple de Diagramme de Gant obtenu par SPEA II de problème 3x9.

Alors on a $C_{max}=16$, $\sum T=0$, le temps d'exécution est 12134 ms , l'espace mémoire utiliser est 154mb

▪ l'ordre de l'ordonnancement obtenu après l'application de PAES :

Tâches	1	2	3	4	5	6	7	8	9
n (indice de machine)	3	2	3	1	3	2	1	3	1
K (indice de position sur la machine)	1	1	2	1	3	2	3	4	2

Table 4.12 La table de la solution avec PAES Exemple 3.

Les matrices Pr, R, D,C,T sont:

$$Pr = \begin{pmatrix} 5 & 4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 2 & 6 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 4 & 9 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 8 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 12 & 18 & 20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 12 & 14 & 18 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 9 & 13 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 10 & 16 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagramme de Gantt :

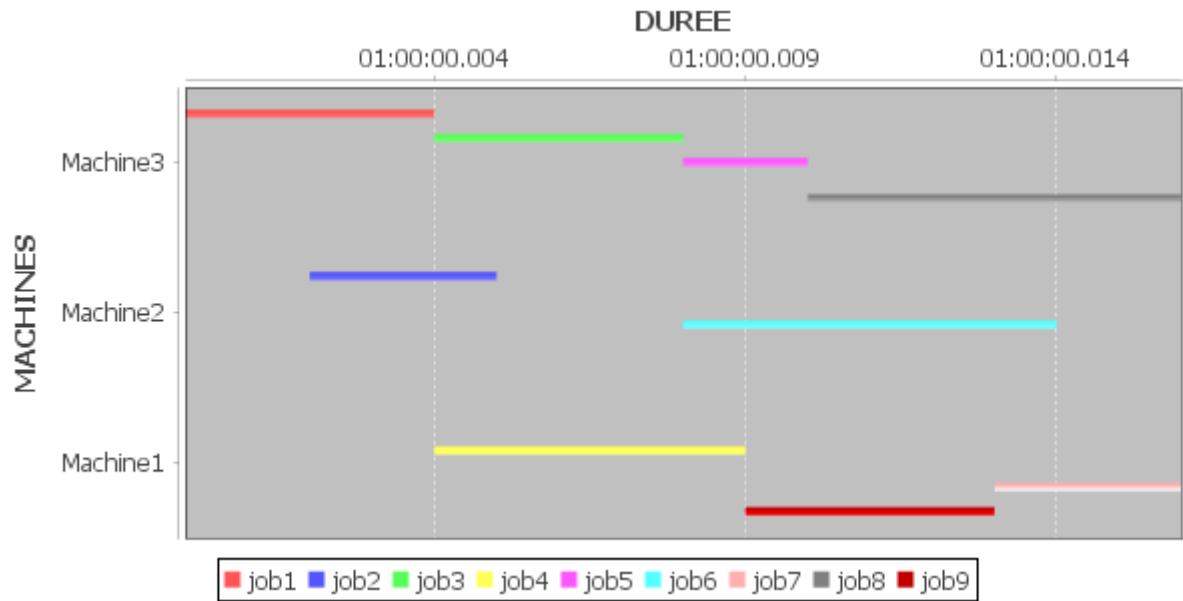
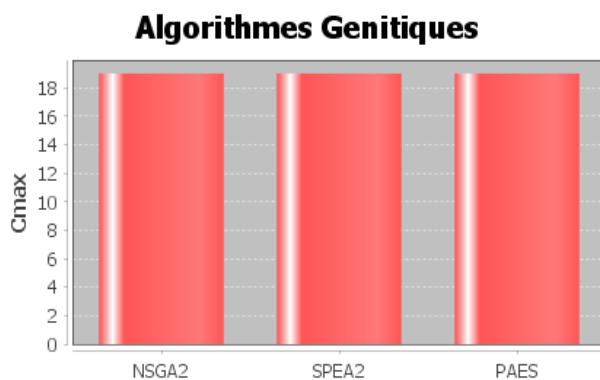


Figure 4.9 Exemple de Diagramme de Gant obtenu par PAES de problème 3x9.

Alors on a $C_{max}=16$, $\sum T=0$, le temps d'exécution est 181 ms , l'espace mémoire utiliser est 85mb.

3. Comparaison

Après, les exemples qu'on a fait, nous avons trouvé différents résultats des algorithmes génétiques que nous avons implémenté sur le problème de M-machines identiques en parallèle



En remarqué que les trois algorithmes on la même performance en terme de C_{max} [makespan, c'est le temps d'exécution nécessaire].

