الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة أبي بكر بلقايد – تلمسان –

Université Aboubakr Belkaïd– Tlemcen –

Faculté de TECHNOLOGIE

THESE

Présentée pour l'obtention du **grade** de **DOCTEUR EN SCIENCES**

En : Génie Industriel

Spécialité : Productique

Par : HOUARI Habiba

Sujet

L'intelligence artificielle pour résoudre les problèmes d'ordonnancement des systèmes flexibles de production

Devant le jury :

Dr.GHOMRI Latéfa	Maître de Conférences A	Université de Tlemcen	Présidente
Pr.SARI Zaki	Professeur	Université de Tlemcen	Directeur de thèse
Dr.HASSAM Ahmed	Maître de Conférences B	Université de Tlemcen	Co-Directeur de thèse
Pr.HAFFAF Hafid	Professeur	Université d'Oran	Examinateur 1
Dr.HACHEMI Khalid	Maître de Conférences A	Université d'Oran	Examinateur 2
Dr.KOULOUGHLI Sihem	Maître de Conférences A	Université de Tlemcen	Invité

Année universitaire 2017/2018

Remerciements

Ce présent travail a été effectué au sein de Laboratoire de Productique de Tlemcen (MELT) Université Abou Bakr Belkaid Tlemcen, Algérie.

Je remercie Dieu tout puissant de la patience et de la volonté qu'il m'a donné pour réaliser ce travail.

En préambule à cette thèse, je souhaite adresser ici tous mes remerciements aux personnes qui m'ont apporté leur aide et qui ont ainsi contribué à l'élaboration de ce travail.

Je tiens à remercier vivement mon encadreur Monsieur le Professeur Zaki SARI, de m'avoir fait l'honneur d'accepter la direction de cette thèse, il m'a accordé sa confiance en me laissant une grande liberté de pensée et d'action, tout en me faisant part de ses avis, conseils et suggestions.

C'est un agréable devoir pour ma d'exprimer ma très vive reconnaissance à Monsieur HASSAM Ahmed, Maître de Conférences classe B à l'Université de Tlemcen pour m'avoir guidé durant toute l'élaboration de ce mémoire avec le sérieux et la compétence qui le caractérise. Qu'il trouve ici le témoignage de ma très profonde gratitude.

Je suis particulièrement sensible au grand honneur que Mlle GHOMRI Latéfa, Maître de Conférences classe A à l'université de Tlemcen, ma fait en acceptant de présider le Jury de soutenance. Qu'elle trouve ici l'expression de ma profonde reconnaissance.

Je tiens également à remercier Monsieur HAFFAF Hafid, Professeur à l'Université de d'Oran Es-Senia de m'avoir fait le très grand honneur d'accepter de participer au jury de cette Thèse.

Mes remerciements s'adressent également à Monsieur HACHEMI Khalid, Maître de Conférences classe A à l'université d'Oran ; pour l'intérêt qu'il a bien voulu porter à ces travaux en acceptant de participer à notre Jury de soutenance.

Je tiens à remercier vivement Madame KOULOUGHLI Sihem, Maître de Conférences classe A, à l'université de Tlemcen, Directrice du Laboratoire de Productique de Tlemcen (MELT), d'avoir bien voulus accepter de rapporter sur ce travail. Qu'elle trouve ici, le témoignage de ma profonde reconnaissance.

Que serait la vie d'une femme sans l'homme et l'amour de sa vie ?

Je remercie ce lui qui m'a soutenu quand j'en avais réellement besoin,

La personne qui a comblé ma vie par sa gentillesse

La personne qui a été à mes côtés durant la majorité de la réalisation de ce travail,

La personne qui m'a encouragé sans relâche,

Je dédie ce travail à mon mari bien-aimé BENAHMED Mohamed

À mes enfants,

À ma mère et défunt père

Table des Matières

Table des Figures

Table des Figures

Liste des Tableaux

Introduction générale

La personnalisation des produits manufacturés, la concurrence internationale, les normes réglementaires nationales et internationales, le coût de l'énergie, la variation et la diversification de la demande pour ne citer que celles-là, sont des contraintes externes aux quelles les entreprises du secteur industriel doivent faire face. En plus de cela il faut rajouter les contraintes internes telles que : délai de livraison, le temps de production, les transports et la disponibilité des ressources. Face à ces aléas, les entreprises doivent faire évoluer leurs outils de production en leur apportant de la flexibilité.

Cette flexibilité implique des coûts importants dû aux prix des machines à commande numérique, des moyens de transport de plus en plus robotisés et automatisés et des systèmes de gestion de plus en plus sophistiqués et complexes. Ainsi l'adoption des systèmes de production flexibles nécessite des études approfondies avant leurs mises en application, tant sur le plan économique que sur le plan des nouvelles performances qu'elles pourraient apporter à l'entreprise. Pour cela les entreprises ont besoin d'outils afin de pouvoir évaluer les changements qu'elles pourraient subir de manière involontaire (variation du niveau de commande, pannes, ...) mais aussi celles qu'elles provoquent (introduction d'un nouveau produit, acquisition d'une nouvelle machine, …). Les organes de décision ont donc besoin d'outils de prédiction permettant d'observer l'impact que pourrait provoquer une de leur décision ou une de leur action sur les performances de leurs appareils de production. Ces outils de prédiction sont le plus souvent basés sur des modèles de simulation et des méthodes d'aide à la décision.

Un système de production robuste doit non seulement être capable de souplesse ou d'adaptation mais il doit également continuellement rechercher l'optimalité de ses décisions en termes de coûts, délais et qualité. Pouvoir gérer des systèmes de production à plus ou moins grande échelle est particulièrement difficile. Cela nécessite une gestion performante de l'information et de la communication. Une erreur de critique, un mauvais calcul dans la gestion des stocks, exagérer les rythmes de fabrication peuvent être fatals à l'entreprise. Heureusement,

des solutions informatiques existent pour optimiser votre activité par un système d'ordonnancement des tâches très complexe.

Les problèmes d'ordonnancement dans le secteur industriel, sont parmi les problèmes d'optimisation les plus étudiés. Améliorer le rendement des ressources et minimiser les coûts de production sont devenu les objectifs des industriels. Chercher le meilleur moyen de maximiser son profit est aujourd'hui l'un des objectifs principaux de toute entreprise. C'est dans ce contexte qu'entre nos travaux de recherche exposés dans cette thèse.

Le développement de la théorie de la complexité des algorithmes a permis de clarifier la difficulté du problème. La plupart des problèmes d'ordonnancement sont classés NP-difficiles. L'appartenance à la classe des problèmes de ce type signifie qu'on ne pourra probablement jamais résoudre ce problème d'une manière optimale. Il s'ensuit que la plupart des problèmes de taille industrielle (plusieurs milliers de tâches, ressources complexes et contraintes spécifiques) sont impossibles à résoudre de manière exacte.

Les systèmes de production, de tous genres, ont déjà intégré l'IA (l'intelligence artificielle) au sein de leur process de production. On la retrouve à présent notamment dans les machines et dans les logiciels d'ordonnancement. L'IA fait partie de ces éléments qui dessinent l'industrie de demain, La production est optimisée grâce à l'intelligence artificielle qui permet aux machines de procéder à des calculs de toutes sorte pour adapter la cadence, augmenter le taux d'occupation, épauler un opérateur sur la ligne de production, gérer les stocks de manière optimale… L'IA accélère la production, augmente la productivité, diminue les frais et les délais. Les bénéfices sont colossaux et grâce à l'utilisation de l'IA dans les logiciels on pourrait augmenter ces bénéfices de 40% d'ici 2035.

Une partie des techniques de l'Intelligence Artificielle repose sur l'utilisation des algorithmes évolutionnaires, également appelé les variantes des AG, dont le fonctionnement très simplifié et inspiré d'évolution biologique des humains. Les algorithmes évolutionnaires sont des approches particulièrement intéressantes pour résoudre des problèmes d'optimisation. Ces approches sont basées sur la manipulation d'un groupe de solutions admissibles à chacune des étapes du processus de recherche.

Dans cette thèse nous avons étudié deux types de systèmes de production flexible (job shop et système à machines parallèles identiques), nous nous sommes intéressés aux processus d'ordonnancement dans un système de production job shop avec présence d'incertitudes externes caractérisées par le non déterminisme des ordres de fabrications et internes, liées aux

pannes de machines et les incertitudes sur les durées opératoires. En raison de l'existence de machines identiques, les pièces à usiner peuvent avoir des routages alternatifs, ce qui justifie la présence de la flexibilité de routage dans ce système. Les décisions d'ordonnancement sont prises selon l'ordre d'arrivée des pièces (les ordres de fabrication), pour cette raison nous avons développé un algorithme pour un système de production flexible de machines parallèles identiques afin d'obtenir des résultats plus fiables.

Ce travail est partagé en cinq chapitres. Le premier chapitre représente des généralités sur l'ordonnancement des systèmes de production, où nous allons parler des méthodes de résolution des problèmes d'ordonnancement, dans le deuxième chapitre nous nous sommes intéressés de manière un peu plus approfondie à l'intelligence artificielle (IA), nous citerons quelques méthodes liées au domaine d'ordonnancement ainsi que les éléments nécessaires à la compréhension de ces méthodes. Ce chapitre est composé de deux parties principales, la première traite de manière générale ces techniques d'intelligence artificielle, leurs définitions, leurs domaines d'utilisation, leurs caractéristiques. La seconde partie est réservée à la présentation des algorithmes évolutionnaires et les variantes des algorithmes génétiques les plus utilisées en incluant leurs définitions, leurs origines et leurs algorithmes de base.

Nous présentons dans ces deux chapitres des informations qui existent déjà dans la littérature afin que les non spécialistes puissent comprendre le domaine. Notre travail proprement dit sera présenté dans les trois chapitres qui restent.

Une étude bibliographique et un état de l'art sur les méthodes de résolutions du problème d'ordonnancement job shop seront abordés dans le troisième chapitre, Le troisième chapitre décrit les problèmes d'optimisation combinatoire de manière générale. Il comprend les notions principales des métaheuristiques et leur classification ensuite, nous citons les différentes algorithmes évolutionnaires (AE). Puis nous parlons d'analyse du comportement de l'intensification et de la diversification pour des variantes d'algorithmes génétiques et GA. Nous présentons une comparaissent entre ces variantes. Enfin nous terminons par une conclusion.

Le quatrième chapitre, représente le cœur de notre travail, il est réservé aux méthodes de sélection de routages alternatifs. Dans ce chapitre nous commençons par donner une introduction, puis une description complète du modèle FMS étudié, ainsi nous proposons deux algorithmes pour la prise des décisions de routage de pièces, ces algorithmes peuvent répondre aux variations des demandes de fabrication même en cas de présence de pannes de machines. Finalement, l'interaction des décisions de routages de pièces basée sur les algorithmes proposés et les autres décisions prises par les différents algorithmes qui sont déjà utiliser est investiguée

en supposant que le temps opératoire est toujours déterministe, puis nous allons interpréter les résultats des simulations avec présence de pannes. Nous terminons le chapitre par une conclusion.

Le cinquième chapitre est réparti sur deux grandes partie. La première partie de ce chapitre sera réservée à la présentation d'une étude bibliographique et un état de l'art sur les méthodes de résolutions du problème d'ordonnancement à machines parallèles identiques. Au cours de cette partie, nous avons rappelé quelques généralités et définitions normatives sur ces méthodes de résolutions qui sont classées en deux types à savoir, les méthodes exactes et les méthodes approchées heuristiques ou métaheuristique.

L'algorithme que nous allons développer a été inspiré de l'optimisation d'un réseau de neurone d'Hopfield appliqué aux problèmes d'ordonnancement d'ateliers à machines parallèles. Cette nouvelle technique développée est le sujet de la deuxième partie du chapitre cinq, elle constitue une amélioration de la convergence du réseau d'Hopfield vers une solution optimale ou proche de l'optimale pour minimiser le Makespan du problème d'ordonnancement des ateliers de machines parallèles identiques. Les résultats de simulation de cette méthode ont confirmé la puissance de cette dernière. Une étude comparative avec l'algorithme utilisant LPT a prouvé que la méthode développée est plus efficace en termes de résolution grâce à l'équilibre des charges sur les différents composent du système et que les makespan trouvés convergent beaucoup aux solutions optimales avec une réduction du temps de calcul due à la diminution de l'espace de recherche du makespan.

En fin, cette thèse présente trois contributions essentielles, soient l'adaptation de deux algorithmes à base des variantes d'algorithmes génétiques pour les décisions de routages de pièces, l'analyse du comportement de l'intensification et de la diversification pour ces variantes pour l'amélioration des performances des systèmes de production flexibles et l'apport d'algorithme pour la recherche du makespan optimal pour un problème NP-difficile tel que l'ordonnancement de machines parallèles identiques.

Enfin, nous terminons notre thèse par une conclusion générale en présentant un bilan final de notre travail et en ouvrant de nouvelles perspectives de recherches.

Chapitre 1

1 Généralités sur l'ordonnancement des systèmes de production

Ce chapitre est dédié aux notions fondamentales concernant les systèmes de production qui seront traités dans cette thèse. Pour cela, nous rappelons des généralités et définitions d'ordonnancement, l'objectif est d'introduire les notions et les notations de ces problèmes, nous étudions ensuite les éléments des systèmes de production et leurs différents types. Les dernières sections de ce chapitre sont consacrées aux problèmes d'ordonnancement d'ateliers, à savoir la définition d'un problème d'ordonnancement, les objectifs de l'ordonnancement, les problèmes d'ordonnancement, évaluation et la complexité d'un problème d'ordonnancement, et les méthodes de résolution.

Sommaire

1.1 Introduction

Dans plusieurs cas de la vie réelle, les décisions à prendre sont souvent limités par des exigences spécifiques. Plus important encore, ces contraintes sont généralement problématiques dans la nature. Le processus de prise de décision devient de plus en plus compliqué avec l'augmentation du nombre de contraintes. La modélisation et le développement de méthodologies de solution pour ces scénarios, ont été le challenge pour les chercheurs dès le départ. Différents algorithmes, heuristiques et formulations ont été développés pour différentes catégories de problèmes.

L'ordonnancement est le champ de recherche qui étudie ces catégories de problèmes, c'est une activité très courante dans les milieux de l'industrie entre autres. Tous les jours, des réunions sont programmées, des délais sont fixés pour les projets, les périodes de vacances et de travail sont définies, de maintenance et de mise à niveau des opérations sont prévues, les salles d'opération sont réservées et des jeux de sport sont programmées et les arénas réservés, etc. L'ordonnancement adéquate permet à diverses activités, emplois ou tâches à être exécuter d'une manière organisée, tout en évitant les conflits de ressources. Parmi les exemples d'activités on peut citer les tâches d'une machine qui doit effectuer au cours d'une journée de travail, les opérations de fabrication dans une entreprise de semi-conducteurs, etc. L'objectif de l'ordonnancement peut-être la réduction au minimum du temps de remplir toutes les activités, la minimisation du retard des activités qui ne peuvent pas être terminé à temps, l'achèvement des activités les plus importantes sur le temps, la maximisation du nombre de clients ou patients servi, etc.

L'ordonnancement est une fonction essentielle dans la gestion de la production, il détermine ce qui va être fait, quand, où et avec quelles ressources. L'ordonnancement de la production est important dans le plan opérationnel et il est un problème difficile en fonction du nombre de calculs et de contraintes. Il cherche la combinaison optimale pour minimiser le temps de fabrication, les délais des stocks, les temps d'attente des machines et donc le temps d'attente du client. Plusieurs solutions algorithmique, heuristiques et d'adaptation ont été proposées pour divers problèmes d'ordonnancement de production. Dans de nombreux systèmes de production, les jobs de production doivent passer par plusieurs étapes, et à chaque étape, il y a plusieurs machines parallèles avec une puissance de traitement différente pour gérer les jobs. Dans ce chapitre nous présentons les problèmes d'ordonnancement déterministes et leur classification et on illustre la représentation graphique d'une solution. Nous focalisons notre intérêt sur les

problèmes à machines parallèles identiques et job shop. Ils modélisent plusieurs problèmes pratiques et théoriques.

1.2 Préliminaires

1.2.1 Généralités et définitions

L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison.

Ordonnancer c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leur date d'exécution, de sorte qu'une fonction objectif soit optimisée. La réalisation peut être, un programme informatique ou le carnet de commandes d'un atelier. Alors les tâches sont des processus informatiques ou des traitements à appliquer à certaines pièces par des machines. Les ressources sont alors des processus et de la mémoire ou des machines et du personnel. Établir un ordonnancement revient donc à coordonner l'exécution de toutes les tâches, en utilisant au mieux les ressources disponibles **[Chrétienne et al 1988].**

Les différentes données d'un problème d'ordonnancement sont les tâches, les ressources, et les contraintes. Dans un système de production, un problème d'ordonnancement se définit comme étant la localisation dans le temps de la réalisation d'un ensemble de tâches $T = \{T_1,T_2,...,T_n\}$, compte tenu des contraintes temporelles (une date de début avec une durée ou une date de fin) et de contraintes portant sur l'utilisation et la disponibilité des ressources $M = \{M_1, M_2,..., M_m\}$ requises par les tâches. Dans ce cas, une machine est considérée comme une ressource **[Boukef 2009].**

Une tâche se définit comme un ensemble d'opérations. Une opération est le passage d'une tâche *j* sur une machine *i* et elle est notée (i,j) ou O_{ij}. La durée de traitement ou temps d'exécution d'une opération est son temps de passage sur une machine donnée p_{ij} ; c'est-à-dire le temps entre le moment où l'opération débute sur une machine et le moment où elle la quitte.

Deux types de tâches sont distingués :

- Les tâches morcelables (préemptives) qui peuvent être exécutées en plusieurs fois facilitant ainsi la résolution de certains problèmes,
- Les tâches non morcelables (indivisibles) qui sont exécutées en une seule fois et ne peuvent pas être interrompues avant qu'elles soient complètement terminées.

La Figure 1.1 illustre les principales contraintes temporelles relatives à une tâche. Une tâche c'est un job i localiser dans le temps par les notations suivantes :

P_i: La durée opératoire de la tâche i (processing time).

r_i: La date de début au plutôt ou date de disponibilité de la tâche i (release time).

d_i: La date de fin au plus tard ou échéance de la tâche i.

S_i: La date de début d'exécution de la tâche i.

C_i: La date de fin d'exécution de la tâche i (complétions time).

T_i: Le retard vrai de la tâche.

$w_i = c_i - r_i$ La durée de séjour de la tâche i depuis sa disponibilité jusqu'à la fin de son exécution (flow time).

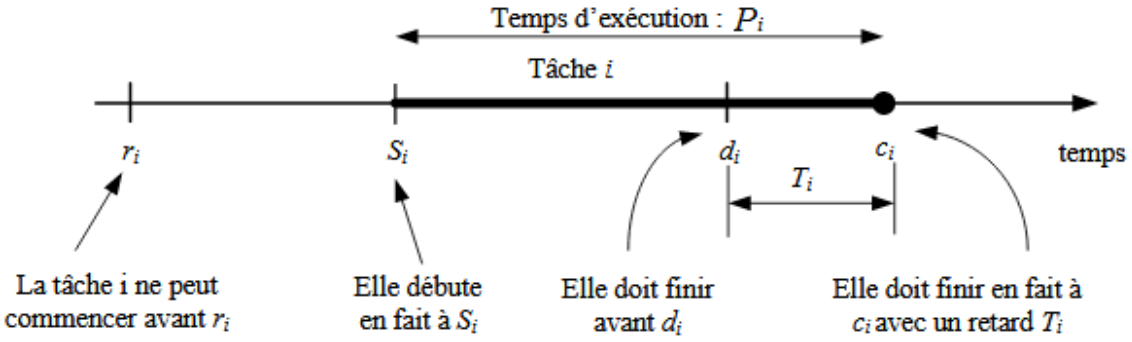


Figure 1.1. Les principaux paramètres d'une tâche.

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. Ce moyen technique est donc nécessaire et indispensable pour le bon fonctionnement du cycle de fabrication. Dans un atelier, plusieurs types de ressources sont distingués :

- Les ressources renouvelables, qui, après avoir été allouées à une tâche, redeviennent disponibles et qui peuvent être réutilisées (machines, personnel, etc.), Lorsqu'elle demeure disponible en même quantité
- Les ressources consommables, qui, après avoir été allouées à une tâche, ne sont plus disponibles, et sont donc épuisées (argent, matières premières, etc.),
- Les ressources partageables qui peuvent être partagées entre plusieurs tâches. Ces ressources peuvent être classées d'une autre manière :
  - Les ressources de type disjonctif qui ne peuvent exécuter qu'une opération ou une tâche à la fois.
  - Les ressources de type cumulatif qui peuvent exécuter plusieurs opérations simultanément.

Une machine est considérée comme un type de ressource qui n'est caractérisé que par son horaire de travail **[Pinedo 2008].**

De nombreuses contraintes peuvent être imposées. Les contraintes s'agissent des conditions à respecter lors de la construction d'un ordonnancement pour qu'il soit réalisable. Les contraintes des problèmes d'ordonnancement expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision. On distingue les contraintes temporelles et les contraintes des ressources.

- ✓ ***Contraintes temporelles*** : elles sont relatives aux dates limites des tâches (délais de livraisons, disponibilité des approvisionnements) ou à la durée totale d'un projet.
- ✓ ***Contraintes de ressources*** : elles expriment la nature et la quantité des moyens utilisés par les tâches, les caractéristiques d'utilisation de ces moyens, la disponibilité des ressources et la quantité des moyens disponibles au cours du temps.

Dans un problème classique d'ordonnancement, une machine ne peut exécuter qu'une tâche à la fois et une tâche ne peut être exécutée que sur une machine à la fois. Une affectation des tâches qui respecte les contraintes liées à l'environnement des machines et aux caractéristiques des tâches est dite *ordonnancement réalisable*. Si l'ordonnancement optimise le critère d'optimalité, il est dit *optimal*.

1.2.2 Production

C'est une opération qui transforme des matières premières et/ou composants en produits finis. Dans une entreprise, la production dépend d'autres fonctions. D'après Bénassy **[Bénassy 1987]**, la position de la production dans un environnement industriel complet est représentée par le schéma simplifié de la figure 1.2.

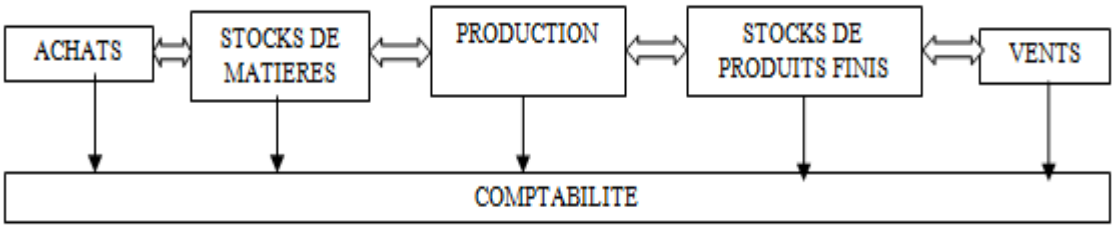


Figure 1.2. Position de la production

La production a quatre objectifs : la quantité, le prix, la qualité et le délai. Le problème principal consiste à arbitrer entre ces objectifs qui sont dépendants entre eux afin d'arriver à un compromis satisfaisant pour le client et pour l'entreprise.

1.2.3 Système de production

Un système de production est constitué d'un ensemble de ressources qui répondent à des objectifs de la production, c'est-à-dire la transformation de matières premières et/ou de composants en produits finis. On distingue quatre types de ressources : les équipements, les hommes, les matières et les informations techniques, procédurales ou relatives à l'état et à l'utilisation du système productif.

Giard **[Giard 1988]** propose deux types de typologie des systèmes de production : la première est basée sur le fait qu'une production peut être réalisée soit pour répondre à une commande, soit pour alimenter un stock et la seconde est liée à la régulation du monde de la production, ce qui varie en fonction de l'importance et de la variété des flux de produits traités par les systèmes productifs. Dans **[Le Moal et Tarondea 1979]** la classification de systèmes de production est présentée dans la figure 1.3 qui inclut les types mentionnés avant.

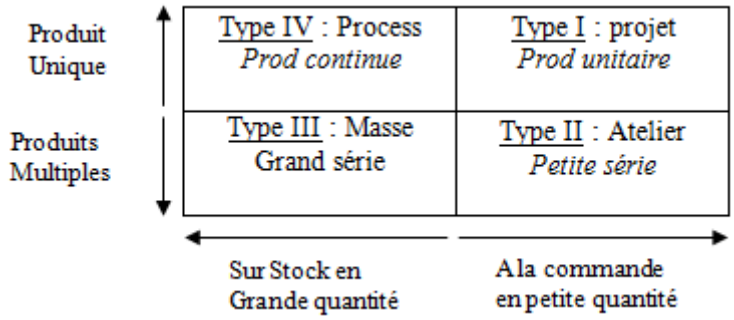


Figure 1.3. Classification de Systèmes de Production

1.3 Problèmes d'ordonnancement d'atelier

Pour représenter les problèmes d'ordonnancement on utilise un système de notation. Ce système contient trois champs comme suit :

Type d'atelier/ caractéristiques des opérations/ critère choisi.

1.3.1 Classification des problèmes d'ordonnancement

Dans ce qui suit, nous introduisons la notation de Graham et al **[Graham et al 1979]**, Cette notation se compose de trois champs α, β et γ :

1.3.1.1 Type d'atelier

Le premier champ $\alpha = \alpha_1\ \alpha_2$ présente l'environnement de la machine. Dans la littérature, trois types d'environnement pour les machines, Chaque environnement peut être divisé en plusieurs autres environnements ci-dessous :

- ***Une seule machine*** : il y a seulement une machine à traiter les tâches ou les jobs.