Figure 4.10 Diagramme obtenu par Exemple 2

Les trois algorithmes ont la même performance en terme de $\sum T$ [la somme des retards]. Maintenant on voit très bien que le nombre de machines est important, puisque si le nombre de machines augmente, le temps d'exécution tend vers zéro et vice-versa.

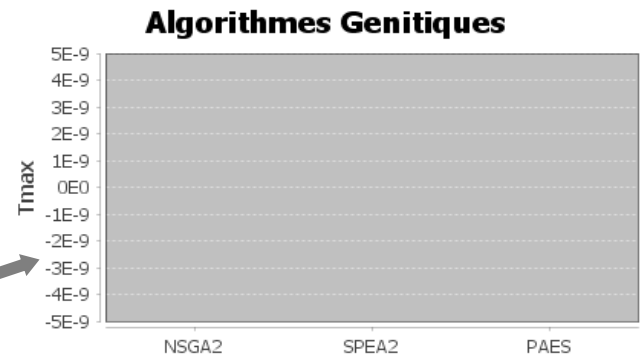


Figure 4.11 Diagramme obtenu par Exemple 4

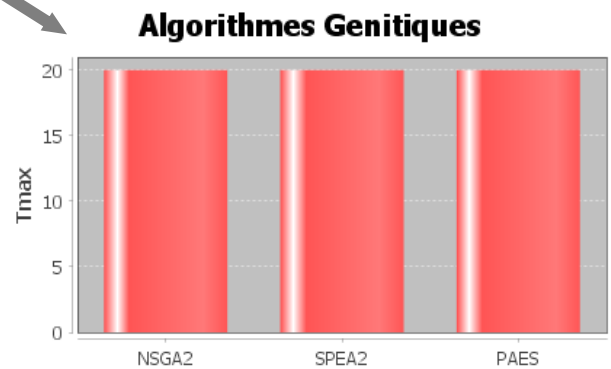


Figure 4.12 Diagramme obtenu par Exemple 2

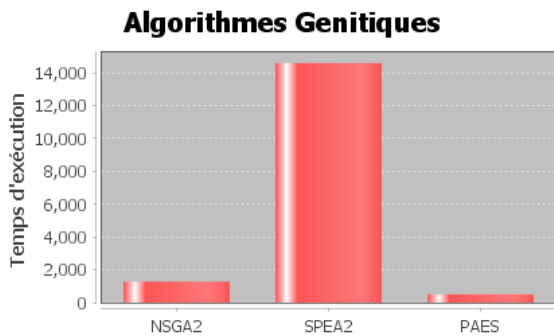


Figure 4.13 Diagramme obtenu par Exemple 4

En termes de temps d'exécution, on remarque que le PAES a la meilleure performance.

En termes d'espace mémoire, on remarque que le NSGA II a la meilleure performance.

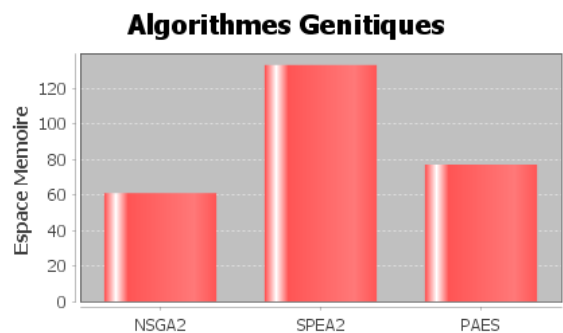


Figure 4.14 Diagramme obtenu par Exemple 4

4. Quelques interfaces de l'application

Les figures suivantes ce sont Les différentes interfaces existant dans l'application :