- ***Machines parallèles****(Figure1.4)* plus d'une machine exécute la même fonction. Les machines peuvent être :

  - Identique : toutes les machines ont la même vitesse et ils peuvent traiter tous les jobs.

  -Uniforme : chaque machine a sa propre vitesse indépendante de la tâche.

  -Unrelated : il n'y a aucune relation entre les machines. Les vitesses d'exécution des machines dépendent des tâches et sont différentes.

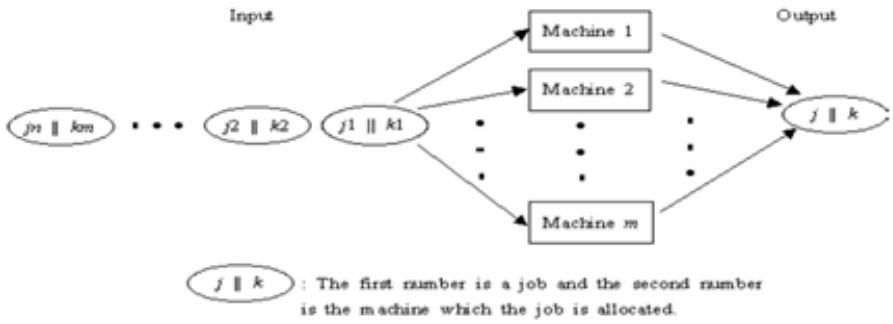


Figure 1.4. Structure du processus machines parallèles.

- Dans le cas de *machines spécialisées*, il y a trois modèles ou types d'exécution de tâches : le *flow shop*, l'*open shop* et le *job shop*. Le tableau1.1nous donne quelques exemples classiques d'atelier

Tableau 1-1 les types d'atelier.

Type	Signification
1	Une seule machine.
Pm	m machines à performances identiques ou machines parallèles
Fm (Flow Shop)	m machines (les tâches utilisent toutes les machines et dans le même ordre)
Im(Job Shop)	m machines (les tâches utilisent toutes les machines et dans un ordre différent)
Om(Open Shop)	m machines (les gammes sont quelconques)

Flow Shop **(F)** : c'est un cas particulier du problème d'ordonnancement d'atelier pour lequel le cheminement des jobs est unique : les *n* jobs utilisent les *m* machines dans l'ordre 1,2, . . ., *m* (lignes de production).

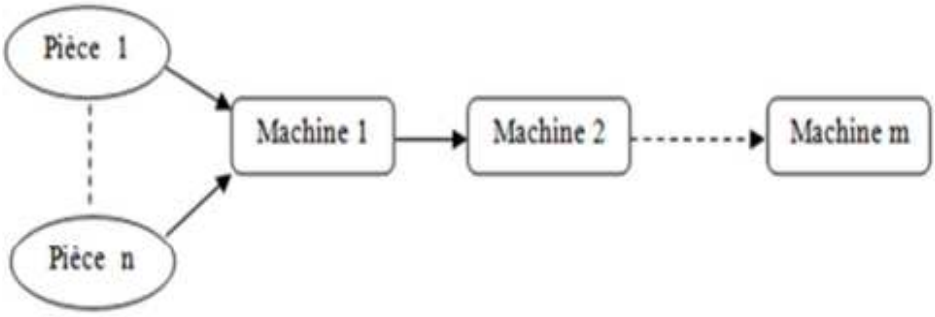


Figure 1.5. Atelier Flow shop

Dans ce type d'atelier (*Figure 1.5),* on dispose de pièces qui doivent s'exécuter suivant le même ordre sur les m machines.

Flow shop hybride : il s'agit d'ateliers flow shop dans lesquels un « étage » donné de la fabrication peut être assuré par plusieurs machines en parallèle ou bien une machine peut être remplacée par un ensemble de machines parallèles identiques. Dans ce type d'ateliers, chaque job passe par chaque étage et l'ordre de passage des jobs sur les étages est le même pour tous les jobs.

Job Shop (J) : Ce type d'ateliers est nommé aussi atelier à cheminements multiples, les *n* jobs sont exécutés sur plusieurs machines, à la différence de celles du flow shop où les jobs doivent obligatoirement visités toutes les machines et que les séquences opératoires relatives aux différents jobs peuvent être distinctes et sont propres à chaque job. Dans les ateliers de type « job-shop », les opérations sont réalisées selon un ordre total bien déterminé, variant selon la tâche à exécuter. Dans ce cas, plusieurs modifications d'outils sont à anticiper et plusieurs routages existes.

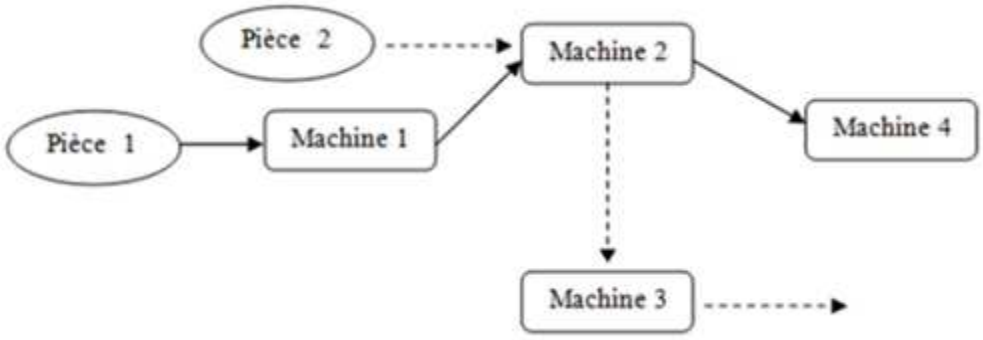


Figure 1.6. Atelier Job shop.

Job-shop flexible est une extension du modèle job shop classique. Le principal avantage est que de nombreuses machines sont capables de réaliser un sous-ensemble

d'opérations. Spécifiquement, une opération est associée à un ensemble contenant toutes les machines capables d'effectuer cette opération.

Open Shop (O) : c'est un modèle d'atelier à cheminement libre où les jobs peuvent visités plusieurs machines plusieurs fois et sans ordre prédéfini. Autrement dit, chaque produit à traiter doit subir un ensemble d'opérations sur un ensemble de machines, mais dans un ordre totalement libre.

Pour simplifier la description des différents problèmes d'ordonnancement qui sont étudiés dans la littérature ainsi que dans le cadre de cette thèse, nous allons prendre la notation de Lawler **[Lawler et al 1989]** qui consiste à décrire un problème d'ordonnancement donné par trois champs séparés par des « slashs » $\alpha|\beta|\gamma$, dont la signification de chaque champ est la suivante :

Le paramètre $\alpha_1 \in \{1 \text{ ou } \varnothing, P, Q, R, O, F, J, FH\}$ caractérise le type de machines utilisées :

$\alpha_1 = 1$: auquel cas, nous avons un problème à une seule machine.

$\alpha_1 = P$: machines identiques parallèles.

$\alpha_1 = Q$: machines parallèles uniformes.

$\alpha_1 = R$: machines parallèles quelconques.

$\alpha_1 = O$: système open shop.

$\alpha_1 = F$: système Flow shop.

$\alpha_1 = J$: système Job shop.

$\alpha_1 = FH$: système flow shop *hybride*.

$\alpha_2 \in \{\varnothing, k\}$ est un entier qui représente le nombre de machines dans le problème.

$\alpha_2 = \varnothing$: le nombre de machines est supposé être variable.

$\alpha_2 = k$: le nombre de machines est égal à k (k entier positif).

1.3.1.2 Caractéristiques des opérations

Le paramètre $\beta = \beta_1\ \beta_2\ \beta_3\ \beta_4\ \beta_5\ \beta_6\ \beta_7\ \beta_8$ représente l'ensemble des contraintes et caractéristiques du processus. Le champ de β spécifie les caractéristiques de job concernant les délais de traitement, les dates de sortie, préséances, etc. tout d'abord, différents modes d'exécution sont possibles : soit avec préemption (l'exécution d'une opération peut être interrompue puis reprise sur la même machine ou sur une autre), soit sans préemption (lorsqu'une opération est commencée, elle doit être terminée avant de pouvoir exécuter une autre opération sur la même machine). En outre, r_j indique les dates de disponibilité pour les jobs et *pmtn* indique que la préemption des jobs est autorisée. Si les relations de précédence sont données, *prec* est utilisé (*ou des chaînes, arbre, sp-graphe* si les précédences sont une

structure spéciale). De plus, dans un environnement des tâches multiprocesseur avec les processeurs parallèles, la $size_j$ d'entrée dit que le job j a besoin de plusieurs processeurs simultanément.

- Le paramètre $\beta_1 \in \{ø, pmtn\}$ indique la possibilité de la préemption de la tâche.
  ø : La préemption n'est pas autorisée, pmtn : préemption autorisée.
- $\beta_2 \in \{ø, res\}$ caractérise les ressources additionnelles.
  ø Aucune ressource additionnelle n'existe, res : contraintes de ressources spécifiées.
- $\beta_3 \in \{ø, prec, tree, chains\}$ reflète les contraintes de précédence.
  ø : Les tâches sont indépendantes, prec : contraintes de précédence quelconques, tree : contraintes de précédence formant un arbre, chains : contraintes de précédence formant une chaînes.
- $\beta_4 \in \{ø, r_j\}$. ø: Tous les instants au plutôt sont nuls, r_j :tous les instants au plutôt sont différents.
- $\beta_5 \in \{ø, p_j = p, p \leq p_i \leq p\text{--}\}$. ø: Les durées d'exécution sont arbitraires, égaux ou dans l'intervalle défini.
- $\beta_6 \in \{ø, \check{d}\}$ décrit les deadlines. ø: Aucune date de fin impérative, d : des dates sont imposées.
- $\beta_7 \in \{ø, n_j \leq k\}$ décrit le nombre maximum d'opérations constituant une tâche. ø: Aucune contrainte, le nombre est inférieur ou égal k.
- $\beta_8 \in \{ø, no\text{-}wait\}$ décrit pour un problème à machines spécialisées, une propriété d'attente. ø: La région tampon est de capacité illimitée, no-wait : sans attente, les capacités d'attente sont nulles, une tâche qui termine son exécution sur une machine passe instantanément sur une autre machine.

1.3.1.3 Critère choisi

Enfin, le paramètre γ concerne les critères d'évaluation d'un ordonnancement. Les objectifs visés sont liés à une bonne utilisation des ressources, une minimisation du délai global ou encore le respect d'un maximum de contraintes. Des exemples de possibles fonctions objectives pour être minimisée sont :

- Makespan (C_{max}) Le makespan, défini comme max (... C_1, C_m), est équivalente à la date de réalisation de la dernière tâche. Un makespan minimum implique généralement une bonne utilisation de la machine (m).
- Maximum Lateness (L_{max}) Le retard maximum, L_{max}, est défini comme max $(L_1,..., L_n)$. ce critère mesure la plus grande violation des dates d'échéance souhaitées.

- Total weighted completion $(\sum_j^m w_j C_j)$ somme pondérée des dates de fin d'exécution. Ce critère représente la somme des encours des tâches dans le système.
- **Exemples :**

1) P‖Cmax désigne le problème de minimiser la longueur d'ordonnancement de tâches indépendantes, arrivant aux instants 0, à machines parallèles identiques. La préemption n'est pas autorisée.
2) $1|r_j, prmp|\sum_j^n w_j C_j$: dénote un problème à une machine dont les tâches ne sont disponibles qu'à la date r_j. Lors de l'exécution d'une tâche, la préemption notée *prmp* est possible. Le but est de minimiser la somme des temps d'accomplissements pondéré ou $\sum_j^n w_j C_j$
3) $J2||Tmax$: représente le problème de job shop à deux machines, noté *J2,* dont l'objectif est la minimisation du retard total.

Ainsi, nous pouvons classifier les problèmes d'ordonnancement dans cet atelier à partir du nombre de machines et l'ordre utilisé pour la fabrication du produit, qui dépend de la nature de l'atelier impliqué. Comme le montre la figure 1.7:

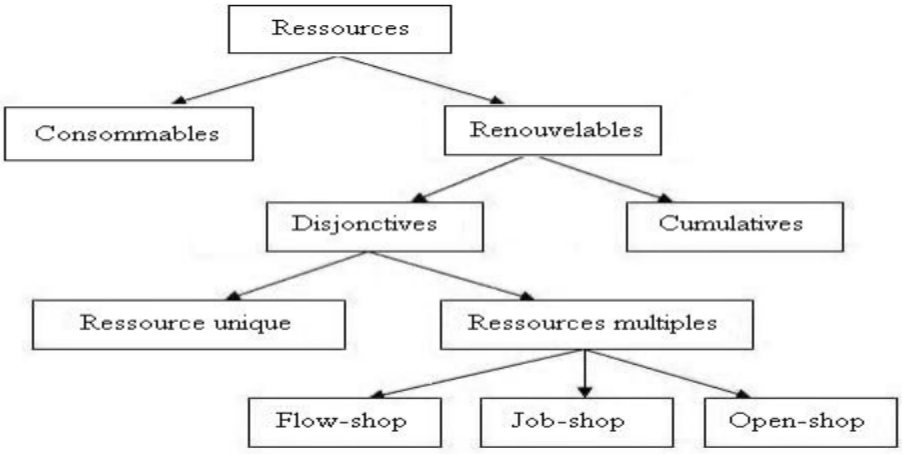


Figure 1.7. Classification des types d'ateliers

1.4 Représentations des ordonnancements

Il existe trois sortes de représentations pour les solutions d'un problème d'ordonnancement d'atelier le diagramme de Gantt, le graphe disjonctif et la méthode PERT.

1.4.1 Diagramme de Gantt

Le diagramme de Gantt est un outil permettant de montrer la planification des séquences de traitement sur chaque machine nécessaire à la réalisation d'un projet. Il s'agit d'un outil élaboré en 1917 par Henry L. Gantt. Le diagramme de Gantt présente la liste des tâches, notées

T_i à exécuter par les machines notées M_j et en abscisse on a l'échelle du temps, comme le montre la figure 1.8, il se compose de lignes horizontales désignant les machines ; les opérations y sont représentées, en fonction des machines correspondantes, à partir de leur date de début d'exécution, sous forme de barres.

Un ordonnancement est défini grâce aux séquences de tâches, et leurs dates de début. En général, un ordonnancement est représenté sous forme d'un diagramme, appelé diagramme de Gantt. Il permet de visualiser, en fonction du temps, l'exécution des différentes opérations pour résoudre un problème d'ordonnancement, c'est de trouver correspondance entre les moyens disponibles et une tâche à effectuer. Cela consiste donc à déterminer les dates de début t_y de chaque processus S_y et leur allocation des ressources, de façon à ce que les contraintes soient respectées. Pour cela, il faut déterminer l'ordre de passage des tâches sur chaque machine. Cet ordre de passage est appelé séquence. Une séquence est une permutation de n tâches ou autrement dit, l'ordre dans le quelles n tâches sont alloués à une machine.

- **Exemple**

Par exemple, si un gestionnaire souhaite représenter l'ordonnancement de la pose d'une porte de voiture dans le cadre de la chaîne de montage déjà mentionnée. Comme décrit dans le Tableau 1.2, la tâche de pose d'une porte est composée de cinq opérations. Puis que dans cet exemple, il n'y a qu'une tâche, i est la machine sur laquelle l'opération devra se réaliser.

Tableau 1-2 Description des processus d'installation de porte.

Opérations	Saisir la porte	Placer la porte dans les charnières	Placer les boulons	Visser les boulons	Vérifier que la porte est bien fixée
Notations	Q_{11}	Q_{12}	Q_{21}	Q_{22}	Q_{13}
Durée p_{ij}	$p_{11} = 1$	$p_{12} = 2$	$p_{21} = 2$	$p_{22} = 0.5$	$p_{13} = 1$
Date de fin d'exécution t_{ij} (minutes)	$t_{11} = 1$	$t_{12} = 3$	$t_{21} = 5$	$t_{22} = 5.5$	$t_{13} = 6.5$

Les opérations sont supposées se dérouler sans interruption. Pour réaliser ces opérations. Les deux machines notées M_1 et M_2, elles sont disponibles au temps =0. Les opérations : Saisir la porte notée O_{11}, mettre la porte dans les charnières, notée O_{12}, examiner que la porte est bien fixée notée O_{13}, s'effectuent sur la machine M_1. Quant aux opérations, placer les boulons, notée O_{21} et les visser notée O_{22} c'est la machine M_2 qui s'en charge. Comme le montre la Figure 1.8, nous obtenons le diagramme de Gantt suivant :

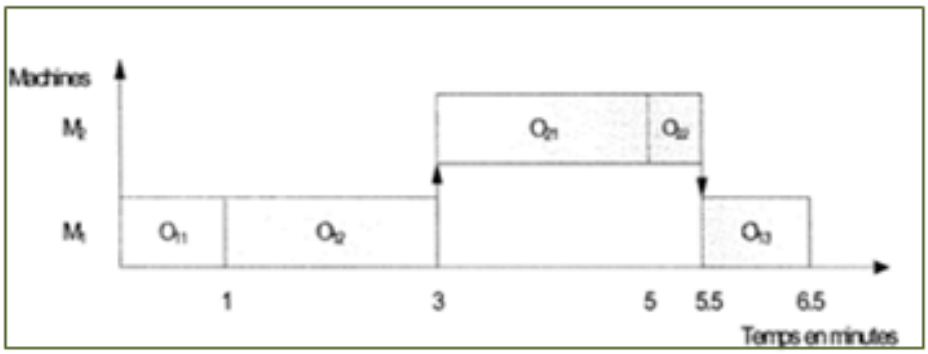


Figure 1.8. Diagramme de Gantt

Grâce au diagramme de Gantt, si le gestionnaire de la chaîne de montage veut savoir combien de temps il faudra pour mettre la porte, il a seulement besoin de lire le diagramme et il aura un temps d'achèvement de 6.5 minutes.

1.5 Évaluation d'un problème d'ordonnancement

Dans la résolution d'un problème d'ordonnancement, un ou plusieurs objectifs construits sur la base d'indicateurs de performance sont pris en considération.

Les critères à optimiser sont exprimés sous forme d'une fonction appelée fonction objectif : c'est le but que l'on cherche à atteindre en résolvant un problème d'ordonnancement. La fonction objective est une fonction de minimisation ou de maximisation d'un ou plusieurs critères. Les critères d'optimalité les plus utilisés font intervenir la durée totale de l'ordonnancement, le délai d'exécution, les retards de l'ordonnancement, etc.

Nous cherchons donc à minimiser ou à maximiser de tels objectifs qui peuvent être liés au temps, aux ressources ou à des coûts, tels ceux de lancement, de production, de stockage, etc.

La durée totale d'ordonnancement, notée C_{max} *(makespan)*, C'est la *longueur d'ordonnancement* (représente le temps de fin de n job). Minimiser le retard total (*Tardiness)* noté $\sum_j^n D_j$ qui est égal à la somme des retards de tous les travaux ; $\sum_j^n D_j = \sum_j^n \max(C_j - d_j, 0)$ où *Cj* représente le temps defin du travail *j, n* le nombre de travaux et d_j la date due du travail

j. Le flow time pondéré : $\sum_j^m w_j C_j$ permet d'estimer le coût des stocks d'encours. En effet, une tâche est présente dans l'atelier entre les instants r_j et C_j. Les stocks dont elle a besoin doivent être disponibles entre ces deux dates, d'où le coût : $\sum_j^n w_j(c_j - r_j)$ égale, à une constante prés, à : $\sum_j^m w_j c_j$.

Par la suite, il ne sera essentiellement question que du makespan comme fonction objectif. Cette fonction étant le critère le plus souvent utilisé pour évaluer le coût d'un ordonnancement.

1.6 Complexité des problèmes d'ordonnancement

La théorie de l'ordonnancement comporte également une variété de méthodologies. En effet, le domaine de l'ordonnancement est devenu un point focal pour le développement, l'application et l'évaluation des procédures combinatoires, techniques de simulation, et des approches de solutions heuristiques.

La sélection d'une méthode appropriée dépend principalement de la nature du modèle et le choix de la fonction objectif. Dans certains cas, il est logique de considérer des techniques alternatives. Pour cette raison, il est important d'étudier les méthodes ainsi que des modèles. Un point de vue utile sur la relation des problèmes d'ordonnancement et de leurs techniques de solution vient de développements d'une branche de la science informatique appelé théorie de la complexité.

Pour les problèmes calculables, il existe des algorithmes pouvant les résoudre selon deux critères importants qui sont le temps et l'espace en d'autres termes les algorithmes résolvant un problème donné sont-ils efficace ? Si le temps de calcul est trop grand alors l'algorithme sera inefficace. La fonction de complexité mesure les ressources utilisées pour sa résolution, elle est influence par la taille du problème, pour trouver la solution au problème donné elle est définie en fonction du nombre d'instructions à exécuter donc nous ne nous intéressons qu'à la complexité temporelle. La complexité est une théorie qui permet de classer les problèmes de décision en problèmes « faciles » et « difficiles » à résoudre qui se définit comme étant le temps nécessaire d'un problème. Cook **[Cook 1998]** a élaboré une théorie de la complexité en mesurant le temps de calcul en fonction de la taille du problème. Sa théorie distingue entre les algorithmes polynomiaux, de complexité de l'ordre d'un polynôme, et les autres dits exponentiels. La puissance de calcul ne résout rien pour des complexités exponentielles, il est faux de croire qu'un ordinateur peut résoudre tous les problèmes combinatoires. La complexité permet de répondre à la question suivante : étant donnée un problème x, peut on trouver un algorithme efficace pour sa résolution ? D'après SAK **[Saka 1984]** il définit qu'un algorithme

efficace est un algorithme dont le nombre d'opérations nécessaires pour la résolution de x est borné par une fonction polynomial d'un paramètre caractérisant la taille du problème.

Dans la littérature, il existe plusieurs classes de complexité, mais les plus connues sont les problèmes de la classe *P* et les problèmes de la classe *NP*. Un problème de décision sera dit polynomial où appartenant à la classe P s'il existe un algorithme polynomial qui résout. Un problème appartient à la classe NP s'il existe un algorithme qui permet de vérifier qu'une réponse est correcte. Par exemple, dans le cas problème du voyageur de commerce, il n'existe pas d'algorithme polynomial qui permette de répondre à la question : existe-t-il un parcours de longueur inférieure à L ? Par contre on peut vérifier par un algorithme polynomial si une réponse oui est juste.

Les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoire *(POC)* dont la complexité est établie pour la plupart d'entre eux. La majorité des problèmes d'ordonnancement se classe dans la catégorie des problèmes dits *NP-difficiles* **[Garey 1979].** L'ordonnancement est un problème complexe à résoudre, qui prend un temps prohibitif avec des méthodes exactes. C'est pour cela que l'utilisation d'heuristiques est la seule solution acceptable dès que la taille du problème devient grande.

1.7 Méthodes de résolution des problèmes d'ordonnancement

Les problèmes d'ordonnancements sont des problèmes NP-difficiles, il existe une multitude de méthodes pour les résoudre. La résolution d'un problème d'ordonnancement est constituée de deux étapes principales. La première étape consiste à identifier et à modéliser le problème en décrivant les contraintes qui doivent être respectées et mettant en valeur les critères à optimiser. La deuxième étape se traduit par la recherche de la méthode adéquate pour résoudre le problème considéré. En effet, l'exploitation de plusieurs méthodes est importante afin de trouver une solution qui permet, par la comparaison de leur efficacité d'établir des décisions robustes.

Les méthodes exactes s'appliquent aux problèmes qui peuvent être résolus de façon optimale et rapide et de prouver son optimalité pour toutes les instances t. Elles ont l'avantage d'obtenir des solutions dont l'optimalité est garantie. Leur inconvénient majeur est le temps exponentiel de résolution qui croît en fonction de la taille du problème. Ces méthodes sont basées soit sur une résolution algorithmique ou analytique, soit sur une énumération exhaustive de toutes les solutions possibles. Comme méthodes exactes appliquées aux problèmes d'ordonnancement de production, nous citons la programmation linéaire (PL), la

programmation linéaire en nombres entiers, les procédures par séparation et évaluation progressive (Branch and Bound), la relaxation lagrangienne, la programmation dynamique…. Au contraire, les méthodes approchées visent à générer des solutions de haute qualité en un temps de calcul raisonnable, mais il n'existe aucune garantie de trouver la solution optimale. Ces méthodes approchées sont elles-mêmes divisées en deux sous-catégories : les algorithmes d'approximation et les heuristiques. Contrairement aux heuristiques, les algorithmes d'approximation garantissent la qualité de la solution trouvée par rapport à l'optimal. En fin, il existe encore deux sous-classes de méthodes heuristiques : les heuristiques spécifiques à un problème donné, et les métaheuristiques qui regroupent des méthodes plus génériques.

Il existe pour plusieurs problèmes d'optimisation combinatoire avec une taille critique de l'espace de solutions admissibles. La méthode permettant d'obtenir une solution optimale est bien évidemment celle de l'énumération complète de l'espace de recherche. Cette dernière est dans la plupart des cas prohibitive malgré l'évolution permanente des calculateurs et les progrès permanents de l'informatique.

Compte tenu de ces difficultés, la plupart des spécialistes de l'optimisation combinatoire ont orienté leur recherche vers le développement de méthodes heuristiques. Une méthode heuristique est souvent définie comme une procédure exploitant au mieux la structure d'un problème, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [**Roubellat 2001].** D'autres méthodes, nommées métaheuristiques, sont plus générales et peuvent être appliquées à plusieurs catégories de problèmes d'ordonnancement combinatoire *POC*. Les métaheuristiques les plus connues sont la recherche locale *(RL),* le recuit simulé *(RS),* la recherche avec tabous *(RT),* les algorithmes génétiques *(AG)* et l'optimisation par colonie de fourmis *(OCF)*…..