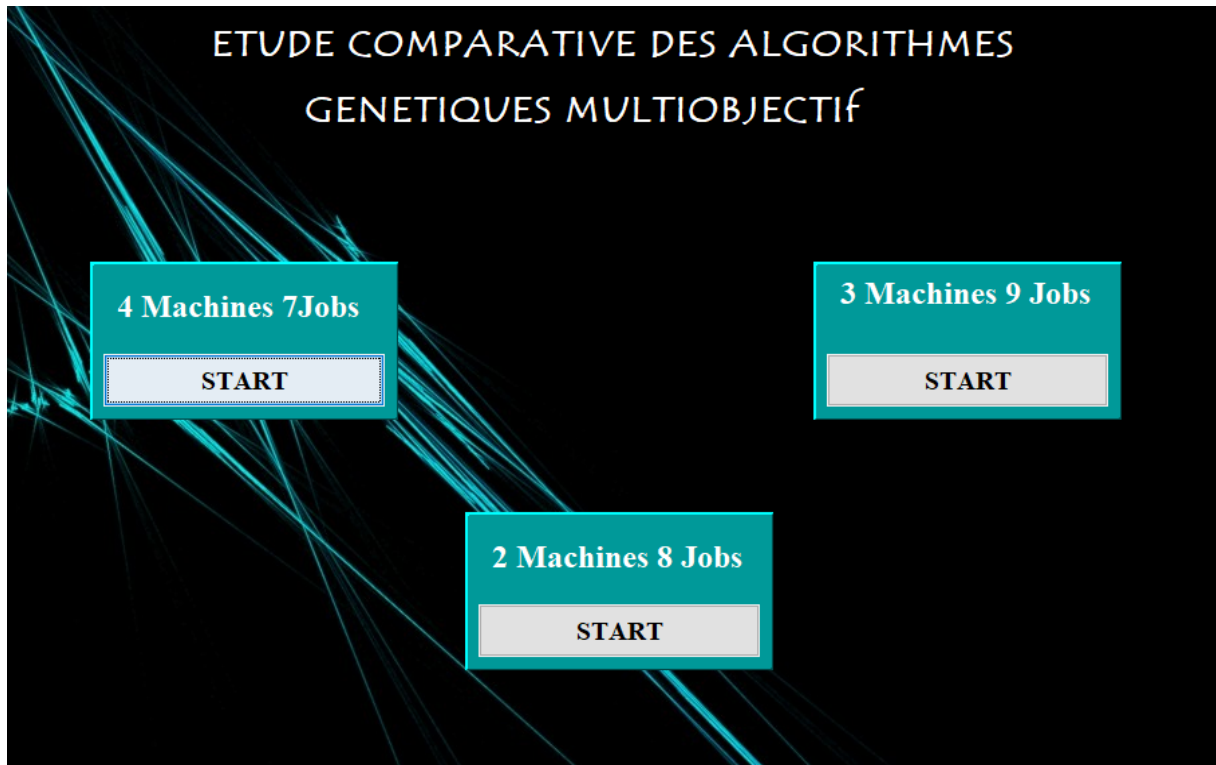


Figure 4.15 Interface principale de l'application

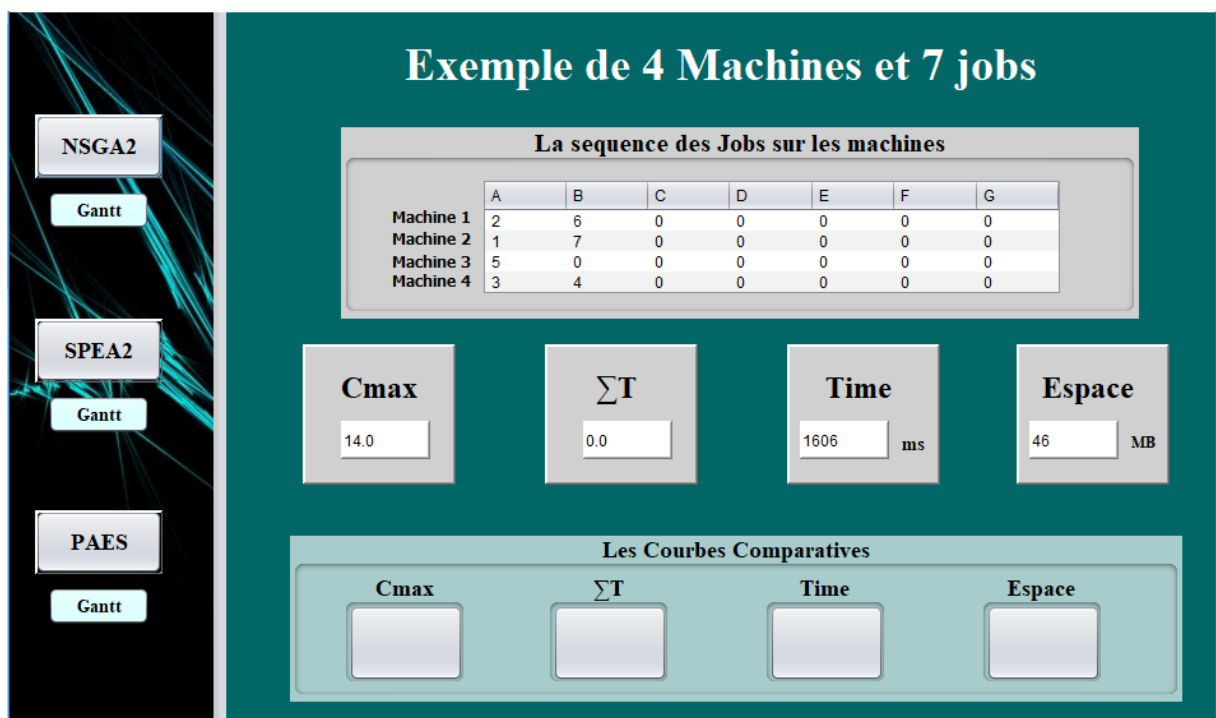


Figure 4.16 Exécution de premier exemple de NSGA II.