L'intelligence artificielle a fourni elle aussi de nouvelles approches et tendances en ordonnancement depuis deux décennies, notamment par l'utilisation de réseaux de neurones [**Sabuncuoglu 1998**] ou de systèmes experts **[Smith 1992].** L'utilisation de la logique floue dans l'ordonnancement n'a pas pour but de proposer une nouvelle méthode, mais d'apporter ponctuellement des techniques qui vont permettre d'être plus proche de la réalité. L'incertitude et l'imprécision caractérisant les données d'une entreprise, bien que fréquemment constatées, sont rarement prises en compte à court terme **[Sabuncuoglu 1998].** La gestion de cette incertitude et de cette imprécision est plus souvent faite par le biais de méthodes statistiques, mais l'utilisation de la logique floue et de la théorie de possibilités a connu un succès certain **[Souier 2010]**.

Les méthodes d'ordonnancement issues de l'intelligence artificielle sont toutefois peu implantées dans l'industrie car elles ne sont pas encore suffisamment matures ni suffisamment transparentes pour l'utilisateur.

1.8 Conclusion

Dans ce chapitre, nous avons abordé les spécifications d'ordonnancement dans les systèmes de production. Nous avons tout d'abord présenté les caractéristiques générales des systèmes de production. Puis, nous avons décrit la fonction ordonnancement d'une manière générale. Ensuite, nous avons abordé la problématique de la modélisation des problèmes d'ordonnancement et les méthodes de résolutions. Ces problèmes d'ordonnancement dans les systèmes de production sont dans le cas général des problèmes de type NP-difficile ou encore de complexité ouverte, ce type de complexité justifie l'utilisation de méthodes de résolution approchées pour traiter des instances de problème de taille importante.

L'évolution technologique observée depuis le début des années 80 dans le domaine de la production a permis de mettre en lumière la capacité de l'IA à traiter ces problématiques complexes dans un contexte industriel. Le chapitre suivant est dédié à une classe particulière de méthodes approchées de type heuristique qui sont les algorithmes évolutionnaires. Ces derniers sont des applications de l'IA. Ils sont utilisés pour résoudre les problèmes d'ordonnancement dans les systèmes étudiés dans cette thèse.

Chapitre 2

2 Intelligence Artificielle et Les algorithmes évolutionnaires

Dans ce chapitre nous nous intéressons de manière approfondie à l'intelligence artificielle (IA), nous citerons quelques méthodes rattachées à ce domaine. Il contient les éléments nécessaires à la compréhension de ces méthodes. Il est composé de deux parties principales, la première traite de manière générale ces techniques, leurs définitions, leurs domaines d'utilisation, leurs caractéristiques. La seconde partie est réservée à la présentation des algorithmes évolutionnaires et les variant des algorithmes génétiques les plus utilisées en incluant leurs définitions, leurs origines et leurs algorithmes de base.

Sommaire

2.1 Introduction

Nous avons dit dans le chapitre précédent qu'il n'existait pas d'algorithme efficace pour résoudre les problèmes d'optimisation NP-complets. Rappelons qu'un algorithme efficace veut dire un algorithme pouvant trouver une solution optimale du problème en un temps borné par une fonction polynomiale. Des problèmes aussi importants que les problèmes d'ordonnancement de système de production ont suscité l'intérêt de nombreux chercheurs. Plusieurs méthodes ont été élaborées en intelligence artificielle (IA). La recherche s'est orientée vers des heuristiques spécifiques aux problèmes, puis elle s'est progressivement intéressée aux méthodes plus générales, c'est-à-dire les métaheuristiques spécialement les algorithmes évolutionnaires. Tout cela avait pour but faire un compromis intéressant entre le temps d'exécution et la qualité de solution obtenue. Il existe dans la littérature une panoplie de méthodes de résolution de problèmes NP-complets, ces méthodes peuvent être classées globalement en deux familles : les méthodes exactes qui garantissent la solution optimale, leur inconvénient majeur est le temps exponentiel de résolution qui croit en fonction de la taille du problème. Les méthodes approchées qui ne garantissent pas l'optimum mais qui sont plus efficaces car elles donnent une solution proche de l'optimum, voir l'optimum en un temps raisonnable.

2.2 Intelligence Artificielle

L'IA n'est absolument pas quelque chose de nouveau pour nous, nous l'utilisons tous les jours. Que nous envoyons un e-mail, que nous utilisons une carte bancaire, que nous voyageons ou que nous lançons une recherche sur Internet, l'IA est le socle sur lequel toutes ces activités sont basées. Les algorithmes intelligents vérifient et détectent constamment des fraudes à la carte de crédit, des avions qui décollent et atterrissent, suivent en permanence des niveaux de stocks et réalisent même des produits dans des usines robotisées.

Au début des années 80, beaucoup des grandes entreprises commencèrent à s'intéresser à l'IA et à former leur propre équipe de recherche ce qui leur permit d'économiser des dizaines de millions de dollars**.** Analyser comment les business plans vont évoluer avec ces technologies est un travail sans fin. Mais une chose est claire, le monde du travail est en plein mouvement et s'adapter à ces changements est la seule façon de survivre économiquement. Comme le dit le chief product officer d'Uber, Jeff Holden : « Si vous vous tournez vers le futur, nous allons connaître des changements structurant en intelligence artificielle qui vont impacter les modèles économiques et les opportunités de business. »

Nous sommes sur le point de constater des impacts bien plus larges de l'IA sur l'industrie, les entreprises et le monde économique. Ce n'est que le début de l'aventure pour l'humanité qui va assister à l'arrivée de l'IA, et à la découverte de nouveaux horizons dessinés par cette technologie. L'intelligence artificielle (IA) consiste à faire reproduire, par un système informatique, le raisonnement humain et l'acquisition de la connaissance. L'intelligence artificielle a été imaginée dans les années 50, notamment par Turing. Il s'agit là d'une entreprise fabuleuse dont la mise en œuvre a été rendue possible dès le début des années 80 par le développement de l'informatique personnelle.

Pour continuer, nous citerons quelques méthodes rattachables au domaine de l'intelligence artificielle

2.2.1 La logique floue

Les bases théoriques de la logique floue (fuzzy logic) ont été établies pour la première fois en 1965 lorsque le professeur Lotfi A. Zadeh, de l'université de Berkeley aux USA, publie un article intitulé « Ensembles flous ».

La logique floue **[Kaufmann 1992]**, est une partie de l'intelligence artificielle qui associe les notions de « sous-ensemble flou » et de « théorie des possibilités, visant essentiellement à formaliser de manière « informatisable » l'imprécision et l'incertitude considérées comme inhérentes à beaucoup de connaissances humaines. Il est plus intuitif que la logique classique dans le mode de raisonnement. Il permet aux concepteurs de mieux appréhender les phénomènes naturels, imprécis et difficilement modélisables en s'appuyant sur la définition de règles et de fonctions d'appartenance à des ensembles dits « ensembles flous ». Un domaine d'application de la logique floue qui devient fréquent est celui du réglage et de l'ordonnancement des systèmes industrielles. Cette méthode permet d'obtenir une loi de commande souvent efficace, sans devoir faire appel à des développements théoriques importants. Elle présente l'intérêt de prendre en compte les expériences acquises par les utilisateurs et opérateurs du processus à commander.

L'utilisation de la logique floue à l'ordonnancement peut être divisée en deux grandes catégories

- Intégrer dans des systèmes experts des connaissances imprécises et incertaines.
- Décrire des contraintes « flexibles » en propagation de contraintes **[Roubellat 2001],** ce qui ne va plus poser le problème de décrire une base de connaissance complexe.

2.2.2 Systèmes experts

La notion de systèmes experts est une notion assez ancienne qui a apparu dans les années 70 avec l'apparition du système expert célèbre MYCIN dont le but était d'aider les médecins à effectuer le diagnostic est le soin des maladies infectieuses du sang.

Un système expert utilise la connaissance correspondante à un domaine spécifique afin de fournir une performance comparable à l'expert humain. Il se compose d'une base de faits (Code la connaissance sur l'étude en cours. Son état évolue en cours d'expertise (mémoire de travail)) qui regroupe les suppositions « vraies » et d'une base de règles qui renferme la connaissance sur le problème à traiter (Code la connaissance sur le domaine. Fixe pour plusieurs expertises). Un moteur d'inférence permet donc de déterminer les conditions des règles qui sont vérifiées et les conséquences qui peuvent en être conclues et ajoutées à la base de faits **[Kaufmann 1992]**. Les problèmes combinatoires d'ordonnancement ont fait que, beaucoup de tentatives ont été faites pour mettre en boîte des connaissances sur le domaine ou sur un atelier donné pour orienter l'élaboration d'un ordonnancement. Ces expériences se sont trouvées face à la difficulté du problème, où il y'a peu de connaissances génériques semblent exister et le développement d'une base de connaissances relative à un atelier donné demande un effort considérable. De plus, les connaissances mises en jeu en ordonnancement semblent peu se plier à un schéma aussi binaire que celui de règles de production simples **[Farbe 2009].** L'utilisation du système expert est fréquemment pour améliorer des ordonnancements déjà existants. Dans, les auteurs proposent de faire un ordonnancement par satisfaction de contraintes, puis d'améliorer le résultat grâce à un système expert. On peut aussi citer le système OPIS **[Ow et al 1988]**, qui détermine grâce à un système expert basé sur l'utilisation de tableaux noirs, les ressources goulots et relance un ordonnancement par satisfaction de contraintes en planifiant ces ressources en premier, plusieurs autres ont pensé **[Bel et al 1988]** et **[Bensana et al 1988]** à utiliser le système expert pour résoudre la complexité des problèmes d'ordonnancement et surmonter leur difficulté, ceux proposés par O'kane **[O'kane 2000]**, Ruiz et al. **[Ruiz et al 2001]**, Kovàcset al. **[Kovàcs et al 1994],** Li et al. **[Li et al 2006].**

2.2.3 Réseaux de neurones

Parmi les autres branches de l'intelligence artificielle on peut citer les réseaux de neurones. Cette section non seulement imite l'esprit humain mais aussi la structure du cerveau et en particulier ses capacités d'apprentissage. Les réseaux de neurones artificiels sont introduits par W. James en 1890, célèbre psychologue américain. Il a introduit le concept de mémoire

associative, et proposé ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb **[Heeb 1949].**

Généralement un réseau de neurones est organisé en couches, connectées par des processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit en entrée. Le nombre de couches, le nombre de neurones sur chaque couche ainsi que le mécanisme d'interconnexion sont des paramètres pour spécifier l'aspect structurel et fonctionnel du réseau. Toute structure hiérarchique de réseaux est évidemment un réseau

Beaucoup de problèmes d'ordonnancement ont utilisé plusieurs types de réseaux de neurones. L'application des réseaux de neurones et des algorithmes génétiques aux problèmes d'ordonnancement classiques a été présentée dans les travaux de Lee et *al*. **[Lee et al 2000].** Dans la thèse d'Abada **[Abada 1997],** une citation détaillée de ces différents types de réseaux de neurones a été effectuée. Seule une description qualitative est effectuée dans cette thèse Sik et *al* **[Sik et *al*. 2003]** proposent pour le choix de règles de décision dans des problèmes d'ordonnancement comme la gestion des conflits, le couplage des réseaux de neurones avec la simulation **[Senties 2007]**. Il y a aussi les travaux de Akyol et Araz **[Akyol 2007],** Wang et al. **[Wang et al 1995],** Min et al. **[Min et al 1998]** qui ont tous utilisé les réseaux de neurones pour résoudres les problèmes d'ordonnancement.

- **Réseaux de Hopfield**

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés. Dans ce type de réseau, chaque neurone est connecté à chaque autre neurone et il n'y a aucune différenciation entre les neurones d'entrée et de sortie. L'application principale des réseaux de Hopfield est l'entrepôt de connaissances mais aussi la résolution de problèmes d'optimisation. La fonction d'énergie est exprimée de la manière suivante **[Dreyfus et al. 2004]** :

$$E = -\frac{1}{2}\overrightarrow{V^t}.w.\vec{V} - \vec{I}.\vec{V} \quad (1)$$

- Le réseau peut être caractérisé par une fonction d'énergie.
- Une fonction objectif remplace la fonction d'énergie.
- L'optimisation consiste à minimiser la fonction objectif.

L'idée de Hopfield est que, puisqu'un réseau neuronal va chercher à minimiser la fonction d'énergie, il sera possible de construire un réseau pour la minimisation de cette fonction en associant les variables du problème d'optimisation aux variables de la fonction d'énergie.

En d'autres termes, cela revient à mettre en équation le problème d'optimisation sous une forme équivalente à la fonction d'énergie (1). Ce dernier évolue à partir d'un état initial (pris par exemple aléatoirement) de façon à réduire son énergie ; l'évolution se fait ainsi jusqu'à ce que le réseau atteigne son équilibre. L'ensemble des états des neurones sur lesquels il s'est stabilisé correspondra à une solution au problème.

Hopfield a montré que l'énergie du réseau décroît à chaque changement d'état (l'état de S) donc converge vers un état stable qui est appelé attracteur dépendant de l'état initial. Cet état stable correspond à un minimum local de la fonction d'énergie.

2.2.4 Algorithmes Évolutionnaires (variantes des algorithmes génétiques)

De tout temps, les sciences de la vie et les processus naturels ont constitué des bases d'inspiration et d'imitation pour les chercheurs et les ingénieurs. Les mécanismes du monde vivant sont à l'origine des systèmes artificiels utilisables dans des contextes variés. Les méthodes évolutives qui sont présentées dans cette section constituent la base d'un nouveau champ de la programmation informatique en pleine effervescence.

A la fin des années 1950, les algorithmes évolutionnaires (AE) sont apparus. Ils utilisent des techniques de recherche inspirées par l'évolution biologique des espèces. Du premier abord, les algorithmes évolutionnaires ont suscité un intérêt minime, ceci étant principalement dû à leur important coût d'exécution.

Contrairement aux méthodes de recherche locale qui font intervenir une solution unique, les méthodes évolutives manipulent un groupe de solutions admissibles à chacune des étapes du processus de recherche. L'idée centrale consiste à utiliser régulièrement les propriétés collectives d'un ensemble de solutions distinguables, appelé population, dans le but de guider efficacement la recherche vers de bonnes solutions dans l'espace de recherche.

Cette classe d'algorithmes (les algorithmes évolutionnaires) regroupe plusieurs paradigmes incluant les algorithmes génétiques **[Globerg 1989]**, la stratégie d'évolution **[Hoffmeister 1991], [Schwefelet al 1998],** la programmation évolutionnaire **[Fogel 1993]**, la recherche dispersée **[Glover 1995]** et les algorithmes mémétiques **[Moscato1993]**.

En règle générale, la taille de la population reste constante tout au long du processus. Après avoir généré une population initiale de solutions, une méthode évolutive tente d'améliorer la qualité moyenne de la population courante en ayant recours à des principes d'évolution naturelle **[Widmer et al 2001].** Les algorithmes génétiques constituent à ce jour, l'approche la plus utilisée parmi les méthodes évolutives. Cette méthode donne souvent de très

bons résultats aux problèmes d'optimisation, mais un grand nombre de solutions sont susceptibles d'être évaluées, ce qui peut demander un temps de traitement important.

L'intelligence artificielle est susceptible d'être d'un grand intérêt pour l'ordonnancement, lui permettant notamment d'être plus proche de la réalité. Les méthodes d'ordonnancement issues de l'intelligence artificielle sont toutefois peu implantées dans l'industrie car elles ne sont pas encore suffisamment matures ni suffisamment transparentes pour l'utilisateur.

L'apprentissage d'une démarche scientifique pour aborder des problèmes d'optimisation, les résoudre et présenter les résultats obtenus est également visé par ces méthodes d'intelligence artificielle tel que les algorithmes évolutionnaires, l'algorithme du recuit simulé, l'algorithme génétique, la recherche avec tabous et l'optimisation par colonie de fourmis.

Le terme génétique en informatique désigne un outil du monde de l'intelligence artificielle, permettant de converger vers un optimum (jeu de coefficients la plupart du temps) de configuration d'un système (matériel ou logiciel). Par rapport à une approche "brutale" consistant à évaluer toutes les combinaisons possibles, cette approche permet de traiter des problèmes dont la combinatoire est telle (dite "explosive") qu'une exploration systématique prendrait des temps infinis même avec des machines très puissantes.

Il faut bien avoir conscience que, du fait des principes utilisés par cette méthode, elle ne garantit pas de trouver l'optimum absolu (qui d'ailleurs n'existe pas la plupart du temps si l'optimisation est de type multi-objectifs) mais de converger vers un optimum représentant un très bon compromis entre le niveau de qualité de la solution et le temps de calcul mis pour la trouver. Ce n'est donc pas une méthode formelle exacte, mais une méthode de résolution approchée. Autre point important : la qualité du résultat dépend également des paramètres de réglage de la méthode (critère d'évaluation des solutions candidates, critères de convergence, méthode et paramètres de combinaison des générations successives, ...). Il est donc important de bien garder à l'esprit que ce n'est pas un outil qu'il suffit de lancer pour résoudre une équation, mais d'un outil d'aide à l'exploration de l'espace des solutions permettant de converger rapidement vers une solution approchée de bonne qualité.

Tout d'abord, cette solution est itérative, mais n'utilise pas la "force brute" consistant à croiser toutes les combinaisons possibles comme cela a été présenté en introduction. Comme en génétique naturelle, il faut que des parents s'assemblent pour donner des enfants, qui à leur tour se croiseront pour refaire des générations successives.

L'analogie avec la génétique du vivant prend tout son sens quand on décortique les mécanismes d'IA : comme en biologie, il faut une part de hasard pour obtenir les meilleures combinaisons. Prendre mécaniquement les meilleurs scores ne permettent pas d'atteindre toujours les meilleures solutions génétiques et il faut réintroduire des valeurs plus faibles. Il s'agit en fait de sortir de ce qu'on nomme un "maximum local.

2.2.4.1 Algorithme génétique

Ces algorithmes d'IA se fondent sur le système expert, la logique floue, les réseaux neuronaux artificiels. Un autre composant de l'IA sont les algorithmes qui sont utilisés pour la créer. Un bon exemple de cela est l'algorithme génétique (AG), qui est une méthode de recherche heuristique utilisée en IA et dont l'approche est inspirée de la théorie de Darwin sur l'évolution. L'AG est utilisée pour trouver des solutions optimisées à des problèmes liés aux théories de la sélection naturelle et de l'évolution biologique comme la sélection, la mutation, l'héritage et la recombinaison.

Les variantes des algorithmes génétiques AG sont au cœur des approches proposées dans cette thèse. Dans la présente section, nous précisons le vocabulaire nécessaire à la compréhension des AG et nous identifions les principaux opérateurs et composantes de ces algorithmes.

Charles Darwin a observé des phénomènes naturels il y a deux siècles et a fait plusieurs observations **[Darwin 1859]**. En effet, l'évolution opère en réalité directement sur l'ADN contenu dans les chromosomes des êtres vivants. La première description du processus des algorithmes génétiques qui sont le type d'algorithme le plus connu et le plus utilisé des algorithmes évolutionnaires a été donnée par Holland en 1975 **[Holland 1975]**, puis approfondie par Goldberg **[Goldberg 1989].**

Il est nécessaire d'introduire un certain nombre de termes avant d'aborder plus précisément les algorithmes génétiques. Par conséquent, nous définissons les termes suivants (tableau 2.1) :

Tableau 2-1 des définitions des termes nécessaire pour AG

Les termes scientifiques	**Définition**
Gène	On appellera gène la suite de symboles qui codent la valeur d'une variable.
Chromosome	Un chromosome est constitué d'une séquence finie de gènes. Un chromosome est un élément de l'espace de recherche et représenté sous forme de chaînes.
Individu	Un individu est représenté par un seul chromosome.
Population	Une population est un groupe d'individus sur lequel l'algorithme génétique effectue un certain nombre d'opérations pour engendrer la nouvelle population.
Génération	Une génération est l'ensemble des opérations qui permettent de passer d'une population à une autre.
Fonction objectif	Les fonctions objectifs sont utilisées en optimisation mathématique pour guider un système à un état optimal et refléter la qualité d'une solution au problème.

La terminologie utilisée dans les AG est empruntée à la génétique : les chromosomes sont les éléments dans lesquels les solutions sont développées (individus).

Ces chromosomes sont rassemblés en population et la combinaison des chromosomes est le stade de reproduction. Celle-ci se réalise à l'aide d'un opérateur de croisement et/ou un opérateur de mutation. D'autres notions sont propres au domaine des AG tel que la faculté d'adaptation par le fitness, également appelé indice de performance, qui est une mesure permettant de classer les chromosomes, par une formule théorique qui permettent le calcul l'indice de qualité de ces chromosomes.

La Figure 2.1 présente le fonctionnement itératif simplifié d'un AG. Tout d'abord, une population initiale P(0) est construite, Cette population choisie aléatoirement constitue la population initiale. Ensuite, Chaque individu x de la population représentée comme une forme génétique du problème, itérativement, après avoir effectué un calcul de la fitness f(x) de chacun des individus, l'algorithme procède à une sélection des parents pour la reproduction par le croisement et la mutation, puis à un remplacement. Ce processus est répété jusqu'à ce qu'un critère d'arrêt soit satisfait. Le remplacement sert à gérer l'insertion des nouveaux individus générés, appelés enfants, avec traitement de la population, c'est donc du processus de passage de la population P(t) à la population P(t+1).

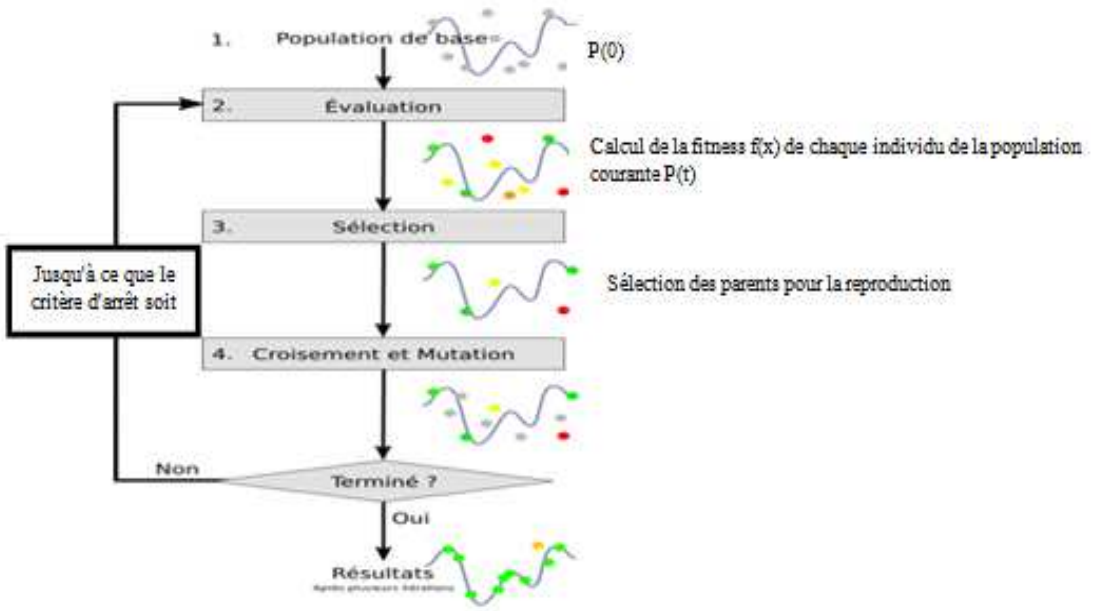


Figure 2.1. Fonctionnement d'un AG de base.

Ce qui suit une description non exhaustive des principales composantes d'un AG : La représentation, le fitness, la population initiale, les opérateurs génétiques et le critère d'arrêt. Cette séquence d'opérations forme le cycle de base des algorithmes génétiques présenté dans la Figure 2.1, et chaque nouvelle population est dite génération de solutions. La procédure se répète jusqu'à satisfaire un critère d'arrêt. Le principe des approches génétique est illustré par l'algorithme 2.1 **[Meigman 2004].**

Algorithme 2.1 : Principe des algorithmes génétiques

Générer la population initial P
Evaluer les individus de P
Tant que le critère d'arrêt n'est pas atteint **faire**
  Croiser les individus de P pour obtenir P'
  Muter les individus de P'
  Evaluer les individus de P'
  Sélectionner la nouvelle génération P à partir de P et P'
Fin

2.2.4.1.1 Représentation

À l'aide de la représentation, toutes les solutions possibles du problème doivent pouvoir être codifiées c'est-à-dire que cette représentation doit être complète. De plus, toutes les solutions codifiables doivent être compatibles à des solutions réalisables. Historiquement, le codage utilisé par les algorithmes génétiques est représenté dans des séquences de bits qui contiennent toutes les informations nécessaires pour décrire un point dans l'espace de recherche. Ce type de codage est utile pour créer des opérateurs de croisement et de mutation

simples. La structure du problème est préservée dans le codage. À la Figure 2.2, nous observons trois représentations différentes : binaire pour Chromosome a, ordinale pour Chromosome b et alphabétique pour Chromosome c. Il est possible de retrouver d'autres représentations dans la littérature **[Globerg 1989].**

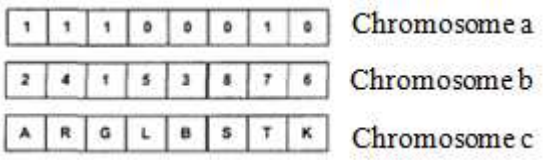


Figure 2.2. Exemple de représentation

2.2.4.1.2 Fitness

Une fonction à optimiser, appelée fitness ou fonction d'évaluation de l'individu. Détermine la qualité de chaque chromosome par rapport au problème, il est couramment utilisé pour sélectionner les chromosomes pour la reproduction. Les chromosomes sont de bonne qualité et ont alors plus de chances d'être choisis pour la reproduction, veille à ce que la prochaine génération de la population hérite de leurs caractéristiques génétiques.

2.2.4.1.3 Population initiale

La population initiale d'un AG est la population à démarrer de l'algorithme. Elle doit contenir des chromosomes qui sont distribuer dans les solutions spatiales AG pour fournir une variété du matériel génétique**,** ce mécanisme doit être capable de produire une population d'individus non homogène servant de base pour les générations futures. le choix de la population initiale est important car il affecte la vitesse de la convergence vers l'optimum global; Si la position de l'optimum dans l'espace de solutions est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace de solutions et en veillant à ce que les individus produits respectent les contraintes pour demeurer dans cet espace **[Michalewicz 1996],** dans une situation où il y a peu des informations sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur l'ensemble du champ de la recherche.

2.2.4.1.4 Sélection

L'opérateur de sélection tend à augmenter l'importance de l'utilisation des solutions de bonne qualité pour la génération de descendants**.** La sélection joue un rôle très important dans les algorithmes génétiques, d'une part et pour diriger les recherches vers les meilleurs individus d'autre part, pour maintenir la diversité des individus dans la population. Plusieurs stratégies

de sélection ont été mises en œuvre et les paragraphes suivants décrivent deux des plus répandues qui sont la sélection par la fitness et la sélection par tournoi.

- **Sélection par fitness**

  La sélection par la fitness ou sélection par proportionnalité, est en général implémentée comme une roulette biaisée introduite par **[Globerg 1989]**. Elle consiste à relier à chaque individu un segment (ou case de la roue) qui est proportionnelle à son fitness comme l'illustre la Figure 2.3. Au démarrage de la roue, l'individu sélectionné est celui qui à arrêter la roue. La roue étant lancée, l'individu sélectionné est celui sur lequel la roue s'est arrêtée.

  Cette méthode favorise les meilleurs individus, mais tous les individus ont encore conservé une chance d'être sélectionnés.

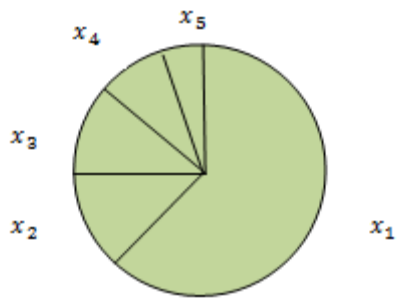


Figure 2.3. Roulette pour une population de 5 individus avec $f(x_i)=\{70,15,10,10,5\}$.

- **Sélection par tournoi**

  Un groupe de q individus sont choisir aléatoirement dans la population de P individus, où q est un paramètre appelé taille du tournoi. Il y a une sélection par tournoi déterministe ou probabiliste. Dans le cas du tournoi déterministe, le meilleur des q individus gagne le tournoi. Dans le cas probabiliste, chaque individu peut être choisi comme vainqueur avec une probabilité proportionnelle à sa fonction d'évaluation. Il est tout à fait possible que certains individus participent à plusieurs tournois. L'avantage de cette méthode de sélection est qu'elle n'est pas coûteuse à exécuter.

2.2.4.1.5 Opérateur de croisement

Généralement, les croisements sont faits avec deux parents et engendrer deux enfants mais il est tout à fait possible d'imaginer des croisements avec x parents pour produire y enfants qui vont hériter certaines caractéristiques de leurs parents. Un croisement consiste à échanger aléatoirement les gènes des parents afin de donner des enfants qui comportent des propriétés combinées.

Son rôle fondamental est de permettre la recombinaison des informations présentes dans les gènes de la population. L'objectif principal du croisement est d'enrichir la diversité de la population en manipulant la structure des chromosomes**.**

Il y a plusieurs types de croisements, nous pouvons citer, par exemple, le croisement peut être simple à un point, les croisements à deux points de coupure, les croisements à multipoint de coupure, les croisements ordinaux, les croisements barycentriques, les croisements à chemin, etc. Dans notre travail, nous allons utiliser le croisement à un point qui est une des premières stratégies de croisement utilisées par les AG **[Holland 1975].**

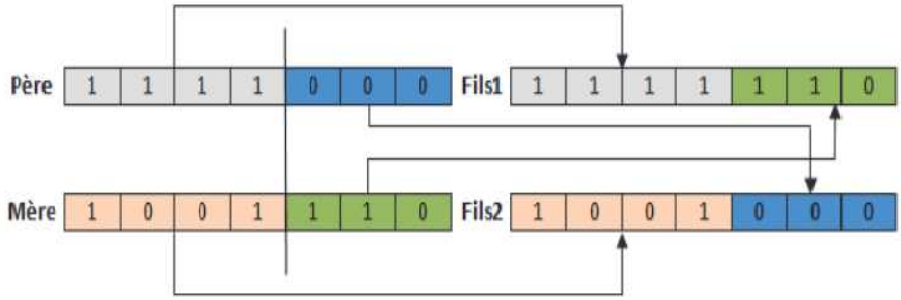


Figure 2.4. Illustration du croisement à un point.

À la Figure 2.4, le point de coupure entre les deux parents se situe à la position 4. À partir du père, le premier enfant est obtenu en copiant les gènes de la mère à droite du point de coupure. Le deuxième enfant est obtenu à partir de la partie droite du point de coupure des gènes du père collé à la partie gauche des gènes de la mère.

2.2.4.1.6 Opérateur de mutation

Les propriétés de convergence des algorithmes génétiques sont fortement dépendantes de l'opérateur de mutation sur le plan théorique donc il joue un rôle de bruit et tente d'empêcher une convergence trop hâtive. C'est une modification légère dans le chromosome par un remplacement de façon aléatoire d'un gène au sein d'un chromosome par un autre.

L'objectif de cet opérateur est de diversifier la population afin d'examiner d'autres parties de l'espace de recherche et d'éviter la convergence trop rapide de l'algorithme vers un minimum local. De nombreux opérateurs de mutation ont été établis, parmi lesquels l'inversion et l'échange.

- **Mutation par inversion :**

  Utilisée habituellement en codage binaire. Le quatrième gène est sélectionné et inversé. À la Figure 2.5, la position 4 est sélectionnée et le gène est inversé, il passe de 1 à 0.

Figure 2.5. Illustration de la mutation par inversion

- **Mutation par échange :**

  Échanger les positions respectives des deux éléments choisis. À la Figure2.6, les gènes aux positions 5 et 7 sont échangés.

Figure 2.6. Illustration de la mutation par échange

2.2.4.1.7 Opérateur de remplacement

Souvent, les meilleures solutions remplacent les plus mauvaises et il en résulte une amélioration de la population. Pour former la nouvelle population parente, deux stratégies peuvent être adoptées :

- **La première stratégie** : l'approche(μ, λ), consiste à favoriser les μ meilleurs individus parmi les λ enfants.
- **La deuxième technique** : les μ meilleurs individus dans les populations parents et enfants combinées sont gardés.

2.2.4.1.8 Critères d'arrêt

L'objectif des AG est de produire des solutions variées et de qualité. Nous pouvons trouver différents critères d'arrêt utiliser par l'AG comme le nombre déterminé de générations, le temps, une valeur particulière de la fonction objectif, un nombre fixé d'évaluations etc, son rôle c'est de juger la qualité des individus.

Généralement les AG nous donnent des bons résultats et des solutions de bonne qualité, mais souvent pour les rendre plus efficace en termes de qualité de solutions obtenues, on lui réunit des procédures de recherche locale.

2.2.4.2 Algorithmes mémétiques

Les algorithmes mémétiques sont introduits pour la première fois par Moscato **[Moscato 1989]**. Une combinaison entre une méthode à population (algorithme évolutionnaires) et la recherche locale, cela nous donne les algorithmes mémétiques. Cette combinaison, communément appelée hybridation, représente une avenue de recherche très prometteuse qui nous donne un potentiel de diversification fourni par l'approche à population et la capacité d'intensification offerte par la recherche locale. Ceci laisse à penser que l'intégration de processus d'intensification et de diversification permettrait d'améliorer encore davantage les

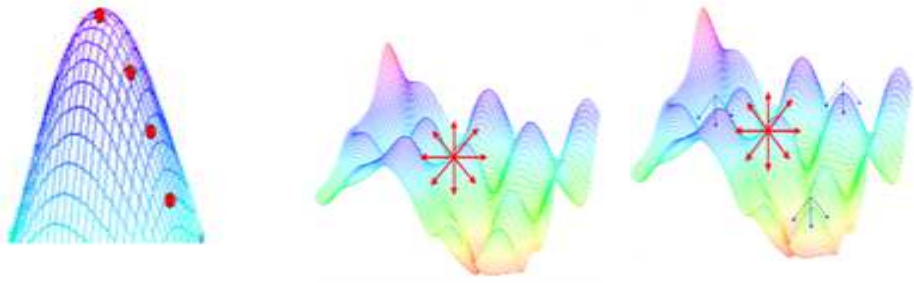
performances. On peut considérer un algorithme mémétique comme un algorithme génétique étendu grâce à la recherche locale.

En général, l'hybridation consiste à combiner deux ou plusieurs méthodes de résolution de problèmes d'optimisation combinatoire en un unique algorithme. Les approches hybrides, en particulier les métaheuristiques hybrides, gagnent maintenant en popularité car ce type d'algorithme produit généralement les meilleurs résultats pour plusieurs problèmes d'optimisation combinatoire **[Talbi 2009].**

Les algorithmes génétiques peuvent être une bonne approche pour résoudre des problèmes d'optimisation combinatoire mais un inconvénient de cette méthode est que les opérateurs standard de croisement et de mutation ne permettent pas d'intensifier suffisamment la recherche comme le rappellent **[Hoos et al 2004].** Tout comme pour un algorithme génétique, dans un espace de solutions on cherche de trouver les éléments de cet espace ayant la meilleure adaptation à l'environnement posé. L'objectif d'un AG est de faire évoluer cette population afin de trouver la meilleure solution. Dans ce but, à chaque génération, les individus de la population sont sélectionnés ainsi la reproduction favorise les individus les plus sains. Lors de reproduction il y a deux phénomènes qui la caractérisent, l'opérateur des croisements sont envisagés avec deux parents et génèrent des enfants, il consiste à échanger les gènes des parents afin de donner des enfants qui comportent des propriétés combinées, le but de ce phénomène est d'enrichir la diversité de la population en manipulant la structure des chromosomes. Ensuite, des modifications des informations contenues dans le chromosome peuvent survenir, ces modifications peuvent avoir des conséquences soit en améliorant soit en le dégradant, c'est ce que l'on appelle la mutation. Cependant cette opération est remplacée par un processus de recherche locale son but est d'améliorer la qualité des combinaisons générées. En comparaison avec l'opérateur de mutation, la recherche locale joue spécialement le rôle de l'intensification de la recherche.

Les algorithmes génétiques ont été appliqués avec succès à une variété de problèmes combinatoire. Malgré leurs capacités méthodiques, un inconvénient d'un algorithme génétique est que les opérateurs standard de croisement et de mutation ne permettent pas d'intensifier suffisamment la recherche et obtenir une convergence efficace **[Hoos et al 2004].** Le rôle de mutation est de favoriser la diversification des individus à partir de légère modifications à l'individu qui apporte. C'est pourquoi on hybride les algorithmes génétiques avec des méthodes de recherche locale. On peut explorer rapidement les zones intéressantes de l'espace de recherche pour les exploiter à partir de deux méthodes :

1. La mutation détecte de bonnes régions dans l'espace de recherche



a) mutation b) Recherche locale c) Algorithme mémétique

Figure 2.7. Comportement schématique d'une mutation, d'une recherche locale et d'un algorithme mémétique**.**

2. La recherche locale intensive à explorer ces zones de l'espace de recherche donc Ces deux méthodes sont complémentaires **[Holland 1975]**.

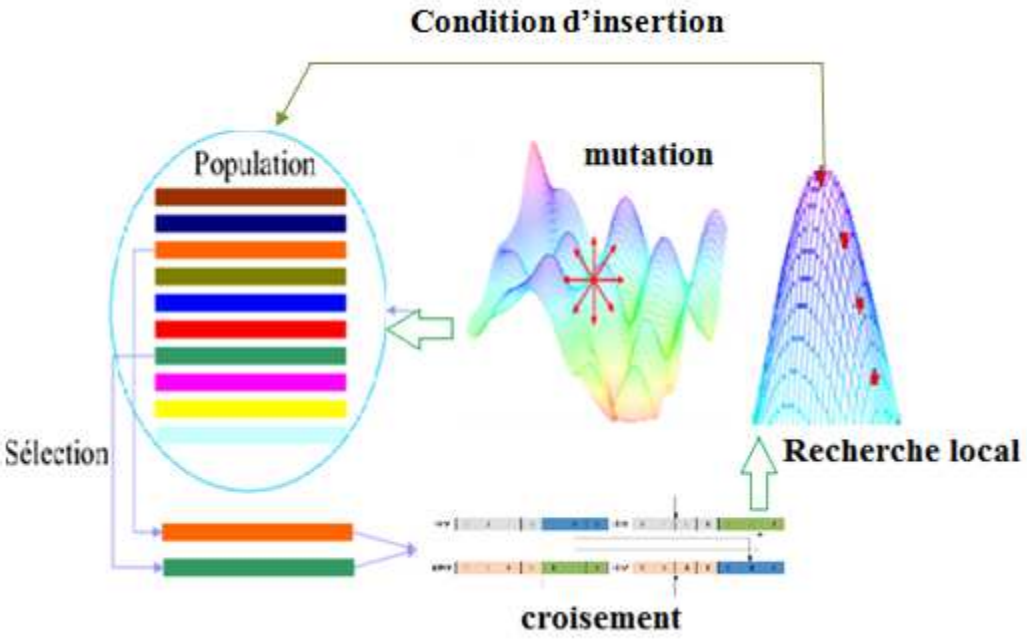


Figure 2.8. Schéma des algorithmes mémétiques

2.2.4.2.1 Recherche locale

la méthode de recherche ou bien l'algorithme de recherche locale est la méthode qui converge vers un optimum local, ce sont des méthodes itératives qui se bassent sur une solution d'algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [**Moscato 1993**]. Les méthodes de recherche locale ou métaheuristiques à base de voisinages s'appuient toutes sur un même principe. A partir d'une solution unique x_0, considérée comme point de départ (et calculée par exemple par une heuristique constructive), la recherche consiste à passer d'une

solution à une solution voisine par déplacements successifs. L'ensemble des solutions que l'on peut atteindre à partir d'une solution *x* est appelé *voisinage N(x)* de cette solution. Déterminer une solution voisine de *x* dépend bien entendu du problème traité **[Sevaux 2004]**.

En générale, les opérateurs de recherche locale s'arrêtent lorsque la meilleure solution est trouvée localement, c'est à dire quand il n'existe pas de meilleure solution dans le voisinage. Nous trouvons dans la littérature de nombreuses méthodes locales. Les plus anciennes et les plus utilisées sont : la méthode de descente, la recherche tabou, le recuit simulé,…etc. Dans un algorithme mémétique, on peut utiliser une méthode de recherche locale simple telle que les méthodes de descente car c'est l'une des heuristiques de recherche locale les plus simples et on peut très facilement implémenter. Cette méthode s'articule tout autour d'un principe simple :

- L'Algorithme suivant présente le squelette d'une méthode de descente (simple descente). A partir d'une solution initiale *x*, on choisit une solution x_0 dans le voisinage *N(x)* de *x*. Si cette solution est meilleure que *x*, ($f(x_0) < f(x)$) alors on accepte cette solution comme nouvelle solution *x* et on recommence le processus jusqu'à ce qu'il n'y ait plus aucune solution améliorante dans le voisinage de *x*.

Algorithme 2.2: La méthode de descente

1. initialisation : trouver une solution initiale x
2. Répéter
3. Recherche dans le voisinage : trouver une solution $x' \in N(x)$
4. Si $f(x') < f(x)$ alors
5. $x \leftarrow x'$
6. Fin si
7. Jusqu'à $f(y) \geq f(x), y \in N(x)$

Une nouvelle génération d'algotrithme mémétique proposée trés récemment par Sörensen et Sevaux (2003). Il s'agit d'une forme enrichie d'une gestion de la population de solutions. Cette version s'appelle (MA | PM) pour Memetic Algorithm with Population Management. Une mesure de distance est utilisé afin d'apporter une certaine diversité dans les chromosomes-parents.

2.2.4.3 Présentation du MA | PM

La principale caractéristique du MA/PM vient de la gestion de la population grâce à une mesure de distance dans l'espace des solutions. Cette opération permet de contrôler la diversité des individus en filtrant l'entrée des enfants dans la population. Ce principe étant appliqué dans la suite au problème d'ordonnancement d'un système flexible de production.

Nous appelons Δ le paramètre de la diversité. En supposant que la qualité de s_k est suffisante, une solution peut être ajoutée à la population si les conditions suivantes sont vérifiées :

$$d_p(s_k) = \min_{s_i \in p}(s_k, s_i) \geq \Delta \quad (2.1)$$

En utilisant la distance $d_p(s_k)$ et la valeur da la fitness (la fonction objectif) $f(s_k)$. Un moyen très simple de faire ceci (en supposant que *f* doit être minimisée) est de calculer $(s_k + \lambda d_p(s_k))$. Si cette valeur ne dépasse pas un certain seuil, la solution est ajoutée, sinon elle est rejetée. λ est un paramètre qui détermine l'importance relative de la diversité par rapport à la qualité de la solution. Si la procédure de recherche locale est suffisamment efficace pour assurer en permanence la qualité des solutions qu'elle produit, une solution peut être ajoutée si l'équation (2) est vérifiée sans prendre la valeur de la fonction objective de la solution en compte **[Sörensen et al 2004].**

La principale caractéristique du MA/PM est basé sur un algorithme génétique mais se différencie des versions classiques par deux principaux éléments :

- La méthode de recherche locale utilisée pour une amélioration des solutions.
- Une technique de gestion de la population par une mesure de distance $d_p(S)$ a une petite population *P* de solutions de bonne qualité (Population Management - PM).

Pour obtenir une métaheuristique plus performante il faut maintenir un équilibre entre l'intensification et la diversification par la gestion de la population PM, cette procédure pouvait contrôler l'équilibre entre ces deux stratégies. Sörensen et Sevaux proposent deux options : soit l'enfant est muté jusqu'à ce que sa distance à la population atteigne au moins le seuil Δ, soit il est tout simplement rejeté. Dans le premier cas, plusieurs mutations successives peuvent être nécessaires et le temps passé pour obtenir un enfant admissible dans la population peut être parfois important.

Le PM signifie qu'une nouvelle solution *s* ne peut intégrer la population courante que si sa distance $d_p(S)$ à la population courante *P* est telle que $d_p(S) \geq \Delta$, avec Δ un seuil donné.
Si Δ=0, l'algorithme se comporte comme un classique MA (sans gestion de la population). Δ≥1 garantit que chaque solution de la population P est distincte des autres (la distance définie est entière). Par contre, si Δ est fixé à une grande valeur, alors la majorité des solutions enfants est rejetée et l'algorithme passe beaucoup de temps dans des itérations improductives. La valeur de Δ peut être alors ajustée dynamiquement entre ces deux extrêmes afin de contrôler la diversité de la population **[Prodhon 2006].**

Sörensen et Sevaux [**Sörensen et al 2004]** proposent différentes politiques de contrôle pour Δ comme la montre la figure 2.9 :

1) Δ constant : garantit un niveau constant de diversification.
2) Initialiser Δ avec une grande valeur, puis décroître doucement au fur des itérations, ceci favorise une intensification en fin d'algorithme.
3) Initialiser Δ avec faible valeur, puis l'augmenter au fil des itérations, ceci permet une diversification en fin d'algorithme.
4) Initialiser Δ avec une grande valeur au début de l'algorithme, puis décroître doucement tant que des solutions améliorantes sont trouvées, sinon, après un certain nombre d'itérations sans amélioration, ré-augmenter la valeur de Δ afin d'introduire de la diversification dans la population : stratégie adaptative ajustant dynamiquement la valeur de Δ pour contrôler la diversité de population.

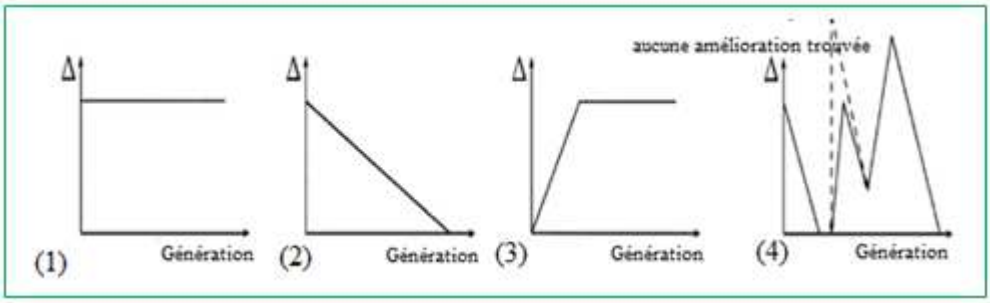


Figure 2.9. Les différentes stratégies de la gestion de population. (1), (2), (3) et (4).

- **Algorithme général**

L'algorithme suivant présente un squelette du **(MA/PM) [Prodhon 2006] :**

Algorithme 2.3: Algorithme mémétique avec gestion de population (MA/PM)

1. **Initialiser** : générer une population initiale P de solutions.
2. Fixer le paramètre de la diversité Δ
3. **Répéter**
4. Sélection : choisir deux solutions p_1 et p_2
5. Croisement : combiner deux solutions parents p_1 et p_2 pour former des Solutions c_1 ,
6. Recherche locale : appliquer une procédure de recherche locale sur c_1, c_2
7. **Pour** tout enfant c faire
8. **tant que** c ne satisfait pas la condition d'addition ($d_p(c) < \Delta$) Faire
9. Mutation(c)
10. **Fin tant que**
11. Supprimer la solution de la population
12. Addition : de solution :$\boldsymbol{p} \leftarrow \boldsymbol{p} \cup \boldsymbol{c}$
13. **Fin pour**
14. Mettre à jour le paramètre de diversité Δ
15. **Jusqu'à** satisfaire un critère d'arrêt.

2.2.4.4 Algorithme de la recherche dispersée

Cette méthode a été proposé par Glover **[Glover 1995]**, La recherche dispersée est une métaheuristique qui est basée sur l'évolution de la population. Durant la recherche, elle capture l'information dans les sous-ensembles pour construire de nouvelles solutions, et elle conserve les deux grandeurs (diversité, qualité) de l'ensemble de référence. La recherche dispersée utilise des stratégies pour diversifier et intensifier la recherche et elle a prouvé son efficacité dans des variétés de problèmes d'optimisation.

La recherche dispersée se décompose en deux phases importantes qui sont : la phase d'initialisation et la phase d'évolution. Les étapes de ces deux phases sont données comme suit **[Benatchba 2005]** :

2.2.4.4.1 Phase d'initialisation

Etape 1 : Générer les solutions de départ : Génère aléatoirement une ou plusieurs solutions d'essai de départ qui seront utilisées pour initialiser le reste de la méthode de recherche.

Etape 2 : Génération des solutions diversifiées : Utiliser la méthode de génération de diversification pour générer des solutions dispersées à partir de la solution de départ.

Etape 3 : Construction de l'ensemble de référence : Pour chaque solution d'essai générée dans l'étape 2, utiliser la méthode d'amélioration pour créer un ou plusieurs solutions plus améliorées. Pendant l'application successive de cette étape, maintenir et faire la mise à jour de l'ensemble de référence qui est constitué par les *b* meilleures solutions trouvées.

Etape 4 : Répéter l'exécution de l'étape 2 et 3 jusqu'à l'obtention d'un nombre voulu de solutions qui constitueront la population initiale de l'ensemble de référence.

Phase d'évolution :

Etape 5 : La génération des sous-ensembles : Générer des sous-ensembles de la population de l'ensemble de référence à partir des quels seront construites les solutions combinées.

Etape 6 : Combinaison des solutions : pour chaque sous ensemble *X* produit à l'étape 5 on applique une technique de combinaison des solutions pour produire l'ensemble *C(X)* qui consiste en une ou plusieurs solutions combinées. Puis traiter chaque élément de *C(X)* individuellement dans l'étape suivante.

Etape 7 : Amélioration et mise à jour de l'ensemble de référence : pour chaque solution produite à l'étape 6 appliquer la méthode d'amélioration pour avoir une (ou plusieurs) solution(s) de qualité supérieure et faire la mise à jour de l'ensemble de référence.

Etape 8 : Si aucune solution n'est produite à l'étape 6, alors on réinitialise la population avec l'ensemble choisit et on reprend à partir de l'étape 2.

Etape 9 : Répéter l'exécution des étapes de 5 à 8 jusqu'à la rencontre d'un critère d'arrêt qui est le plus souvent un nombre maximum d'itérations atteint.