Conclusion

Nous avons présenté dans ce chapitre le logiciel développé, On a présenté notre étude comparative entre les différents algorithmes génétiques par rapport à quatre critères cités respectivement : makespan (C_{max}) , la somme des retards ($\sum T$) , le temps d'exécution et l'espace mémoire en fonction de la mise en œuvre que nous avons fait.

CONCLUSION GENERALE

Les problèmes de l'ordonnancement sont présents dans tous les secteurs de l'économie et constituent une fonction importante en gestion de production. Un problème d'ordonnancement consiste à allouer dans le temps des tâches à des ressources qui existent en quantité limitée, tout en satisfaisant un ensemble de contraintes.

Le travail exposé dans ce mémoire s'intéresse à l'étude comparative entre les algorithmes génétiques concernant la résolution d'un problème d'ordonnancement des ateliers de production. Plus particulièrement, les problèmes de M machines identiques en parallèles en vue de minimiser le makespan et la somme des retards.

Dans le premier chapitre, on a présenté la production et sa gestion ainsi que les problèmes liés aux systèmes de production.

Dans le deuxième chapitre, on a étudié 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 (M-Machine en parallèle, Job Shop, Flow Shop, Open Shop). Ainsi la description de l'approche d'optimisation et un certain nombre de méthodes de résolution.

Après avoir introduit les notions principales de l'étude, Dans le troisième chapitre, on a présenté les algorithmes génétiques et l'optimisation multi-objectifs. Après on a décrit d'une manière détaillée les trois algorithmes génétiques multi-objectifs implémenté (NSGAII, SPEAII et PAES).

Enfin, Le dernier chapitre entamé une étude comparative entre les AGMOs choisis en considérant un problème d'ordonnancement M-machines identiques en parallèles.

Les résultats obtenus montrent qu'il n'y a pas de grande différence entre les différents algorithmes en termes C_{max} et $\sum T$, toutefois en terme de temps d'exécution le PAES a la meilleure performance et pour l'espace mémoire c'est le NSGA II, alors Nous concluons que les trois algorithmes sont capables pour résoudre ce problème.

Comme perspectives, nous envisageons de mettre en place la même comparaison avec autre type de problème, ainsi que comparer ces algorithmes génétiques dans un problème d'optimisation à trois objectifs.

REFERENCES BIBLIOGRAPHIQUES

- [4] A .Letouzey, Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs, Doctorat, TOULOUSE, 2001.
- [20] A. Benhama et Y. Benkhalouf ,Etude comparative des algorithmes génétiques multi-objectifs, Bejaia,2014.
- [16] A . karay , Contribution à l'ordonnancement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides,Doctorat,TUNIS,2011.
- [2] C. Sassine, Intégration des politiques de maintenance dans les systèmes de production. Thèse de doctorat soutenue à l'INP de Grenoble , France, 1998.
- [22] C. A. C. Coello et G. B. Lamont, Applications of multi-objective evolutionary algorithms (Vol. 1). World Scientific ,2004.
- [26] D. Trentesaux. "Conception d'un système de pilotage distribué, supervisé et multicritère pour les systèmes automatisés de production". Thèse de Doctorat en Sciences, Institut National Polytechnique de Grenoble, 1996.
- [19] E. Zitzler, K. Deb, L.Thiele, Comparison of multi-objective evolutionary algorithms: Empirical results, Evolutionary Computation, vol. 8, PP. 173-195, 2000.
- [23] Etude des méthodes d'optimisation multicritère.
- [10] Group Gotha, Les problèmes d'ordonnancement, RAIRO. Recherche opérationnelle, tom 27,n°1,1993.
- [5] G. Javel, Organisation et Gestion de la Production, 3ème édition, DUNOD, PARIS, 2004.
- [14] J. Y.-T.Leung, Handbook of Scheduling : Algorithms, Models and Performance analysis. Chapman & Hall/CRC Computer and information sciences series,2004.
- [17] J.Holland , Adaptation in natural and artificial and systems. 2nd edition. IT Press, 1992.
- [27] J.D.knowles & D.W.Corne, The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization , 1999.
- [28] J. Vacher. "Un système adaptatif par agents avec utilisation des algorithmes génétiques multi-objectifs : Application à l'ordonnancement d'atelier de type job-shop N *M", Doctorat , Université du HAVRE, 2000.
- [3] K . Mebarek, Utilisation des stratégies Méta-heuristiques pour l'ordonnancement des ateliers de type Job Shop, Magister, BATNA, 2008.
- [18] L. R. Haupt et S. E. Haupt, practical genetic algorithms, 2nd ed. John Wiley & Sons, Inc, New Jersey, 2004.
- [6] M. Meziane, Optimisation par phases pour le problème d'ordonnancement des ateliers de type job shop totalement flexibles, Magister, ORAN, 2011.
- [12] M . BOURZIK , Problèmes d'Ordonnancement d'Atelier , 2016.
- [24] Mokeddem .D., Contrôle Flou des Processus Biotechnologiques à Base d'Algorithmes Génétiques, Thèse de doctorat, Université Ferhat Abbas de Setif, (2010).