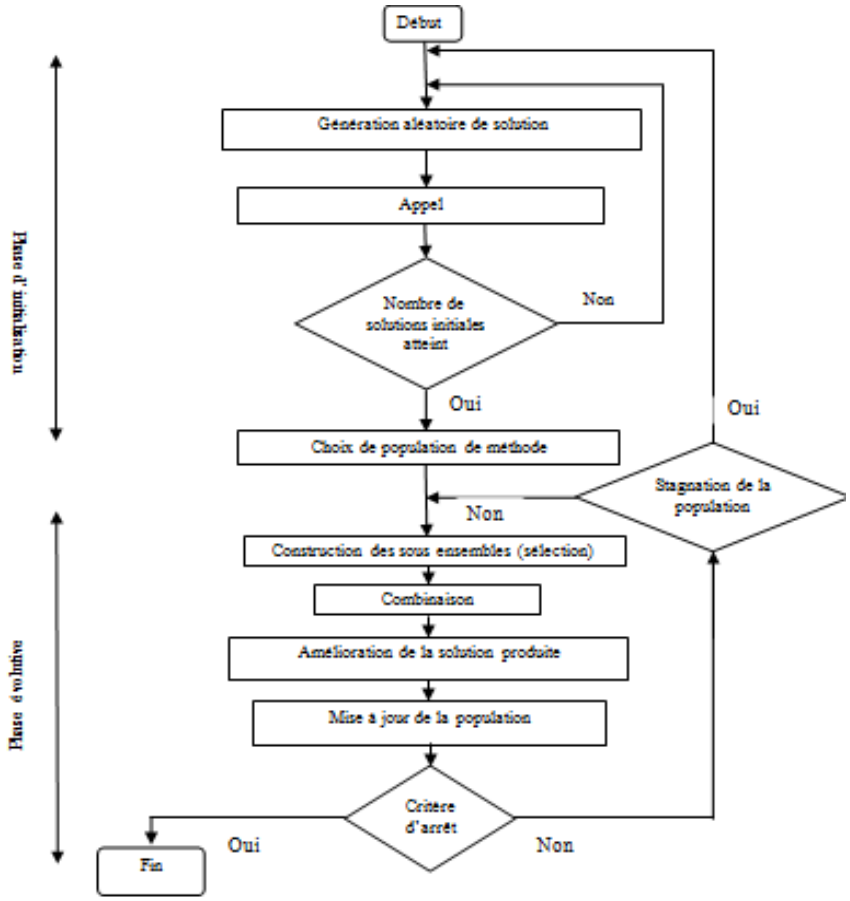


Figure 2.10. Etapes de la recherche dispersée

2.2.4.4.2 Gestion de la population et le générateur de diversification

La recherche dispersée gère sa population différemment des algorithmes génétiques. Dans la phase d'initialisation, elle démarre d'une population variée P de solutions et elle génère des solutions aléatoires. Typiquement, la taille de la population initiale est dix fois plus grande que la taille de l'ensemble de référence.

Ensuite, un ensemble de référence b de solutions est choisie parmi la population courante pour participer à la phase de reproduction et produire d'autres solutions, pour chaque élément de cette population, le degré d'adaptation (fonction objectif) est calculé, ce qui permet d'extraire deux sous-ensembles.

L'ensemble de référence de la recherche dispersée b contient, d'une part, un certain ensemble élite (de taille b_1) de bonnes solutions choisies selon leur fonction objectif. D'autre part, un certain ensemble diversifié (de taille b_2) de solutions qui sont tirées de la population restante $P-b$ et rajoutées à la collection b pour la compléter. Les b_2 solutions sont les solutions les plus éloignées des meilleures solutions de b, elles sont appelées : solutions diversifiées.

Après avoir choisi un ensemble de solutions bonnes et diversifiées, la phase de reproduction est lancée.

La distance entre un individu et une population peut être définie comme étant la distance minimale séparant cet individu de chaque individu de la population élite. Ce processus permet de maintenir une certaine diversité dans la population et évite de rester bloqué dans un optimum local. Il permet, par conséquent d'éviter une stagnation rapide de l'algorithme, ce qui constitue un des inconvénients majeurs de l'algorithme génétique **[Benatchba 2005]**.

Le deuxième ensemble regroupe les individus de la population les plus différents. Soit b_2 son cardinal. Le minimum des distances euclidiennes (Δ_{min}) entre les éléments de P et de l'ensemble de référence est calculé. Les éléments ayant les distances les plus élevées seront intégrés dans ce sous ensemble. Ces éléments sont éliminés de P au fur et à mesure.
L'ensemble de référence évolue dynamiquement durant la phase de combinaison des solutions. Une solution produite peut faire partie de l'ensemble de référence si :

- Sa fonction objectif est meilleure que celle de la moins bonne solution du deuxième sous ensemble. Cette solution remplace donc la moins bonne.
- Sa fonction objectif est plus « éloignée » que celle des éléments du premier sous ensemble. (Δ_{min} de cette solution est meilleure que la solution ayant le mauvais Δ_{min}). **[Cavique 2001], [Benalia 2005]**.

À partir de chaque solution elle génère d'autres solutions qui lui seront très distantes en faisant appel au générateur de diversification, et ce jusqu'à atteindre un nombre maximum de solutions initiales, le processus recommence jusqu'à satisfaction d'un critère d'arrêt.

- **Algorithme général**

L'algorithme suivant présente un squelette de la recherche dispersée **[Sevaux 2004] :**

Algorithme 2.4: La recherche dispersée

1. **Initialiser** : générer une population initiale P de solutions.
2. Mettre les meilleures solutions trouvées dans une population de référence **R.**
3. **Tant que** le critère d'arrêt non satisfait faire.
4. A=R → prendre la population de référence.
5. **Tant que** A≠∅ faire
6. Combiner les solutions (croisement)(B←R×A)
7. Améliorer les solutions B.
8. Mettre à jour les solutions R.
9. Garder les meilleures solutions à partir R ∪ B
10. A←B−R
11. **Fin tant que**
12. **Enlever** de R la mauvaise solution
13. Ajouter des nouvelles solutions diverses dans R
14. **Fin tant que**

2.3 Conclusion

Aprés la présentation des problèmes d'ordonnancement et de leurs principales caractéristiques, différentes méthodes, exactes et approchées, pouvent être utilisées pour la résolution de ces problèmes et ont été introduites dans ce chapitre.

Ce chapitre a décrit des généralités surl'intelligence artificielle et ses méthodes ainsi que les algorithmes évolutionnaires qui s'inspirent de la nature et du comportement social des individus. L'inconvénient de certaines méthodes de voisinage ou de construction est qu'elles tombent facilement dens un optomum local. La solution trouvée dépend fortement de la solution initiale. Ces methodes, en utilisant des stratégies intelligentes, arrivent à échapper aux minimas locaux.

La version du MA | PM et SS définie dans cette partie a fait l'objet d'une présentation dans des actes publiés dans la série « The Open Automation and Control Systems Journal »et « Electrotehnica, Electronica, Automatica Journal». Une des pistes de recherche pour l'améliorer pourrait être l'inclusion de techniques d'apprentissage permettant de guider les méthodes sur un bon sous-ensemble de dépôts, et ainsi de réduire le nombre d'itérations et donc les temps de calcul.

Chapitre 3

3 Investigations sur les métaheuristiques pour l'optimisation combinatoire

Ce chapitre présente une étude bibliographique et un état de l'art sur les méthodes de résolutions du problème d'ordonnancement job shop seront abordés, nous étudions ensuite les problèmes d'optimisation combinatoire de manière générale. Nous présentons les notions principales des Métaheuristiques et leur classification ensuite, nous citons les différentes algorithmes évolutionnaires (AE). Puis nous parlons d'une analyse du comportement de l'intensification et de la diversification pour des variantes d'algorithmes génétiques et GA. Nous proposons une comparaissent entre ces variantes. Enfin nous terminons par une conclusion

Sommaire

3.1 Introduction

Au cours des dernières années, les chercheurs dans le domaine de la recherche opérationnelle se sont intéressés aux systèmes de production présentant une flexibilité sur les ressources telles que les systèmes flexibles de production (FMS). L'intérêt porté à ces systèmes est largement justifié par ce caractère de flexibilité donnant plus de degrés de productivité et s'approchant de plus en plus des problèmes d'ateliers réels. Ils fournissent une augmentation d'utilisation des ressources, la réduction des encours…etc. Le FMS est un système qui contient plusieurs ou une cellule de machine hautement automatisée composée d'un groupe de postes de travail de traitement, chaque machine, équipée d'un magasin d'outils, peut effectuer une variété d'opérations, Interconnecté par un système de transport intégré **[Chunwel et al 2001].**

Le système devrait être flexible, productif et devrait être capable de répondre aux demandes en un temps borné à un coût raisonnable. Un FMS peut être considéré comme flexible s'il est capable à rendre flexible l'ensemble des outils de la production. C'est-à-dire, préparer une technologie à s'adapter aux divers changements de son environnement, sans qu'il y ait besoin d'engager de nouveaux investissements en équipement ou d'engendrer de longues pertes de temps. Donc flexibilité est le moyen de rester en tête dans les affaires. Dans de très nombreux articles, on remarque très souvent que la notion de flexibilité est abordée de manière différente. Les travaux de [**Sethi et al 1990]** et de [**Browne et al 2016**] permettent de définir onze niveaux de flexibilités. On peut citer : la flexibilité des machines, la flexibilité des opérations, la flexibilité des produits, la flexibilité du cheminement des produits, la flexibilité des volumes de produits, la flexibilité de la production et la flexibilité face au marché ect.

Pour profiter des avantages du FMS pour la production, un bon système d'ordonnancement est nécessaire. L'ordonnancement détermine ce qui va être fait, quand, où et avec quelles ressources. Une méthode d'ordonnancement souhaitable doit inclure deux caractéristiques : une formulation facile du problème et une identification rapide des solutions semi-optimales **[Kim 2007].** Plusieurs approches de résolution ont été proposées pour résoudre les problèmes d'ordonnancement FMS mais il n'y a pas de solution efficace pour tous les problèmes due principalement à la complexité de l'ordonnancement des FMS **[Joseph et al 2011**]. Les problèmes d'ordonnancement peuvent être considérés comme plus complexes dans les FMS que dans les systèmes de fabrication traditionnels. La raison non seulement est de spécifier la séquence des jobs, mais aussi de déterminer quelle tâche sera exploité à quelle machine est également crucial pour le problème d'ordonnancement FMS. Pour cette raison, les chercheurs ont utilisé différentes approches pour résoudre ce problème, comme la

programmation mathématique, la simulation, les algorithmes et les heuristiques ... **[Harun et al. 2009].** Le choix de la méthode de résolution dépend fortement de la taille et de la complexité du problème. Si le problème est de petite taille et de complexité réduite, une méthode exacte peut suffire et aboutir à une solution optimale. Dans le cas où le problème est de grande taille ou de complexité importante, les méthodes approchées, et en particuliers les métaheuristiques présentent une alternative intéressante.

De nombreuses études montrent que les métaheuristiques peuvent obtenir de bonnes solutions et nécessitent moins de temps de calcul que les méthodes exactes. L'ordonnancement des FMS est parmi les problèmes d'optimisation combinatoire le plus difficiles et intéressants qui sont souvent classés NP-Difficiles, ce qui justifie l'utilisation des méthodes approchées pour leur résolution. Plusieurs travaux s'intéressent aux problèmes d'ordonnancement des FMS ont été établis depuis 1980.

Il existe plusieurs métaheuristiques dans la littérature, mais elles sont classées en deux catégories, celles à solution unique qui partent d'une seule solution, dite solution initiale et tentent de l'améliorer au cours des itérations et celles à population de solutions, qui examinent un ensemble de solution à la fois. La combinaison de deux métaheuristiques (ou plus) peut avoir lieu, et donne, donc naissance à des métaheuristiques dites hybrides. Parmi les métaheuristiques les plus répondues, on peut citer le recuit simulé et la recherche tabou dans la famille des méthodes à solution unique, les algorithmes génétiques, les colonies de fourmis et les essaims particulaires dans celle à population de solution. Il existe bien d'autres métaheuristiques telles que la recherche dispersée, l'électromagnétisme, la recherche harmonique, et celle basée sur les essaims d'abeilles. Toutes les métaheuristiques s'appuient sur un équilibre entre l'intensification et diversification de la recherche, une bonne métaheuristique est celle qui peut bien équilibrer entre ces deux stratégies, pour ne pas tomber dans des optimums locaux et ne pas avoir une exploration trop longue.

Les algorithmes évolutionnaires et en particulier les algorithmes génétiques ont connu un grand succès depuis leur développement par Holland **[Holland 1975].** Ces algorithmes sont basés sur les mécanismes de sélection, croisement et de mutations inspirées de la reproduction naturelle. En raison de leur simplicité et de leur facilité d'implémentation, ils ont été utilisés pour la résolution de plusieurs problèmes d'optimisation NP-difficile, en particulier ceux d'ordonnancement **([Petrovic 2005], [Pei et al 2008], [Pezzella 2008], [Shu- chen et al 2009], [Gao et al 2009]).** Malgré que les résultats obtenus par les algorithmes génétiques soient en

général satisfaisants, dans certains problèmes on ne se contente pas d'aboutir seulement à la solution satisfaisante mais il devient, impératif de trouver une solution de bonne qualité.

La puissance des algorithmes génétiques réside dans leur capacité de balayer l'espace des solutions et donc de détecter les zones de l'espace prometteuse, mais en contrepartie, ils ne permettent pas d'intensifier suffisamment la recherche. Partant de ce point, plusieurs variantes des algorithmes génétiques sont proposées. Les algorithmes mémétiques proposés par Moscato **[Moscato 1989],** qui combine un algorithme génétique avec une procédure de recherche locale, ayant pour but de compenser la faiblesse de l'algorithme génétique. Ces techniques aussi ont trouvés leur application pour la résolution des problèmes d'ordonnancement **([Hasan et al 2008], [Tavakkoli et al 2008], [Qian et al 2008])**

Les usines de production modernes sont obligées de mettre en place des systèmes capables de fournir la flexibilité et l'efficacité car il existe une tendance croissante vers une plus grande diversité de produits, de petites tailles de lots et des délais de production plus courts. En raison de la flexibilité de chaque machine, chaque processus de pièce peut être traité sur plusieurs machines et le temps de traitement peut varier légèrement sur ces différentes machines. En outre, le traitement des pièces peut être complété par l'un de plusieurs processus de séquence différents. Cela signifie qu'une opération alternative de routage est autorisée **[chunwel et al 2001].** La flexibilité de routage peut être considérée comme le principal facteur contribuant à la flexibilité d'un FMS. C'est la capacité de traiter une pièce par des chemins différents en utilisant des machines alternatives pour produire un ensemble de pièces économiquement et efficacement **[Joseph 2017].** Améliorer la performance d'un FMS en utilisant la flexibilité de routage dans l'ordonnancement est le facteur de motivation pour cette partie de travail. Une grande quantité de littérature concernant les routages alternatifs et l'ordonnancement dynamique est disponible **[Min-chun et al 2008].** Grinsted [**Grinsted 1995**] a discuté des avantages de l'utilisation des routages alternatifs dans un atelier et a proposé quatre règles de décision qui pourraient être utiles. Singh et Mohanty [**Singh et al 1991]** ont proposé une approche floue multi objective pour résoudre le problème de routage dans l'environnement de job shop. Chang et al. [**Chang et al 1986**] ont utilisé une simulation pour étudier un système comportant huit machines et huit types de pièces. Ils ont trouvé que la flexibilité du routage améliorait les performances du système, en particulier lorsqu'il était associé à une technique de répartition anticipée qui envoyait la pièce à la machine avec le moins de charge possible. Warren Liao [**Warren Liao 1994**] a proposé un modèle linéaire pour programmer un nombre entier afin de déterminer le routage optimal qui minimise le coût de production en tenant compte

du volume de production et de la capacité des machines. Rahul Capih et Subhash Wadwa **[Capih et al 1997]** ont étudié l'effet de certains paramètres nécessaires de conception et de contrôle de la performance hypothétique d'un FMS, ils ont présenté un cadre basé sur un modèle expérimental 'Taguchi' pour étudier la nature de l'impact de différents niveaux de flexibilité de routage sur la performance d'un FMS. Ro et Kim [**Kim et Roand 1991]** ont utilisé des techniques de prise de décision à critères multiples pour tester d'autres règles de répartition dans un FMS avec une flexibilité de routage dans la littérature existante sur le routage dynamique des pièces, Caprihan et Wadhwa **[Caprihan et al 1997]** ont montré l'effet de la variation des niveaux de flexibilité de routage (le nombre de routages alternatifs pris par un type de pièce) sur les performances d'un FMS jugé par le makespan. Caprihan et Kumar **[Caprihan et al 2006]** ont proposé une stratégie d'ordonnancement prenant en compte la flexibilité des routages basés sur la logique floue dans un FMS. Chunwi et Zheng Wu **[Chunwi et al 2001]** ont proposé un algorithme génétique (GA) pour résoudre le problème de séquençage de job d'un atelier de production caractérisé par un routage flexible et des machines flexibles. Ils ont essayé de gérer des situations similaires à l'environnement FMS réel notamment les exigences de sélection de routage, de sélection de machine, de sélection de séquence d'opérations et d'ordonnancement dynamique. Ümit Bilge et al. **[Ümit et al 2008]** ont présenté une étude relative à la flexibilité du routage. Cette étude a présenté trois nouvelles approches, y compris une approche de logique floue, pour le routage dynamique des pièces. An-Yuan Chang **[Chang 2007]** a présenté une recherche pour mesurer la flexibilité du routage, il a proposé, entre autres, des modèles mathématiques afin d'évaluer l'efficacité des routages, la polyvalence ainsi que la variété de ces routages. Rajamani et al. **[Rajamani et al 1990]** ont proposé trois modèles de programmation de nombres entiers pour étudier l'effet des routages alternatifs des pièces en tenant compte de l'utilisation des ressources dont l'objectif est de minimiser le cout. Andrea Rossi et Gino Dini **[Rossi et al 2007]** ont proposé un système logiciel basé sur l'optimisation des colonies de fourmis pour résoudre l'ordonnancement des FMS dans un environnement de job shop avec flexibilité de routage et une configuration dépendant de la séquence et du temps de transport. Min-Chun Yu et Timothy J. Greene **[Min-Chun et al 2008]** ont présenté le contexte et la raison pour mesurer la flexibilité de routage pour un système flow shop, une mesure opérationnelle de flexibilité de routage dans un système de production multiproduits multi-étapes. Garavelli [**Garavelli 2001]** a mené une étude de simulation sur la performance de plusieurs systèmes de fabrication flexibles, chacun d'eux étant caractérisé par un degré de flexibilité de routage spécifique. Il a constaté qu'au lieu d'une flexibilité totale, un système avec une flexibilité limitée fonctionnait mieux en termes de délais et de travaux en cours. El Mekkawy et El Maraghy **[El**

Mekkawy et al 2003] ont utilisé un routage flexible pour éviter la stagnation du système causée par les pannes et les temps d'arrêt de la machine. Shukla et Chen **[Shukla et al 2001]** ont proposé un cadre de système de décision pour aider de contrôler le FMS grâce à un lancement de partie intelligente. Le système proposé utilise une heuristique basée sur le concept d'attraction et un modèle de réseau neuronal. Haq et al**. [Haq et al 2003]** ont proposé un heuristique énumératif pour l'ordonnancement d'un FMS dans lequel le planning de production est intégré au planning MHS. Joseph et Sridharan **[Josephet al 2011]** ont étudié les effets de la flexibilité du routage, de la flexibilité du séquençage et de l'ordonnancement des règles de décision sur les performances d'un système de production flexible.

On retrouve également le travail de Mahmoodi et Mosier **[Mahmoodi et al 1999]** dans lequel il étudie l'effet des règles de priorité et des niveaux de flexibilité de routage sur les différentes performances des FMS. Nous retrouvons également le travail de Saygin, Kilick et Who qui ont proposé une plateforme intégrant la planification flexible et l'ordonnancement prédictive, ils ont présenté un concept appelé méthode de maximisation de la dissimilarité (DMM) pour minimiser la congestion dans un FMS. L'idée dans cette méthode est de maximiser les dissimilarités entre les routages occupés. Dans leur algorithme, chaque routage ne contient qu'une seule pièce en même temps. L'efficacité de cette règle dans la résolution des problèmes de sélection des routages a été prouvée dans **[Saygin et al 1999]** et Ghomri **[Ghomri 2007]** où elle a surperformé les deux autres règles de sélection des routages FIFO / FA (First-In First-Out / First Available) et EPL (Equal Probability Loading) si chaque machine utilise la FIFO comme règle de séquençage.

Dans **[Hassam et Sari 2010]**, les auteurs ont étudié la règle DMM et proposé la règle DMM modifiée qui est une modification de la règle DMM qui visait à maintenir la même idée que la maximisation des coefficients de dissimilarité pour la sélection de différents routages alternatifs mais si tous les routages sont sélectionnés par des pièces, la pièce suivante va être acheminée dans le routage où la file d'attente de la première machine de ce routage contient au moins une place libre.

Hassam. A **[Hassam 2012]** a étudié l'approche de résolution le problèmes d'ordonnancement avec les méthodes de sélection de routages et il a proposé et développé de nouvelles méthodes dans ce domaine afin de résoudre certains de ces problèmes et améliorer les performances données par les méthodes existantes. Nous pouvons également trouver dans **[Souier et al 2010]** l'adaptation de plusieurs métaheuristiques (colonie de fourmi, algorithme génétique, recuit simulé, recherche tabou, essaims de particules et électromagnétisme) pour

résoudre un problème de sélection de routages dans un FMS avec un nombre important de pièces mais en temps différé, ils ont ensuite appliqué ces métaheuristiques avec succès sur le même problème en temps réel.

Souier .M **[Souier 2012]** s'est intéressé aux processus d'ordonnancement et de ré-ordonnancement dans un FMS avec flexibilité de routage et sous incertitudes. Les décisions d'ordonnancement sont prises en termes de comment des pièces arrivent au système, comment ces pièces sont routées vers les diverses machines alternatives et comment on fait les séquencements des pièces aux niveaux des files d'entrée des machines. Le système décisionnel proposé utilise les règles de priorité pour le lancement et le séquencement de pièces. En outre, le processus de routage de pièces est basé sur les essaims particulaires.

Dans le chapitre suivant, nous nous intéressons à l'adaptation d'une autre variante des algorithmes génétiques, premièrement l'algorithme mémétique avec gestion de population proposé par Sorensen et Sevaux [**Sevaux 2004**], qui représente une amélioration de l'algorithme mémétique, en se basant sur la gestion de population, en utilisant la mesure de distance dans l'espace des solutions, pour contrôler la diversité d'une petite population de solutions de haute qualité et deuxièmement la recherche dispersée (SS) proposée par Glover **[Glover 1995]**, qui a fait combiner par un processus intelligent, un nombre de solutions jusqu'à atteindre un optimum. La principale différence avec les algorithmes génétiques, est que la recherche de l'optimum local est guidée, dans le sens qu'un échantillon référence R est sélectionné parmi l'ensemble de la population. Cet échantillon est intensifié et mis à jour à chaque itération. Une fois les solutions de cet échantillon sont combinées, une recherche locale est lancée pour tenter d'améliorer les résultats obtenus. L'échantillon est mis à jour avec autant les bonnes que les mauvaises solutions, pour la résolution du problème de sélection de routages alternatifs dans un système flexible de production(FMS).

3.2 L'optimisation combinatoire

Cette thèse s'inscrit dans le domaine de l'optimisation combinatoire. Elle a été réalisée au sein de l'équipe ordonnancement du Laboratoire de productique de Tlemcen MELT. De nombreux secteurs de l'industrie sont concernés par les problèmes d'optimisation combinatoire. En effet, pour cela on s'intéresse à l'optimisation d'un système de production. Un problème d'optimisation combinatoire se caractérise généralement par un ensemble fini de configurations constitué des différentes valeurs prises par les variables de décision et une ou plusieurs fonction(s) dite objectif(s) Il y a de nombreuses formulation sous les définitions mathématiques

d'un problème d'optimisation, nous avons retenu celle de Blum et al**.[Blum et al. 2003]** qui ont décrit un problème d'optimisation P comme un triple (S, Ω, f) où :
S : Dans la plupart des problèmes, l'espace d'état (décision) est fini ou dénombrable. Les variables du problème peuvent être de nature diverse (réelle, entier, booléenne, etc.) et expriment des données qualitatives ou quantitatives.

$f: S \rightarrow R^{+}$: Le but à atteindre pour le décideur est représenté par la fonction objectif.

Ω : L'ensemble de contrainte définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité et permettent en général de limiter l'espace de recherche (solutions réalisables).

$s^{*} \in S\ telque f(s^{*}) \leq f(s),\ \forall s \in S$. La résolution optimale du problème consiste à trouver le point s^{*} ou un ensemble de points de l'espace de recherche qui satisfait au mieux la fonction objectif. Le résultat est appelé valeur optimale ou optimum. Notons que d'une manière similaire, on peut également définir les problèmes de maximisation en remplaçant simplement ≤ par ≥.

Néanmoins en raison de la taille des problèmes réels, la résolution optimale s'est souvent montrée impossible dans un temps raisonnable. Cette impossibilité technique impose la résolution approchée du problème, qui consiste à trouver une solution de bonne qualité (la plus proche possible de l'optimum). Il est vital pour déterminer si une solution est meilleure qu'une autre, que le problème introduise un critère de comparaison (une relation d'ordre). Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre.

La plupart des problèmes d'optimisations appartiennent à la classe des problèmes NP-difficile où il n'existe pas d'algorithme qui fournit la solution optimale en temps polynomial en fonction de la taille du problème et le nombre d'objectifs à optimiser. C'est pourquoi, il est souvent préféré une résolution à base d'heuristiques sur les problèmes de plus grandes tailles pour des raisons d'obtention de résultat. Cook et al ont **[Cook et al. 1998]** élaboré une théorie de la complexité en mesurant le temps de calcul en fonction de la taille du problème. Sa théorie distingue entre les algorithmes polynomiaux, du la complexité de l'ordre d'un polynôme et des autres algorithmes dits exponentiels. La puissance de calcul ne résout rien pour des complexités exponentielles, il est faux de croire qu'un ordinateur peut résoudre tous les problèmes combinatoires. Les méthodes exactes résolvent des problèmes d'optimisation de petit taille, mais si le problème est de grand taille, le temps d'exécution devient exponentiel donc dans ce cas il faut recourir à l'approximation pour trouver une solution proche de l'optimum en temps

raisonnable. Les méthodes approchées se fondent sur le principe des heuristiques, dont certaines sont consacrées au problème traité. Chacune se base sur des typiques particulières au problème considéré, pour cela on ne peut pas être généralisées donc les chercheurs se sont intéressés à des méthodes qui se basent sur des notions plus généralisées, cela a donné naissance à ce qu'on appelle les métaheuristiques, qui présentent une alternative intéressante, ces derniers regroupent différentes classes de méthodes :

- Les méthodes basées sur le voisinage telle que la recherche taboue.
- Les méthodes évolutives

3.3 Notions principales des Métaheuristiques

Les métaheuristiques ont émergé car les heuristiques ont rencontré des difficultés pour obtenir une solution raisonnable et de bonne qualité aux problèmes d'optimisation difficiles. Ces algorithmes présentent actuellement des alternatives intéressantes, elles sont plus complètes et complexes qu'une simple heuristique et permettent généralement d'obtenir une solution de très bonne qualité pour la résolution des problèmes d'optimisation difficile nécessite un temps élevé ou une grande mémoire de stockage pour lesquels on ne connaît pas d'algorithmes classiques plus efficaces. Elles utilisent pour rassembler de l'information des processus aléatoires et itératifs, pour explorer l'espace de recherche, un heuristique partage l'inconvénient d'être spécifiques aux problèmes dont lesquels il était utilisé. Par contre, une métaheuristique peut être adaptée pour différents types de problèmes sans modifications profondes, donc elle réduit les difficultés rencontrées dans les heuristiques. Ces méthodes sont souvent inspirées de ressemblances avec la réalité comme la biologie (recherche tabou, algorithmes évolutionnaires) l'éthologie (essaims particulaires, colonies de fourmis), et la physique (recuit simulé).

Il existe plusieurs concepts des métaheuristiques, les auteurs résument les propriétés attachées à la notion de métaheuristique

✓ Un voisinage : Une méthode basée sur un ou plusieurs voisinages et le risque de s'écouler sur des solutions de mauvaise qualité qui ne peuvent pas être améliorées par un déplacement dans le voisinage, c'est l'inconvénient principal de la recherche local dans ce cas la recherche a atteint un minimum local alors que le résultat risque de ne pas être satisfaisant parce que l'objectif est de trouver une solution proche de l'optimum global.

Par exemple, V (S) est un sous-ensemble de configurations de S qui est atteint à partir d'une transformation donnée de S, et S'∈ V (S) est une solution dite voisine de S [**Youcef et Ould-Saadi 2012].**

- ✓ Diversification : permet de générer des diverses solutions afin d'explorer l'espace de recherche dans une échelle globale **[Yang et al. 2014].**
- ✓ Intensification : le processus d'intensification vise à forcer une solution donnée à tendre vers l'optimum local de la zone à laquelle elle est attachée **[Boisson et al. 2008].**

Les principes communs et essentiels aux métaheuristiques sont exprimés par les notions classiques d'intensification et de diversification, d'une part et par les concepts de stratégie de recherche et d'adaptation ou d'auto adaptation de cette stratégie de recherche, d'autre part.

- ✓ Mémoire et apprentissage : les techniques utilisées dans les métaheuristiques vont de la simple recherche locale qui permet à l'algorithme de ne tenir compte que des zones où l'optimum global est susceptible de se trouver, évitant ainsi les optimas locaux, à des procédures complexes d'apprentissage.

On peut résumée cette analyse par les deux points suivants :

- ✓ Une métaheuristique peut modifier sa stratégie de recherche, pendant le cours même de la recherche, pour être plus efficace. Donc c'est l'adaptation de la stratégie de recherche.
- ✓ Une métaheuristique combine des tendances d'intensification et de diversification.

Les trois phases d'une métaheuristique itérative sont représentées par la figure suivante :

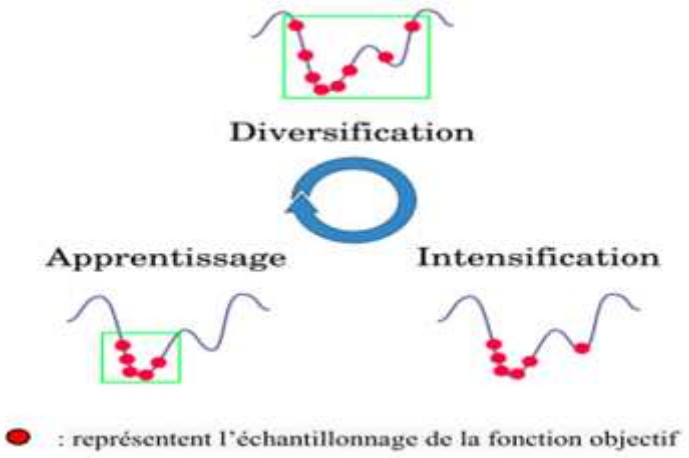


Figure 3.7. Les trois phases d'une métaheuristique itérative.

3.4 Classification des métaheuristiques

Malgré que les métaheuristiques ont plusieurs caractéristiques communes, il y'a des différences entre ces techniques par rapport à plusieurs critères de classification **[Talbi 2009]**.

Les métaheuristiques peuvent être classées selon leur principe de fonctionnement durant la recherche de la solution en deux catégories :

- Les métaheuristiques à solution unique (méthodes de recherche locale ou méthodes de trajectoire) et à base d'une population de solutions. Les métaheuristiques qui ne manipulent qu'une seule solution à la fois, tentent itérativement de la transformer ou de l'améliorer.
- Les métaheuristiques de la deuxième catégorie construisent un ensemble de solutions dans l'espace de recherche afin de pouvoir se diriger vers des solutions optimales.

Par fois les métaheuristiques sont classées en fonction du nombre de structures de voisinages utilisées. Les métaheuristiques les plus intéressantes sont basées sur plusieurs types de voisinages, puisqu'un minimum local relativement à un type de voisinage n'est pas nécessairement pour un autre type de voisinage.

On retrouve une classification dans plusieurs articles selon leur origine :

- Les métaheuristiques qui s'inspirent de phénomènes naturels
- Les métaheuristiques qui ne s'en inspirent pas.

Certain critère sépare les métaheuristiques à base de modèle des autres à base d'une mémoire de solutions. Les métaheuristiques qui font usage de l'historique de la recherche peuvent le faire de diverses façons. On différentie généralement les méthodes ayant :

- Une mémoire à court terme.
- Une mémoire à long terme.

Les métaheuristiques peuvent également être classées selon leur manière d'utiliser la fonction objective. Étant donné un problème d'optimisation consistant à minimiser une fonction f sur un espace S de solutions, certaines :

- Les métaheuristiques statiques : travaillent directement sur f alors que d'autres,
- Les métaheuristiques dynamiques : font usage d'une fonction g obtenue à partir de f en ajoutant quelques composantes qui permettent de modifier la topologie de l'espace de solutions, ces composantes additionnelles peuvent varier durant le processus de recherche.
- Une autre façon est de classifier les métaheuristiques en deux classes itératives et gloutonnes.

- Les algorithmes itératifs, on commence par une solution complète (ou population de solutions) puis on la transforme à chaque itération en utilisant les opérateurs de recherche.
- Les algorithmes gloutons commencent à partir d'une solution vide et à chaque étape une variable de décision du problème est affectée jusqu'à l'obtention d'une solution complète. La plupart des métaheuristiques sont de type itératif.

On peut aussi distinguer deux métaheuristiques :

- Les métaheuristiques déterministes qui utilisent des décisions déterministes
- Les métaheuristiques stochastiques : dans cette famille d'algorithmes, certaines règles aléatoires sont appliquées durant la recherche.

Néanmoins, nous y retrouvons toujours cet intérêt d'intensification et de diversification dans la recherche. La stratégie employée guide la recherche, parfois vers des zones spécifiques, parfois vers de nouvelles régions. Souvent doivent être ajustées précisément par le paramétrage de l'algorithme pour entrelacées les deux tendances. Elles partagent aussi les mêmes inconvénients : les difficultés de réglage des paramètres et le temps de calcul élevé…

Tableau 3-7 Classification des principales métaheuristiques

Métaheuristique	Nombre de solutions utilisées simultanément		Nombre minimum de structures de voisinage exploitées			Utiliser la fonction objectif.		Type de mémoire guidant la recherche		Utilisent de décisions	
	Mono-solution	Population	Aucune	Une	Plusieurs	Statiques	Dynamiques	A base de solution	A base de modèle	Déterministes	Stochastiques
Recherche locale itérée	X			X		X		X		X	
Recuit simulé	X			X			X	X			X
Recherche à voisinage variable	X				X	X		X		X	
Recherche tabou	X			X			X	X		X	
Recherche locale Guidée	X			X			X	X		X	
Algorithme évolutionniste		X	X					X			X
Algorithme mémétique		X		X				X			X
Algorithme Gestion de population		X		X		X		X			X
Recherche dispersée		X		X				X			X
Algorithme à estimation de distribution		X	X				X		X	X	
Optimisation par colonie de fourmis		X	X			X			X		X
Optimisation par essaim particulaire		X		X		X		X		X	

3.5 Les Algorithmes Evolutionnaires (AE)

Ce sont généralement des méthodes qui essayent de dissimuler le processus de l'évolution naturelle dans un environnement hostile lié au problème à résoudre et des mécanismes imitant des phénomènes observés en biologie. On peut nommer les algorithmes à colonies de fourmis, et plus exactement, ceux basés sur la biologie génétique, s'inspirant de phénomènes comme la reproduction, la sélection naturelle et les mutations

D'une manière schématique, on peut les assimiler à un processus d'optimisation où des individus qui évoluent dans le temps, afin de devenir de plus en plus acclimaté à un environnement donné. Le fonctionnent des AE est généralement de manière itérative c'est-à-dire une population initiale de solutions réalisables évolue progressivement, guidée par les

mécanismes cités précédemment, ils se sont prouvés être des alternatives efficaces aux méthodes traditionnels pour la résolution de nombreux problèmes d'optimisation. L'origine de ce type d'algorithmes était les algorithmes génétiques, d'autres formes plus évoluées ont fait leur apparition.

On distingue classiquement quatre catégories principales d'approches évolutionnaires : la programmation évolutionnaire (Evolutionary Programming), les stratégies d'évolution (Evolution Strategies), les algorithmes génétiques et la programmation génétique **[Blum and Roli, 2003]**. Nous donnons ci-dessous quelques éléments de comparaison tirés de **[Bäck et *al*., 1997]** :

- **Algorithmes génétiques** : l'opérateur de croisement est considéré comme étant le plus important des opérateurs. La mutation est appliquée avec des probabilités très faibles et agit en tant qu'opérateur d'arrière-plan. Le codage des solutions consiste en une représentation généralement binaire des individus.
- **Stratégies d'évolution** : le codage des solutions peut être réalisé par des structures de données plus complexes que dans les algorithmes génétiques. Par ailleurs, les opérateurs de mutation ont une place aussi importante que les opérateurs de croisement.
- **Programmation évolutionnaire** : elle est fondée essentiellement sur l'opérateur de mutation et n'utilise pas d'opérateur de croisement. Comme les Stratégies d'évolution, le codage des solutions peut faire intervenir des structures de données complexes. Cette approche a été développée initialement pour faire évoluer des automates à états finis.
- **Programmation génétique :** le nombre de programmes dans la population peut varier entre les générations, lorsque la sélection et la reproduction sont en cours.

3.6 L'Intensification et la diversification

Les métaheuristiques appliquent de manière itérative un ensemble d'opérateurs pour trouver des solutions, peut-être pas toujours optimales, en tout cas très proches de l'optimum et en un temps raisonnable dans le but d'améliorer ces solutions jusqu'à ce qu'un résultat satisfaisant soit calculé. Les principaux points critique pour tout métaheuristique ce sont l'intensification et la diversification, ils consistent à trouver un compromis entre les deux tendances duales suivantes

- Il s'agit d'une part d'intensifier l'effort de recherche vers les zones les plus prometteuses de l'espace de solutions.

- Il s'agit d'autre part de diversifier l'effort de recherche de façon à être capable de découvrir de nouvelles zones contenant de meilleures combinaisons.

Mais comme mentionné dans **[Mitchell 1998]**, il est essentiel pour la réalisation d'une métaheuristique de trouver un équilibre optimal entre la diversification et l'intensification. Ne pas préserver cet équilibre conduit à une convergence trop rapide vers des minimas locaux (manque de diversification) ou à une exploration trop longue (manque d'intensification). La métaheuristique peut être considérée comme un moyen efficace de produire des solutions acceptables par des essais à un problème complexe dans un délai raisonnablement pratique. La complexité du problème d'intérêt rend impossible la recherche de toutes les solutions ou combinaisons possibles.

L'objectif est de trouver une bonne solution possible dans un délai acceptable. Il n'y a pas, de garantie que les meilleures solutions peuvent être trouvées et nous ne pouvons même pas savoir si un algorithme fonctionnera et pourquoi si cela fonctionne, même si nous pouvons connaître les composants de base qui peuvent aider à travailler.

L'idée est d'avoir un algorithme efficace mais pratique qui fonctionnera le plus souvent et capable de produire des solutions de bonne qualité. Parmi les solutions de qualité trouvées, on s'attend à ce que certaines d'entre elles soient presque optimales, bien qu'il n'y ait souvent aucune garantie pour une telle optimalité. Cependant, de nombreuses métaheuristiques peuvent généralement avoir des propriétés de convergence globales et par conséquent, elles peuvent généralement trouver l'optimalité globale en pratique dans un nombre relativement restreint d'itérations ou d'évaluations de fonctions. Cela rend particulièrement approprier pour les métaheuristiques de résoudre l'optimisation globale. En principe, pour qu'un algorithme métaheuristique soit efficace, il doit avoir des capacités spéciales et le pouvoir générer de nouvelles solutions qui peuvent généralement être plus susceptibles d'améliorer les solutions précédentes existantes et aussi pouvoir couvrir les domaines de recherche les plus importants où l'optimum global peut résider. Une autre capacité est qu'un algorithme devrait pouvoir échapper à n'importe quel optimum local afin qu'il ne puisse pas être coincé dans un mode local. Une bonne combinaison peut conduire à une bonne efficacité dans des conditions appropriées, ce qui nécessite souvent d'équilibrer deux composantes importantes de toutes les métaheuristiques : exploration et exploitation. Cependant, elle-même est une tâche d'optimisation non résolue.

Un élément important de la métaheuristique moderne est l'exploration, souvent par l'utilisation de la randomisation **[Blum et Roli 2003], [Yang 2008],** ce qui permet à un

algorithme d'avoir la possibilité de sauter de n'importe quel optimum local afin d'explorer la recherche dans le monde entier. La randomisation peut également être utilisée pour la recherche locale autour du meilleur actuel si les étapes sont limitées à une région locale. Lorsque les étapes sont importantes, la randomisation peut explorer l'espace de recherche à l'échelle mondiale. Le réglage précis de la bonne quantité de hasard et l'équilibrage de la recherche locale et de la recherche globale revêtent une importance cruciale dans le contrôle de la performance de n'importe quel algorithme métaheuristique.

La connaissance empirique des observations et des simulations du comportement de convergence des algorithmes d'optimisation commun suggère que l'exploitation tend à augmenter la vitesse de convergence, tandis que l'exploration tend à diminuer le taux de convergence de l'algorithme. D'autre part, une trop grande exploration augmente la probabilité de trouver l'optimalité globale, mais avec une efficacité réduite, alors que l'exploitation forte tend à faire en sorte que l'algorithme soit piégé dans un optimum local. Par conséquent, il existe un bon équilibre entre la bonne quantité d'exploration et le bon degré d'exploitation. Malgré son importance, il n'y a pas de lignes directrices pratiques pour cet équilibre.

Donc, essentiellement, chaque algorithme utilise différents degrés d'exploitation et d'exploration, souvent loin d'être optimal **[Blum et Roli 2003]**. Certains algorithmes peuvent avoir un équilibre intrinsèquement meilleur entre ces deux composants importants que d'autres algorithmes, c'est l'une des raisons pour lesquelles ils peuvent être plus performants **[Yang 2008], [Yang 2010]**.

De plus, en considérant les algorithmes génétiques, il faut que des opérateurs (par exemple, des mutations) injectent diverses solutions dans la population car l'opérateur de croisement dépend fortement de la diversité de la population **[Yang 2010]**. Blum et. Al. **[Blum et Roli 2003]** mentionnent que, pendant une longue période, l'hypothèse commune était que la diversification se faisait par mutation et croisement alors que l'intensification était atteinte avec l'opérateur de sélection.

Eiben et Schippers **[Eiben et Schippers 1998]** étaient les premiers à remettre en question cette vue. On prétend que si la mutation présente de nouveaux matériaux (diversification), elle conserve également la plupart des informations du parent et est donc aussi un opérateur d'intensification. De même, un croisement crée une nouvelle solution (diversification) mais préserve également la plupart des matériaux (intensification). Cette vue est également partagée dans d'autres publications **[Hansheng et al. 1999]** et comme on l'a remarqué dans **[Blum et**

Roli 2003] la ligne entre sélection, croisement et mutation est floue et il est difficile de distinguer entre l'intensification et la diversification. Un autre point mentionné est que d'autres paramètres de l'AG (par exemple, la taille de la population) ont également une influence sur le comportement d'un opérateur et donc sur le ratio d'intensification et de diversification. Malgré la quantité de travail qui a été publiée dans ce domaine, il n'y a pas de définition claire et cohérente de l'intensification et de la diversification. Comme mentionné dans **[Eiben et Schippers 1998]**, la plupart des auteurs laissent leur définition implicite et utilisent la signification intuitive des concepts pour expliquer le fonctionnement des EA. Cela rend difficile la mesure de l'intensification et de la diversification que l'opérateur présente. Dans ce chapitre, nous utiliserons le terme diversification dans le sens où les opérateurs devraient introduire de nouveaux matériaux dans une solution alors que la qualité résultante peut être négligée. Nous définissons la notion d'intensification comme la réutilisation du matériel existant pour améliorer la qualité de la solution, tout en minimisant la quantité de diversification.

3.7 Comparaissent entre les variantes d'algorithmes génétiques et GA

Le premier problème pratique qui se pose à un utilisateur confronté à une application concrète est d'effectuer un choix parmi les différentes métaheuristiques disponibles. Ce choix est d'autant plus difficile qu'il n'existe pas de comparaison générale et fiable des différentes métaheuristiques. Cependant, il est possible de caractériser les métaheuristiques selon les critères (Intensification /Diversification), ce qui pourrait faciliter ce choix. Le tableau de synthèse ci-dessous met en relation ces critères avec quatre métaheuristiques parmi les plus représentées.

L'idée de base de MA | PM vient de la gestion de la population, grâce à une mesure de distance dans l'espace des solutions. Cette opération permet de contrôler la diversité des individus en filtrant l'entrée des enfants dans la nouvelle population et d'éviter ainsi la stagnation et la convergence prématurée. Contrairement aux algorithmes mimétique MA traditionnels, l'activation de la recherche locale ne dépend pas des paramètres de diversité dans l'espace des solutions. La recherche de dispersion SS utilise un ensemble de référence pour combiner ses solutions et en construire d'autres et elle génère un ensemble de référence à partir d'une population de solutions.

Les MA | PM, SS, MA et GA peuvent également être comparés en fonction de l'intensification et de la diversification comme dans le tableau 3.8.

Tableau 3-8 Comparaissent en fonction de l'intensification et de la diversification

Métaheuristiques	**Intensification**	**Diversification**
GA : -Grande population -Des solutions de qualité mixte -Pas d'amélioration locale	-Selection -Croisement -Replacement	-Mutation
MA|PM : -Petite population -Haute qualité -L'amélioration locale	-Sélection -La recherche locale -Mise à jour de Δ	-Mutation
SS : -grande population ensemble de référence de taille réduite -Diversité mesure -La recherche locale	-boucle interne -La recherche locale -Mise à jour de l'ensemble de référence	-Sélection d'un ensemble diversifié -Remplacer une partie de la solution
MA : - taux de recherche locale -Ajout d'un opérateur de mutation -mutation combinaisons infinies	-Selection - La recherche locale	-Replacement -Croissant -Mutation

3.8 CONCLUSION

De nombreuses études montrent que les métaheuristiques peuvent obtenir de bonnes solutions et nécessitent moins de temps de calcul que les méthodes exactes. La planification des FMS est parmi les problèmes d'optimisation combinatoire les plus difficiles et intéressantes.

Pour cette raison, les recherches ont utilisé différentes approches pour résoudre ce problème.

De nombreux chercheurs conviennent que la qualité d'une approche d'optimisation métaheuristique résulte largement de l'interaction entre stratégies d'intensification et de diversification, une bonne métaheuristique est celle qui peut bien équilibrer ces deux stratégies pour éviter les optimums locaux et une trop longue exploration.

Les algorithmes évolutionnaires (EA), en particulier les algorithmes génétiques ont été trouvés pour fournir de bonnes approximations de la solution optimale. En raison de leur simplicité et de leur facilité de mise en œuvre, nous présentons un algorithme mémétique avec gestion de population et approche de recherche de dispersion pour résoudre un problème de sélection de routages alternatifs dans un problème combinatoire NP-difficile avec introduction des pannes.

Dans ce domaine des problèmes dits " combinatoires ", il n'existe pas d'heuristique qui soit meilleure qu'une autre, pour cela dans la suite de la thèse on va adapter deux méthodes pour la résolution d'un problème d'ordonnancement dans un système flexible de production (FMS).

Chapitre 4

4 Algorithmes à base des variantes d'algorithmes génétiques pour les décisions de routages de pièces

Ce chapitre présente le modèle FMS étudié, les algorithmes des métaheuristiques adaptées à la résolution de notre problème ainsi que les résultats obtenus et leurs interprétations après plusieurs simulations du modèle

Sommaire

Chapitre 4 Algorithmes à base des variantes d'algorithmes génétiques pour les décisions de routages de pièces

4.1 Introduction

Les systèmes de production actuels ne cessent de croître en complexité. Cette complexité résulte des exigences du marché, de la concurrence, de la qualité ainsi que de la densité et de la diversité des produits qu'ils traitent. Dans ce type de problèmes on se trouve face à un problème d'ordonnancement. Le choix de la méthode de résolution dépend fortement de la taille et de la complexité du problème. Si le problème est de grande taille et de complexité importante, il n'est pas possible de fournir un algorithme exact pour ce genre de systèmes. On s'intéresse alors naturellement à des algorithmes non exacts appelés 'heuristiques' qui présentent des alternatives plus intéressantes. Parmi les problèmes qui sont souvent classés NP-Difficiles, ceux d'ordonnancement forment une grande partie, ce qui justifie l'utilisation des méthodes approchées pour leur résolution.

Les métaheuristiques sont analysées au travers de composants d'intensification et/ou de diversification**.** L'intensification et la diversification comme étant deux besoins conflictuels dont l'équilibre est essentiel au bon fonctionnement d'une heuristique **[Meignan 2008].**

Dans ce chapitre nous allons interpréter les résultats des simulations avec présence des pannes. Ce chapitre contient deux parties principales, la première est réservée à la présentation du modèle FMS étudié et à l'adaptation des algorithmes l'algorithme mémétique avec gestion de population et l'algorithme recherche dispersée. La seconde partie est organisé comme suit : Une première section où nous présentons une analyse de sensibilité de ces algorithmes, en étudiant l'influence de variation des paramètres internes de la métaheuristique sur ses performances, pour pouvoir tirer des informations sur les meilleures valeurs des paramètres de la méthode. Une deuxième section est consacrée à présenter les résultats des simulations en combinant les différentes méthodes de sélection. Ces approches ont été comparées avec d'autres méthodes évolutionnaires, pour pouvoir tirer une idée sur l'efficacité de chacune des méthodes utilisées. Ensuite nous avons donné les résultats et les interprétations obtenus des différentes simulations effectuées en ajoutant des pannes aux machines du système. Nous terminons le chapitre par une conclusion.

4.2 Présentation du modèle FMS étudié

Ce système de production flexible nous allons étudier a déjà été abordé dans la littérature, **[Saygin et Kilic 1999]** il contient sept machines et deux stations : une station de chargement et une de déchargement. Six types de pièces différentes sont traités dans le système.

(a) Une fraiseuse horizontale

(b) Une fraiseuse verticale

(c) Une tour

(d) Un étau limeur

Figure 4.1. Photographie des machines qui composent le système

Les machines et les stations qui constituent le système étudié sont définies comme suit :

- Deux fraiseuses verticales (FV).
- Deux fraiseuses horizontales (FH).
- Deux tours (T).
- Un étau limeur (TP).
- Une station de chargement (SC).
- Une station de déchargement (SD).