- [25] Multiobjective Optimization Using Adaptive Pareto Archived Evolution Strategy.
- [15] M.E.BOUNIF , Optimisation à base de simulation pour le développement des systèmes décisionnels, M'SILA,2015.
- [9] N. Mouhoub, Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet, Doctorat, SETIF, 2011.
- [21] N.Taghezout , Conception et Développement d'un système multi-agent d'Aide à la Décision pour la gestion de production dynamique , Doctorat, TOULOUSE ,2011.
- [8] P. Lopez et F. Roubellat, "Ordonnancement de la production". Ed. Hermes Science, 2001.
- [11] P.Lopez , Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources, Doctorat ,à Toulouse (France). LAAS CNRS, 1991.
- [13] R.L .Graham , E.L. Lawler et J.K. Lenstra et A.HG.Rinnoy Kan ,“Optimisation and approximation in deterministic sequencing and scheduling: asurvey”, Discrete Mathematics 5, 1979.
- [29] S. AMAZOUZ , M.E. DJEDDA , Optimisation bi-objectif pour un ordonnancement intégré des tâches de production et de maintenance,Master,Boumerdès,2016
- [1] V. Giard, Gestion de la Production, 2ème édition, Economica, PARIS, 1988.
- [7] Y. Bahmani, Optimisation multicritère de l'ordonnancement des activités de la production et de la maintenance intégrées dans un atelier Job Shop, Doctorat , BATNA –II , 2017.

ملخص

هذا العمل هو دراسة مقارنة بين ثلاث خوارزميات وراثية متعددة الأهداف NSGA II و SPEA II و PAES. يتم إجراؤه باستخدام عدة معايير (makepan و ΣT ووقت التنفيذ ومساحة الذاكرة) لكل خوارزمية. استخدمنا مشكلة جدولة آلات متوازية و متطابقة من أجل تقليل وقت التنفيذ الكلي ومجموع التأخير للحصول على النتائج ومقارنتها لتحديد الخوارزميات الجينية الأكثر ملاءمة لهذه المشكلة. .

الكلمات المفتاحية: الخوارزميات الجينية ، NSGA II ، SPEA II ، PAES ، التحسين ، متعددة الأهداف.

Abstract

This work is a comparative study between three multi-objective genetic algorithms: the NSGA II, the SPEA II and PAES. It is conducted using several criteria (makespan, ΣT , execution time and memory space) of each algorithm. We used an identical M parallel scheduling problem in order to minimize the total execution time and the sum of the delays to obtain results and compare them to determine which genetic algorithms are best suited for this problem. .

Key words: Genetic algorithms, NSGA II, SPEA II, PAES, optimization, multi-objectives.

Résumé

Ce travail est une étude comparative entre trois algorithmes génétiques multi-objectifs : le NSGA II, le SPEA II et PAES . Elle est menée en utilisant plusieurs critères (makespan , ΣT , temps d'exécution et l'espace mémoire) de chaque algorithme. Nous avons utilisé un problème d'ordonnancement M machines identiques en parallèles en vue de minimiser la durée totale d'exécution et la somme des retards pour obtenir des résultats et de les comparer afin de déterminer quel sont les Algorithmes génétiques les mieux adaptés pour ce problème.

Mots clés : Algorithmes génétiques, NSGA II, SPEA II, PAES, optimisation, multi-objectifs.