Chaque machine a une file d'attente d'entrée et une file d'attente de sortie avec une capacité de deux pièces, la station de chargement a également une file d'attente d'entrée d'une capacité égale à deux. La station de déchargement a une capacité d'une seule pièce. La configuration du système flexible de production est représentée dans la figure suivante :

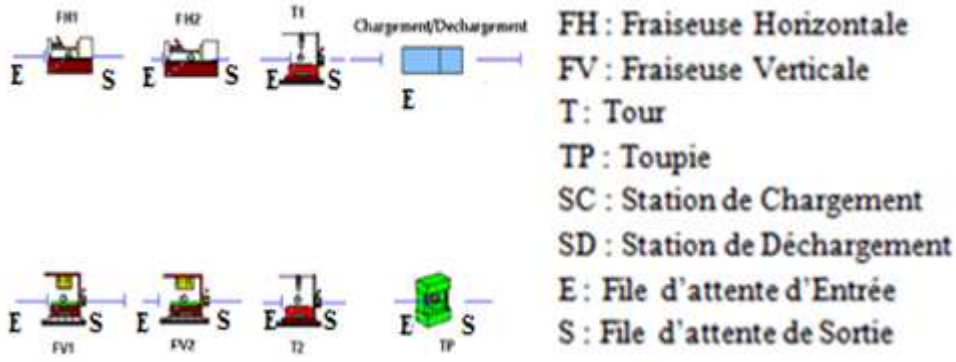


Figure 4.2. Configuration du modèle FMS étudié

Les routages alternatifs et les temps de traitement de chaque type de pièce sont donnés dans le tableau suivant (4.1) :

Tableau 4-1 Routages alternatifs et temps de traitement des pièces **[Saygin et Kilic 1999]**

Type des pièces	Taux d'arrivée	Les numéros des routages	Routages et temps de traitement (min)
A	17%	1	SC-T1(30)-FV1(20)-SD
		2	SC-T1(30)-FV2(20)-SD
		3	SC-T2(30)-FV1(20)-SD
		4	SC-T2(30)-FV2(20)-SD
B	17%	5	SC-T1(20)-TP(1)-FV1(15)-SD
		6	SC-T1(20)-TP(1)-FV2(15)-SD
		7	SC-T2(20)-TP(1)-FV1(15)-SD
		8	SC-T2(20)-TP(1)-FV2(15)-SD
C	17%	9	SC-T1(40)-TP(1)-FV1(25)-SD
		10	SC-T1(40)-TP(1)-FV2(25)-SD
		11	SC-T2(40)-TP(1)-FV1(25)-SD
		12	SC-T2(40)-TP(1)-FV2(25)-SD
D	21%	13	SC-T1(40)-TP(1)-T1(20)-FH1(35)-SD
		14	SC-T1(40)-TP(1)-T1(20)-FH2(35)-SD
		15	SC-T1(40)-TP(1)-T2(20)-FH1(35)-SD
		16	SC-T1(40)-TP(1)-T2(20)-FH2(35)-SD
		17	SC-T2(40)-TP(1)-T1(20)-FH1(35)-SD
		18	SC-T2(40)-TP(1)-T1(20)-FH2(35)-SD
		19	SC-T2(40)-TP(1)-T2(20)-FH1(35)-SD
		20	SC-T2(40)-TP(1)-T2(20)-FH2(35)-SD
E	20%	21	SC-T1(25)-TP(1)-T1(35)-FH1(50)-SD
		22	SC-T1(25)-TP(1)-T1(35)-FH2(50)-SD
		23	SC-T1(25)-TP(1)-T2(35)-FH1(50)-SD
		24	SC-T1(25)-TP(1)-T2(35)-FH2(50)-SD
		25	SC-T2(25)-TP(1)-T1(35)-FH1(50)-SD
		26	SC-T2(25)-TP(1)-T1(35)-FH2(50)-SD
		27	SC-T2(25)-TP(1)-T2(35)-FH1(50)-SD
		28	SC-T2(25)-TP(1)-T2(35)-FH2(50)-SD
F	8%	29	SC-FH1(40)-SD
		30	SC-FH2(40)-SD

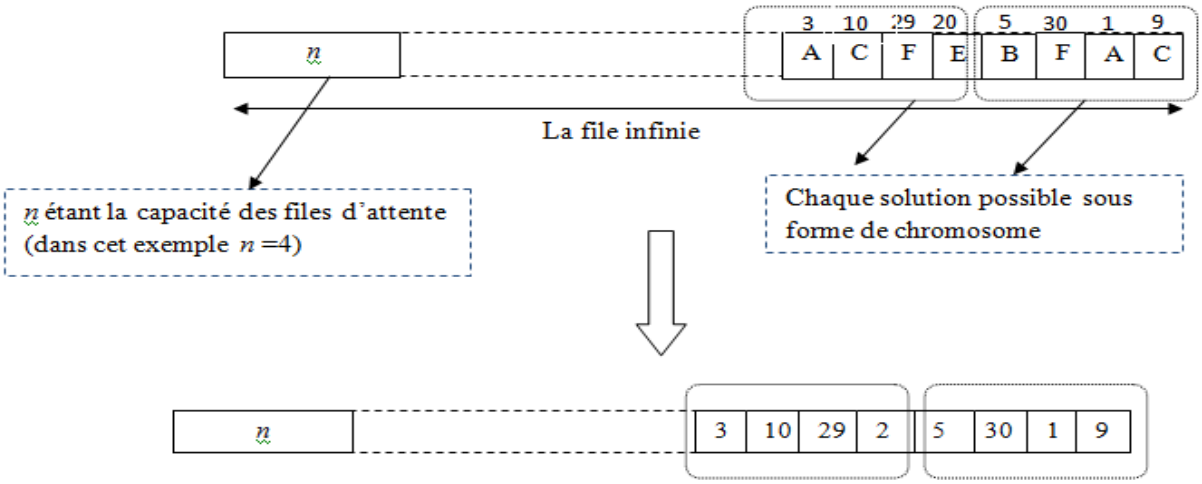
4.3 Adaptation des métaheuristiques pour la résolution de notre problème

Avant de présenter l'adaptation des deux algorithmes il est nécessaire de présenter quelques notions essentielles liées aux deux méthodes :

4.3.1 Chromosomes et évaluation

Parmi les éléments spécifiques des algorithmes de type génétique, le codage des solutions utilisé sous la forme de chromosomes (individu) appropriés est un choix particulièrement critique, dans la conception pour ces algorithmes, nommé.

En particulier pour des problèmes comportant différents niveaux d'information, comme c'est le cas pour le problème des décisions de routages de pièces dans un système flexible de production. Dans notre cas, un individu est une chaîne contenant les routages choisis des pièces présentes dans la file infinie.



A	17%	1	SC-T1(30)-FV1(20)-SD

Figure 4.3. Exemple d'un codage possible pour une capacité de file d'attente=4

Nous donnons un exemple de codage pour notre problème. Etant donné que la capacité des files d'attente est 4 et supposons que les premières pièces dans la file infinie sont comme suit : A, B, E et F. Les routages possibles sont donc : 9, 1, 30, et 5 pour A ; 2, 29, 10, et 3 pour B; 29, et 30 pour E ; 5, 6, 7, et 8 pour F, et pour E les routages sont numérotés de 20, à 28.

4.3.2 Fonction objectif

La fonction objectif est une conception de base pour mise en œuvre d'une métaheuristique Chaque chromosome est évalué par une valeur appelée fitness $F(S)$, représentée comme une fonction qui fait correspondre à chaque solution S une valeur du coût, donc une fonction de coût que l'on cherche à maximiser ou à minimiser par rapport à tous les paramètres et les contraintes concernés, l'élément d'évaluation de la qualité de la solution, choisi à priori au hasard, constitue la population initiale, et elle est utilisée pour guider la recherche vers de bonnes solutions de l'espace de recherche pour cela le processus de recherche peut conduire à de mauvaises solutions si elle n'est pas clairement définie.

La fonction objectif dans notre travail est le produit de charges des routages. Le but étant de maximiser cette fonction, afin d'équilibrer les charges des routages, on parle de charges non pas en terme de nombre des pièces mais en terme du temps opératoire, (on calcule le temps restant de la pièce au cours du traitement et le temps de traitements des pièces en attente d'être traitées sur cette machine, ces pièces sont celles contenues dans sa file d'attente d'entrée et celles contenues dans la file d'attente de sortie de la machine précédente dans le même routage) ce qui donnera donc un bon équilibre entre les routages, la figure 4.4.c présente cette définition.

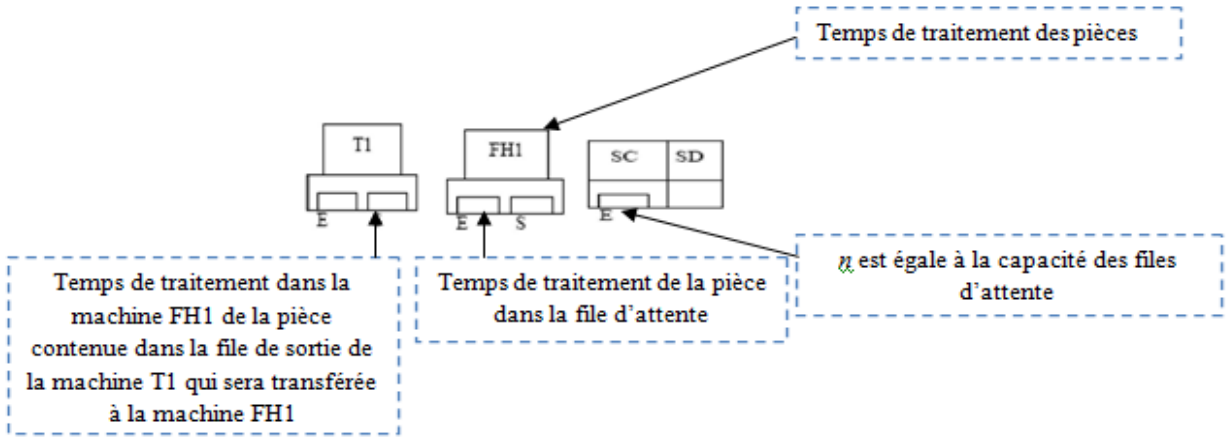


Figure 4.4. Un des routages de la pièce type A(1)

La maximisation de cette fonction (le produit des temps opératoires est défini par cette fonction :

$$F(t) = \prod_{j=1}^{30}(\sum_{i=1}^{m} O(i,j)) \quad (4.1)$$

m : le nombre de machines

j : le nombre de routages

O : temps opératoires

Pour ce faire, nous avons pris en considération les types des premières pièces existantes dans la file infinie (*n* est égale à la capacité de la file d'attente de la station de chargement). Par la suite, chaque pièce sera affectée à un routage selon son type.

4.3.3 Sélection des parents et croisement

La sélection des individus identifie et favorise les meilleurs individus, cette sélection se base sur la fonction objectif où elle se réalise aléatoirement, comme dans notre cas. Le croisement joue un rôle important dans les performances des algorithmes de type génétique, il engendre un ou plusieurs enfants à partir de la combinaison de deux parents, grâce au codage des individus (solutions) sous forme de chromosomes et d'explorer l'espace de recherche en combinant les différentes solutions.

Cela se fait en choisissant un endroit de coupure le long des deux chromosomes, puis en reliant la partie gauche de l'un à la partie droite de l'autre, et vice versa.

Ce type de croisement à un point sur chaque sous-partie du chromosome permet une transmission génétique à la fois fidèle aux parents (une seule coupure par sous-partie), et suffisamment diversifiant, voir les deux chromosomes d'après la figure 4.5 suivants :

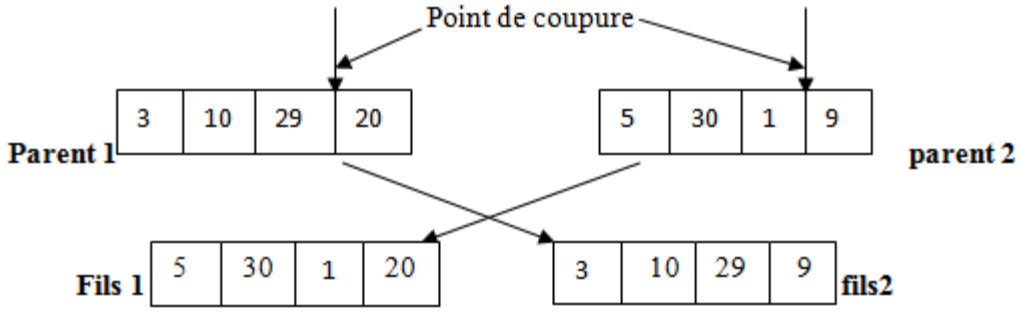


Figure 4.5. Exemple d'un croisement de deux solutions parents, deux enfants obtenus par ce croisement.

Un point de coupure est généré aléatoirement au niveau des deux parents sélectionnés et leurs parties sont inters changés, reliant maintenant les extrémités.

4.3.4 Méthodes de recherche locale

Une des particularités du MA/PM est de travailler sur une petite population de solutions de grande qualité. Pour cela, les solutions générées subissent une recherche locale. L'inclusion d'une recherche locale dans une métaheuristique permet d'intensifier la recherche autour d'une solution donnée.

Nous avons choisi comme méthode de recherche locale la méthode de descente. Le résultat dépend du voisinage retenu, de la solution initiale *x* et des voisins de la solution. En effet, suivant la solution initiale ou l'ordre de visite des voisins, on peut obtenir des solutions différentes. C'est-à-dire qu'à partir d'une solution trouvée on fait une recherche qui passe d'une solution à une autre solution voisine et à chaque fois qu'on trouve une solution améliorante on l'utilise comme nouvelle solution de départ, la nouvelle recherche s'effectue dans le voisinage de la nouvelle solution trouvée. Dans l'algorithme MA⁄PM, cette recherche s'effectue sur chaque individu de la population générée initialement, puis elle s'applique sur chaque nouvel individu obtenu par le croisement, c'est-à-dire sur chaque nouvelle génération de solutions, cela veut dire qu'on tente toujours de choisir les meilleurs individus de la population initiale ou de la génération construite.

4.3.5 Gestion de population

Une autre question importante à nos yeux et indispensable pour le recherche dispersée et l'algorithme mémétique avec la gestion de population est la définition d'une mesure de distance entre individus. Celle-ci doit être capable de refléter une similitude ou une différence entre deux individus. On pourra alors en déduire la distance d'un individu à la population toute entière. En fonction du problème, du codage des individus et de la distance retenue, l'évaluation peut-être plus ou moins couteuse en temps de calcul. Mais parfois il peut être bon de passer du temps pour calculer cette distance si elle conduit au final à une meilleure diversité de la population.

La principale caractéristique de l'algorithme mémétique avec gestion de population vient de la gestion de la population grâce à une mesure de distance dans l'espace des solutions. Cette opération permet de contrôler la diversité des individus en filtrant l'entrée des enfants dans la population. Pour constituer le groupe de solutions élites, une mesure de distance doit être élaborée, une mesure de dissimilarité ou la ressemblance des individus dans la population, pour éviter une convergence rapide de la population, il est important de maintenir un certain niveau de diversité au sein des populations intermédiaires. Le paramètre de diversité peut être mesuré de la manière suivante :

$$\{\Delta = \sum \Delta(\boldsymbol{S_i}, \boldsymbol{S_j})\} \tag{4.2}$$

La distance est le nombre de caractéristique qui différent entre deux solutions. Si S_1=(9,30,10,5) et S_2=(20,15,10,2) alors $\Delta(S_1, S_2)=3$. Plus deux individus sont distants, donc ils sont considérés comme étant différents.

4.3.6 Mutation

Son principal rôle est la diversification de la population qui, après plusieurs générations, tend à s'uniformiser. Dans notre cas la mutation est la modification des routages de certaines pièces en lui donnant un routage différent parmi les routages possibles. La modification du routage se fait aléatoirement.

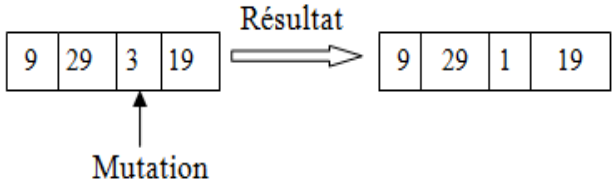


Figure 4.6. Exemple de mutation

4.4 Algorithme mémétique avec gestion de population

L'adaptation de l'algorithme mémétique avec gestion de population utilisé est comme suit :

Algorithme 3.1 : MA/PM

1. **S'il y'a** une place libre dans la station de chargement alors
2. **Initialiser** : générer une population initiale P de solutions (cela revient dans notre cas à affecter les *n* premières pièces de la file infinie à des routages d'une façon aléatoire en respectant les types de pièces), Génération d'une population aléatoire.
3. **Tant que** (critère d'arrêt est non atteint : nombre d'itérations)
4. **Pour** chaque individu
5. **Evaluation** du fitness de cet individu (le produit des charges de routages).
6. **Si** la fonction objectif est améliorée alors mettre à jour la meilleure solution (celle améliorante).
7. **Fin pour**
8. **Fixer** le paramètre de la diversité Δ
9. **Répéter**
10. **Sélection** : choisir deux solutions p_1 et p_2
11. **Croisement** : combiner deux solutions parents p_1 et p_2 pour former des nouvelles solutions c_1 , c_2
12. **Recherche locale** : appliquer une procédure de recherche locale sur c_1, c_2
13. **Pour** toute solution c **faire**
14. **Tant que** c ne satisfait pas la condition d'addition **Faire**
15. **Mutation** (c)
16. **Fin Tant que**
17. **Supprimer** la solution de la population
18. **Addition** : de solution :$\boldsymbol{p} \leftarrow \boldsymbol{p} \cup \boldsymbol{c}$
19. **Fin Pour**
20. **Mettre** à jour le paramètre de diversité Δ
21. **Jusqu'à** satisfaire un critère d'arrêt
22. **Fin Tant que**.
23. **Fin Si**

La procédure de recherche locale utilisée dans cet algorithme mémétique est la méthode de descente donnée par l'algorithme suivant :

Algorithme 3.2 : Méthode de descente

(1) Pour chaque individu (qui représente une condition initiale x pour cette méthode)
(2) Tant que la condition d'arrêt n'est pas vérifiée faire
(3) Modifier les routages de certaines pièces (trouver une autre solution voisine x' de x)
(4) Si la fonction objectif est améliorée (produit des charges des routages) :
$f(x') > f(x)$ **alors :**
(5) Remplacer x par x'
(6) Finsi
(7) Fin tant que

4.5 Rechercher dispersée *(scatter search)*

Le pseudo code de la recherche dispersée est donné comme suit :

Algorithme 3.3 : La recherche dispersée (SS)

1. **Initialiser** : générer une population initiale P de solutions cela revient dans notre cas à affecter les n premières pièces de la file infinie à des routages d'une façon aléatoire en respectant les types des pièces
2. **Mettre** les meilleures solutions trouvées dans une population de référence R (on calcule le produit des charges de routages)
3. **Tant que** le critère d'arrêt non satisfait (nombre d'itérations) faire
4. A=R → prendre la population de référence
5. **Tant que** A≠∅ faire
6. **Combiner** les solutions (B←R×A)
7. **Améliorer** les solutions B
8. **Mettre** à jour les solutions R
9. **Garder** les meilleures solutions à partir de R ∪ B
10. A←B−R
11. **Fin** tant que
12. Supprimer mauvaise solution dans R
13. Ajouter de nouvelles solutions diverses dans R
14. **Fin** tant que

4.6 Analyse de sensibilité

Tout comme pour les autres méthodes, les performances d'un algorithme à base génétique dépendent fortement des paramètres choisis pour cet algorithme. Une série de tests et de simulations doivent être effectués pour déterminer les valeurs de ces paramètres. Pour des bonnes performances de ces algorithmes il faut :

- Une population de taille modérée.
- Une probabilité de croisement élevée.

- Une probabilité de mutation faible (inversement proportionnelle à la taille de la population).
- Mesurer la diversité.

4.6.1 Sensibilité par rapport à la taille de la population pour la recherche dispersée

On extrait un *ensemble de référence R* contenant les meilleures solutions de la population initiale. Ensuite ces solutions sont combinées entre elles (et avec les nouvelles solutions générées) puis améliorées jusqu'à ce qu'il n'y ait plus de nouvelles solutions générées par combinaison.

Tableau 4-2 Le taux de production en fonction de la taille de la population pour une taille de file = 2 (T.A .P: Taux d'arrivée des pièces (1/min))

T.A.P	1/5	1/10	1/15	1/20	1/25	1/30	1/35	1/40
N=5	23,07	45,95	68,99	96,64	99,99	99,99	99,99	99,99
N=10	23,54	47,21	71,00	99.99	99.99	99.99	99.99	99.99
N=15	23,88	47,78	71,63	99,99	99,99	99,99	99,99	99,99
N=20	24,17	48,48	72,61	99.99	99.99	99.99	99,99	99,99
N=25	24,34	48,92	723,21	99,99	99,99	99,99	99,99	99,99
N=30	24,64	49,00	73.72	99,99	99,99	99,99	99,99	99,99

Les plus grandes valeurs sont obtenues pour une taille de la population égale à 30 lorsque la population trop grande, le temps de calcul augmente et demande un espace mémoire important

4.6.2 Sensibilité par rapport au paramètre de diversité pour MA/PM

Le paramètre de diversité est un facteur tout aussi important. Il permet en fonction de l'ensemble des individus de la population de plus ou moins diversifier et de plus ou moins intensifier la recherche.

Tableau 4-3 Le taux de production en fonction du paramètre de diversité pour une taille de file = 6 et un taux d'arrivée des pièces =1/18.

Paramètre de diversité	Taux de production (%)
Δ=0	87.71
Δ=1	87.86
Δ=2	87.76
Δ=3	87.84
Δ=4	87.94
Δ=5	88,03

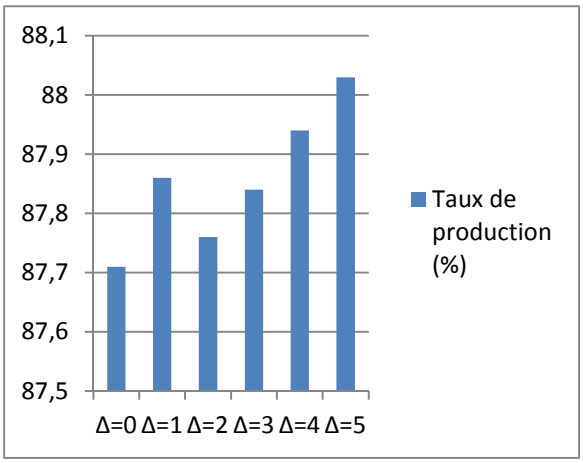


Figure 4.6 Taux de production en fonction du paramètre de diversité.

Pour voir l'effet de l'augmentation des valeurs du paramètre Δ, on augmente à chaque fois sa valeur et on compare, on remarque que plus on augmente Δ plus le taux de production augmente sauf pour Δ=1.

4.7 Etude comparative

Dans cette section, nous présentons les résultats obtenus par l'algorithme mémétique avec gestion de population et SS adapté pour la résolution de notre problème. Pour pouvoir évaluer les performances et l'efficacité de l'algorithme, nous avons effectué plusieurs simulations. Les critères de performance choisis sont : le taux de production du système, le temps du cycle et le taux d'utilisation des machines.

Nous précisons que, pour tout le reste du document, le taux de production est calculé en divisant le nombre de pièces sortantes du système par le nombre de pièces arrivées.
Les résultats de simulations comparés à ceux obtenus par l'algorithme génétique adapté par Souier et al. **[Souier et al. 2010]** et MA adapté par Houbad. y **[Houbad 2011]** pour résoudre le même problème de sélection de routages alternatifs.

Les algorithmes proposés sont codés en Java et simulés sur 20000 heures avec un temps de régime transitoire de 3000 heures et dix réplications sont effectuées. L'étude est réalisée avec présence de pannes dans le système. Une panne se produit selon une distribution exponentielle avec un MTBF = 100 heures et MTTR = 2 heures.

Les critères d'évaluation de la performance étudié sont le taux de production, la capacité et la disponibilité des machines et la variation et la rapidité de l'arrivée des pièces…
Pour pouvoir évaluer les efficacités des deux métaheuristiques, nous avons utilisés comme critères de performance :

- ✓ **Le taux de production :** c'est l'un des critères les plus importants, il est calculé en divisant le nombre de pièces sortantes du système par le nombre des pièces entrantes à la file infinie, afin de faire une régularisation.

  T.A.P: Taux d'arrivée des pièces (1/min).

  C.F.A: Capacité de file d'attente.

  T.M: les types des machines

  MA|PM : Algorithme mémétique avec gestion de population.

  MA: Algorithm memetic

  GA: Algorithm genetic

  SS : Recherche dispersée.

Tableau 4-4 Taux de sortie des pièces pour une capacité de file d'attente = 2

T.A.A(1/min)	1/5	1/10	1/15	1/20	1/25
(AG)	23,12	46,22	69,02	92,00	99,99
(MA)	23,84	47,42	71,21	94,34	99,99
(MA|PM).	23.95	47.83	71.61	94.66	99.99
(SS)	23,98	47,90	72,02	95,21	99,99

Le tableau 4-4 donne l'évolution du taux de production pour différentes valeurs de la capacité des files d'attente. IL est clair que la recherche dispersée (SS) donne de meilleurs résultats et dépasse les taux de production donnés par les autres algorithmes.

Tableau 4-5 Taux de sortie des pièces pour un taux de création de pièces = 1/20

C.F.A	2	4	6	8
(AG)	92.00	98,99	99,99	99,99
(MA)	93,85	99,99	99,99	99,99
(MA|PM)	94,34	99.99	99,99	99,99
(SS)	95,21	99,99	99,99	99,99

A partir des résultats montrés sur tableau 4-5, nous pouvons remarquer que le taux de production du système est plus important pour les algorithmes mémétique avec gestion de population et la recherche dispersée pour un taux de création de pièces égale à 1/20 et pour toutes la taille des files d'attentes égale à 2 ce qui veut dire que le système donne un meilleur rendement si on utilise ces deux algorithmes.

- ✓ **Le temps de cycle :** le temps de cycle d'une pièce est le temps de présence d'une pièce dans le système depuis qu'elle entre dans la station de chargement jusqu'à ce qu'elle quitte la

station de déchargement. Nous nous intéressons au temps du cycle moyen (par minute) de toutes les pièces sortantes du système.

Tableau 4-6 Temps de cycle pour une capacité de file d'attente = 2

T.A.p(1/min)	1/5	1/10	1/15	1/20	1/25
(AG)	176,20	175,20	174,60	170,80	104,60
(MA)	173,90	174,60	175,10	170,12	103,2
(MA|PM).	167,97	167,27	174,13	167,97	102,00
(SS)	166,93	169,95	167,59	166,93	97,27

Tableau 4-7 Temps de cycle pour un taux de création de pièces =1/20

C.Q	2	4	6	8
(AG)	170,80	161,00	150,90	145,40
(MA)	170,12	157,80	144,70	149,60
(MA|PM).	167.97	150,48	145,94	151,55
(SS)	166,93	145,22	150,71	149,16

Le tableau 4-6 et Le tableau 4-7, montrent que les valeurs du temps de cycle obtenus sont proches pour toutes les métaheuristiques, nous pouvons dire que la recherche dispersée a pu améliorer le temps de cycle pour un système très saturé et de petites files d'attente où elle donne les meilleurs temps de cycle, si la taille de file d'attente augmente l'algorithme mémétique donne les meilleurs temps de cycle pour taille file égale à 6 mais pour taille file égale 8 les temps de cycle sont les minimaux pour AG.

- ✓ **Le taux d'utilisation des machines**

Le taux d'utilisation des machines est un critère important dans la mesure de la performance des systèmes de production, ce taux a été défini par le pourcentage du temps d'utilisation des machines par rapport au temps de simulation. Nous présentons le taux d'utilisation des machines T, FV, FH et TP.

Tableau 4-8 Taux d'utilisation de toutes les machines du système pour une capacité de file d'attente=2

T.M	T.A.P (1/min)	1/5	1/10	1/15	1/20	1/25
T	AG	88,24	88,32	88,29	88,53	76,57
	MA	88,31	88,42	88,32	88,41	76,17
	SS	88,67	88,63	88,44	88,13	76,91
	MA|PM	88,68	88,68	88,55	88,74	76,62
FV	AG	28,04	28,05	27,72	27,10	25,02
	MA	29,20	29,86	28,87	27,83	25,57
	SS	32,16	31,96	31.75	31.20	24,38
	MA|PM	30.34	30,50	30,54	30,16	25,02
FH	AG	41,25	41,22	41,36	41,63	34,62
	MA	40,66	40,72	40,73	40,82	40,66
	SS	38,50	38,68	38,91	39,22	35,51
	MA|PM	39,99	39,80	39,91	39,91	34,62
TP	AG	2,324	2,322	2,330	2,340	1,950
	MA	2,291	2,295	2,295	2,300	1,985
	SS	2,165	2,176	2,178	2,207	2,001
	MA|PM	2,351	2,340	2,346	2,347	1,950

On voit clairement que l'algorithme mémétique avec gestion de la population est puissant par rapport aux autres algorithmes pour les taux d'utilisation des machines T et TP, mais pour le taux d'utilisation des machines FH, les meilleurs résultats sont donnés par l'AG et SS.
Les taux d'utilisation des machines des créations de partie = 1/25 pour les MA dans la HMC et TP sont plus élevés que ceux obtenus par tous les algorithmes. Les taux d'utilisation des machines FV sauf pour la création de piéces = 1/25 pour les SS sont plus élevés que ceux obtenus par le reste des algorithmes.

D'après les résultats expérimentaux, on peut voir que les résultats obtenus par l'algorithme mémétique avec la gestion de population et la recherche de dispersée sont, en général, meilleurs que ceux obtenus par l'Algorithme Génétique et MA. Un MA | PM et SS sont structurés un peu comme un algorithme génétique standard, mais diffère dans l'utilisation de la gestion de la population. Nous pouvons donc conclure que le paramètre Δ est le paramètre le plus important et que la gestion de la population joue un rôle important dans le processus MA | PM. À partir des solutions de cet échantillon sont combinées une recherche locale est lancée pour essayer d'améliorer les résultats obtenus.

Une procédure de gestion de la population est nécessaire pour maintenir une population de solutions de haute qualité. Cette procédure de gestion de population devrait permettre d'améliorer rapidement la qualité d'une solution produite par l'opérateur de croisement, sans le diversifier dans d'autres régions de l'espace de recherche. L'algorithme SS utilise des stratégies pour diversifier et intensifier la recherche qui a prouvé son efficacité dans des variétés de problèmes d'optimisation.

Nous pouvons également dire qu'inclure une solution heuristique de haute qualité peut aider l'AG à améliorer ses performances en réduisant la probabilité de sa convergence prématurée. De plus, le contrôle de la diversité a un grand effet sur les performances métaheuristiques.

4.8 Conclusion

Dans ce chapitre, nous avons présentés notre contribution représentée par l'adaptation de deux métaheuristiques à base de population de solutions. Ce sont des méthodes évolutionnaires appelées algorithme mémétique avec gestion de la population (Memetic Algorithm with Population Management – MA/PM) et recherche dispersée (Scatter Search).

Nous avons présenté une analyse de sensibilité de ces algorithmes, en étudiant l'influence de variation des paramètres internes de la métaheuristique sur ses performances, pour pouvoir tirer des informations sur les meilleures valeurs des paramètres de la méthode. Notre objectif était de déterminer la performance de l'algorithme mémétique avec la gestion de la population et la recherche de dispersion par rapport à l'algorithme génétique et mémétique pour un problème d'ordonnancement d'un système de production flexible. Les résultats montrent que les performances de l'algorithme mémétique avec la gestion de population et la recherche dispersée ont donné les meilleurs résultats en présence de pannes dans le système de production.

Ces approches ont été comparées avec d'autres méthodes évolutionnaires, pour pouvoir tirer une idée sur l'efficacité de chacune des méthodes utilisées. Ensuite nous avons donné les résultats et les interprétations obtenus des différentes simulations effectuées en ajoutant des pannes aux machines du système.

Des simulations ont été effectuées avec des variations, à partir de laquelle un certain nombre de conclusions peuvent être tirées

- ✓ L'étude de sensibilité de l'algorithme MA/PG et SS a permis de définir les paramètres qui améliorent les performances des métaheuristiques considérées.

- ✓ Les résultats obtenus ont tous montré que les métaheuristiques que nous avons étudiées ont donné de meilleures performances pour le taux de production, pour un système flexible de production saturé surtout pour de petites capacités de files d'attente
- ✓ Pour le temps de cycle, nous ne pouvons pas dire qu'une métaheuristique donne un temps de cycle meilleur que les autres pour toutes les capacités de files d'attente, mais on peut dire que l'algorithme mémétique avec gestion de population donne un temps de cycle raisonnable pour toutes les capacités des files d'attentes.
- ✓ Le taux d'utilisation des machines, l'algorithme mémétique avec gestion de population et la recherche dispersée en génerale sont les plus efficaces pour un taux de création de pièces supérieur à 1/20 tandis que pour les machines FH1 et FH2, les algorithmes génétiques donnent les meilleures utilisations pour ces deux machines

Chapitre 5

5 Développement d'algorithme pour l'ordonnancement de machines parallèles identiques

Ce chapitre présente le développement de l'algorithmeun pour l'ordonnancement de machines parallèles identiques. Dans ce chapitre nous donnant un état de l'art sur le systeme de production « machine parallèle identique »,puis on examine le problème d'ordonnancement de n jobs sur m machines parallèles identiques (IPM).L'algorithme développépour la résolution de problème d'ordonnancement de machines parallèles identiquespour minimiser le makespan, sera évalué et les résultats trouvés seront interprétés et comparés avec d'autres téchniques de résolution de ce type de problème d'ordonnancement existant déjà.

Sommaire

5.1 Introduction

Dans la vie d'aujourd'hui, le rôle de la fabrication devient essentiel pour enrichir la société grâce à un processus de production approprié en transformant les matières premières, les composants ou les pièces en produits finis répondant aux attentes des clients, notamment fonctionnelles, respectueuses de l'environnement et fiables. L'objectif général de la fabrication est de créer les produits avec des attentes réduites et un coût minimum.

La principale responsabilité de l'environnement de fabrication est d'établir les niveaux de priorité et les objectifs impliqués ainsi que le suivi des performances. Un autre facteur important est l'utilisation des ressources disponibles telles que les ressources disponibles en main-d'œuvre, technologie, capital, efforts, énergie, matériaux, environnement, information pour atteindre les objectifs définis par l'ingénieur industriel de fabrication. L'ordonnancement est un processus de prise de décision principalement lié à l'allocation des opérations sur les machines d'une manière telle que les mesures de performance requises, telles que Makspan, Maximum, Lateness, Total weighted completion ……

Les procédures d'ordonnancement disponibles aujourd'hui sont non seulement assez réalistes, mais possèdent également une capacité de génération de solution rapide. Il trouve des applications répandues à la fois dans la fabrication et industries de services. Pour le processus de prise de décision, l'ordonnancement et le séquençage existent à plusieurs niveaux selon plusieurs chercheurs (**Baker 1974, Browne, Harhen et Shivnan 1988, Muchnik 1992 et Morton et Pentico 1993**).

Ces niveaux sont les suivants : ordonnancement à long terme, ordonnancement à moyen terme, ordonnancement à court terme, ordonnancement prédictive et ordonnancement de la réaction / contrôle. Les environnements de séquençage et d'ordonnancement sont classés en quatre types selon Convay, Maxwell et Miller **[Conway et al. 1967]**. Selon le type de machine, le nombre des travaux et des machines et les types d'affectations, les environnements de séquençage et d'ordonnancement sont classés comme suit:

5.2 Problème d'ordonnancement de machine unique

Un seul atelier d'usinage peut être grossièrement classé en deux types principaux :

- ✓ Avec des processeurs uniques, en exécutant une seule tâche dans la même machine.
- ✓ Avec des processeurs parallèles, en exécutant plusieurs tâches simultanément sur la même machine.

La classification de l'environnement de planification d'une seule machine est illustrée à la Figure 5.1.

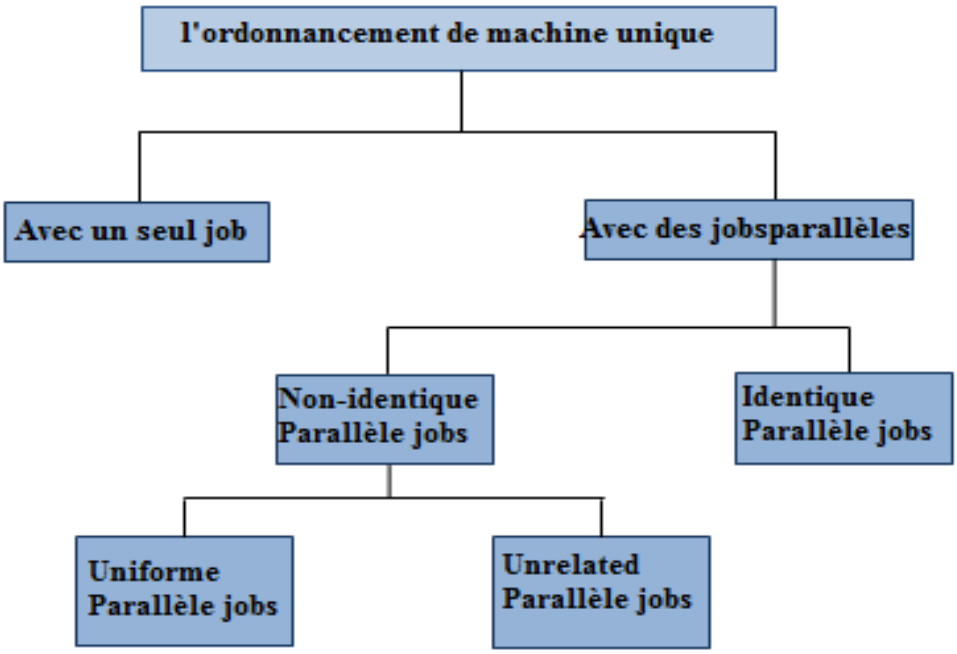


Figure 5.1. Classification de l'environnement de planification d'une seule machine**.**

La planification d'une seule machine avec un seul processeur consiste en une seule machine pour traiter n tâches. L'objectif de ce problème est d'optimiser la mesure de performance donnée en planifiant n tâches sur la machine unique. Ces jobs peuvent être indépendants ou dépendants, selon la condition. Lorsque les temps de traitements des jobs sont considérés comme indépendants de la séquence de processus des travaux, le problème d'ordonnancement est appelé problème d'ordonnancement de machine unique avec des travaux indépendants, dans le cas contraire, il s'agit d'un problème d'ordonnancement de machine unique avec des travaux dépendants.

Si le nombre de machines est supérieur à un, et si t_{ij} est le temps de traitement du travail j sur la machine i, pour i = 1,2,3, ..., m et j = 1,2,3, ..., n, alors les problèmes d'ordonnancement de la machine parallèle peuvent être définis en utilisant les temps de traitement, comme indiqué dans le Tableau 5.1

Tableau 5-1 Comparaison de l'environnement de planification des processeurs parallèles

Identical Parallel Processor	Uniform Parallel Processor	Unrelated Parallel Processor
Pij = P1j pour tout i et j. Les temps de traitement sont identiques et identiques dans toutes les machines	$P_{ij} = P_{1j} / S_i$ pour tout i et j, où S est la vitesse de la machine i et P_{1j} est le temps de traitement du travail j sur la machine 1. On suppose que les vitesses S_1, S_2, S_3, ..., S pour les machines parallèles 1,2,3, ..., m respectivement avec la relation $S_1 < S_2 < S_3 < ... < S_m$. Les temps de traitement sur les machines parallèles seront dans le rapport 1 / S_1: 1 / S_2: 1 / S_3: ...: 1 / S_m	P_{ij} est arbitraire pour tout i et j. Il n'y aura aucune relation entre les temps de traitement du travail sur les machines parallèles .

5.3 Problème d'ordonnancement machines parallèles

L'évolution du changement continue, des demandes des clients et la nécessité de fournir des produits de meilleure qualité sont considérées comme des facteurs déterminants pour améliorer les performances des systèmes de production. Le système de production doit posséder une capacité d'adaptation nécessaire pour atteindre les exigences nécessaires le plus rapidement possible. Au cours des trois dernières décennies, ces systèmes ont connu une croissance considérable en introduisant et en mettant en œuvre divers concepts managériaux et divers outils qui contrôlent la procédure de production des différents types d'environnements de production et coordonnent les communications nécessaires entre les différents niveaux d'organisation en utilisant les technologies de l'information avancées.

Avec la complexité croissante dans les environnements de systèmes de production modernes, la plupart des problèmes d'ordonnancement se prouvent être NP-Hard. C'est-à-dire que les besoins en calcul augmentent exponentiellement en fonction de la taille du problème **[Garey 1979].** En plus de cela, l'espace de solution pour les problèmes d'ordonnancement s'avère difficile à comprendre pour concevoir la technique permettant d'atteindre la solution optimale dans l'espace de solution complexe. Le problème exige donc un effort de recherche important pour atteindre son objectif. Selon leurs algorithmes traditionnels, les techniques d'optimisation ont joué un rôle important dans la résolution de problèmes d'ordonnancement complexes.

Les algorithmes d'optimisation inclurent différentes procédures énumératives et des techniques de programmation mathématique telles que: la programmation linéaire (Linear Programming), la programmation entière (Integer Programming), la programmation de but (Goal Programming), le modèle de transport (Transportation model), l'analyse de réseau (Network Analysis), la programmation dynamique (Dynamic programming), la théorie des jeux

(Games theory), le modèles de séquençage (Sequencing Models) et la programmation géométrique (Geometric Programming).

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable, ces méthodes rencontrent généralement des difficultés face aux applications de taille importante car le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème.

En général, les techniques d'approximation incluent la technique d'énumération implicite (BBA, méthode de décomposition, LR, règles de priorité, heuristiques), les algorithmes de recherche locale (recherche ITS, algorithme de recuit simulé SAA, voisinage variable). Algorithme de recherche (VNS et Tabu Search TS) et Algorithmes évolutionnaires (Algorithme génétique GA, Optimisation d'essaim de particules PSO, Algorithme de recherched'harmonieHSA). Parmi eux, certaines de ces techniques sont robustes et donnent de bons résultats, leurs performances sont satisfaisantes tant que les objectifs définis dans les systèmes de production et les conditions de fonctionnement restent les mêmes **[Braker et al. 1988]**

On utilise un schéma de classification connu$\alpha|\beta|\gamma$ proposé par Graham et al **[Graham et al 1978]**, notre problème d'ordonnancement de machine parallèle identique pour minimiser le makespan est noté dans la littérature comme P ‖ C max où P représente les machines parallèles identiques et C_{max} indique le temps d'achèvement maximum ou le makespan. Chaque machine ne peut exécuter qu'une tâche à la fois et chaque tâche ne peut être exécutée que sur une machine à la fois. Le problème d'équilibrage de la charge de la machine est défini comme suit : où n tâches doivent être affectées à m machines parallèles identiques. Chaque tâche j est entièrement caractérisée par son temps de traitement non négatif p.

En donnant une attribution de tâches, nous pouvons définir le concept de la charge d'une machine comme la somme des temps de traitement des travaux qui lui sont affectés **[Mokotof 2001]**. La machine recouvre le problème d'attribution des tâches et maximisation de la charge à partir du dispositif le moins chargé.

Le problème de l'équilibre de la charge sur les machines parallèles en minimisant le makspan a été largement étudié dans la littérature **[Mokotof 2001]**. Comme ce problème est généralement NP-difficile, les meilleurs algorithmes proposés sont efficaces pour résoudre de petites instances.

Pour obtenir une solution optimale pour ce type de problème complexe en temps de calcul raisonnable, l'utilisation d'approches traditionnelles est extrêmement difficile **[Garey 1978]**. La complexité, en général, croît exponentiellement avec le nombre de machines M, ce qui rend difficile la résolution de ces problèmes par des techniques conventionnelles. Cette solution ne convient donc pas à la résolution de problèmes complexes. En raison de l'apparition croissante de ce problème en informatique, outre leur intransigeance, les problèmes d'ordonnancement demeurent parmi les problèmes prioritaires pour les chercheurs. Par conséquent, l'utilisation des heuristiques estdevenue inévitables pour résoudre les problèmes d'ordonnancement dans le monde réel surtout avec les résultats trouvés et qui ont été très satisfaisants dans un temps de calcul très raisonnable **[Ethel 2001]**.

Plusieurs algorithmes de temps polynomiaux ont été proposés pour trouver la solution à ces problèmes. Un algorithme bien étudié pour ce problème est le temps de traitement le plus long LPT (the Longest Processing Time). L'algorithme LPT consistant à trier les tâches dans l'ordre décroissant de leur temps d'exécution a été étudié par Graham **[Graham et al. 1978]**. Il possède un rapport d'approximation de dans l'intervalle [4/3− 1/3] de m, où m est le nombre de machines. Graham exhibe un cas limite où ce rapport est atteint. On voit ici que plus le nombre de machines augmente, plus on se rapproche du pire ratio d'approximation possible, qui est 4/3. Dans le pire des cas, l'algorithme LPT peut fournir une solution 33% plus longue que l'optimum. La règle de la LPT a fait l'objet d'une attention particulière pour la résolution de critère unique de problèmes makespan **[Kedia 1971],** de même que l'heuristique Multifit donnent des résultats satisfaisants cette méthode commence par fixer une valeur de temps de fabrication total M comprise entre la borne de McNaughton **[McNaughton 1959]** et $4M/3$ en notant p_j temps de traitement de la tache j, avec $1 \leq J \leq n$ et m le nombre de machines parallèles la borne est calculée comme suit :

$$\bar{M} = max\left[\sum_{j=1}^{n}\frac{p_j}{m}, max_j\{p_j\}\right]$$

Cette dernière affecte les opérations, dans l'ordre décroissant de leur durée, à la première machine capable de la terminer avant une certaine valeur cible.

Mokotoff et al **[Mokotoff et al 2001]** comparent le degré de performance des autres heuristiques existantes pour l'ordonnancement sur des machines parallèles identiques pour minimiser le makspen et présente une heuristique basée sur les procédures de listes cette heuristique définit une variable de décision qui dépend des temps d'exécution des taches. Quand

cette variable dépasse une certaine limite on applique soit l'algorithme Multi fit soit la règle de LPT.

5.4 Un état de l'art sur le problème d'ordonnancement de MPIMM

Dans la littérature, le problème de la charge des machines parallèles identiques a été associé à différents critères d'ordonnancement de différentes manières, même en considérant le déséquilibre de la charge de tâches comme une limitation ou comme un but. Par exemple, l'approche des réseaux neuronaux a été présentée pour la première fois il y a 60 ans, mais le modèle le plus représentatif a été proposé par **[Akyol et al. 2006]** pour résoudre les problèmes d'optimisation combinatoire avec d'autres modèles typiques. Ils ont proposé un réseau neuronal dynamique qui utilise différents paramètres de pénalité du temps pour la solution du problème. La performance du réseau proposé est évaluée sur un exemple de problème d'ordonnancement sur des machines parallèles identiques.Les expériences de simulation ont prouvé que le réseau proposé générait des solutions réalisables pour tous les ensembles de données et qu'il avait moins de makespan comparativement au LPT. HaoShao et al **[HaoShao et al.]** ont présenté l'algorithme des réseaux neuronaux (NNA) pour minimiser le makespan sur les machines parallèles avec des tailles dejobs non identiques,ils ont introduit une matrice de pondération principale et de nouvelles méthodes de codage. Le résultat était particulièrement efficace pour les instances à grande échelle par rapport à d'autres heuristiques.

Raghavendra et al. **[Raghavendra et al. 2007]** ont proposé une procédure heuristique basée sur un algorithme génétique avec des règles SPT et LPT pour minimiser le déséquilibre de la charge de job entre les machines et réduire le makespan. Les auteurs ont trouvé les meilleurs résultats en termes d'exactitude et de rapidité de solution par rapport aux stratégies proposées dans leur premier article **[Raghavendra et al. 2010]**. La même heuristique basée sur la génétique a été comparée à d'autres approches approximatives proposées dans la littérature. Raghavendra et Murthy ont présenté une étude comparative entre différents exemples de tests pour illustrer l'efficacité de leur algorithme. Les différentes heuristiques proposées par Heinrich **[Heinrich et al. 1995]** et les algorithmes de recuit simulés et génétiques proposés par Liu et Wu. **[Liu et Wu 1999],** Hu **[Hu 2004]** ont utilisés une grande procédure de contribution marginale (LMC) et une heuristique de temps de traitement plus court (SPT) pour résoudre les problèmes d'ordonnancement des tâches et d'affectation des jobs pour les modèles de machines parallèles, qui ont conclu que ces heuristiques produisaient les mêmes résultats. Chaudhry et al **[Chaudhry et al. 2010]** ont présenté un algorithme générique (GA) basé sur un tableur à usage

général afin de minimiser le makspan d'un ensemble de tâches pour des machines parallèles identiques et l'affectation de jobs aux machines. L'implémentation du tableur d'algorithme génétique (The spreadsheet GA) est facile à mettre en œuvre pour répondre aux particularités de tout environnement. Hu **[Hu 2004]** a conclu que la mise en œuvre du tableur GA fournit de meilleures solutions que les stratégies proposées dans une étude antérieure. Dans le cas de périodes de non-disponibilité planifiées dans des tâches réutilisables, **Hu a** considéré que le problème de minimisation de makespan pour l'ordonnancement de machines parallèle est plus compliqué. Hashemian et al **[Hashemian et al. 2012]** formulaient comme un modèle de programmation linéaire à nombres entiers mixtes et résolvaient de manière optimale en utilisant CPLEX pour des problèmes de petite taille à moyennement grande avec plusieurs contraintes de disponibilité sur toutes les machines. Ils ont proposé aussi l'algorithme de dénombrement en utilisant l'ordre lexicographique pour résoudre des problèmes à grande échelle. Pour réduire l'espace de recherche de l'algorithme et améliorer ses performances plusieurs lemmes sont conçus.

Plusieurs expériences ont été menées qui ont montré les améliorations de légitimité et d'exécution garanties par l'affichage **MILP** et le nouvel algorithme d'énumération. Abey Kuruvilla et al **[Abey Kuruvilla et al 2015]** ont proposé un algorithme heuristique qui utilise itérativement les approches LPT et MF sur divers ensembles de tâches et de machines construits en utilisant la solution actuelle pour résoudre un problème d'ordonnancement multiprocesseur classique dans le but de minimiser le makspan. Les résultats computationnels indiquent que l'algorithme proposé est très compétitif par rapport aux algorithmes constructifs bien connus pour un grand nombre d'instances de référence.

Ranjbar et al. **[Ranjbar et al.]** ont développé deux algorithmes de branch and bound pour trouver une solution optimale et le but est de trouver un programme robuste qui maximise le niveau de service à la clientèle, qui est la probabilité que le makespan ne dépasse pas la date d'échéance. Xu et Nagi **[Xu et Nagi 2005]** ont pris en compte les problèmes IPMS dans le but de minimiser la somme des temps d'achèvement pondérés et de faire de makespan. Ils ont formulé le problème en tant que problème de programmation d'entier mixte basé sur les propriétés d'une planification optimale.

Damodaran et Gallego **[Damodaran et Gallego 1979]** ont adressé SAA(Simulated annealing algorithme) pour minimiser le makespan sur un groupe de machines de traitement par lots identiques disposées en parallèle où chaque travail avait un temps de traitement

arbitraire, une taille non identique et un temps de préparation non nul. **Yeh et al. [Yeh et al. 2014]** ont abordé l'ordonnancement parallèle des machines avec des effets d'apprentissage et proposé un algorithme de recuit simulé pour minimiser le makespan. Tseng et Lin **[Tseng et Lin 2009]** ont proposé un algorithme génétique hybride pour le problème de l'ordonnancement de l'atelier de flowshop afin de minimiser le makespan.

Y.Ouazene et al **[Y.Ouazene et al. 2016]** ont étudié les différents critères proposés dans la littérature dans le cas de ressources parallèles identiques. Sur la base de cette étude, les auteurs ont établi que le problème d'équilibrage de la charge de travail peut être formulé comme un problème de minimisation de la différence entre la charge de travail de la machine encombrée et la charge de travail de la machine la plus rapide. Yousef German **[Yousef Germanet al. 2016]** a adopté le modèle de Programmation Linéaire Entier (MLP) pour formuler le problème de machine parallèle dans le but de minimiser le déséquilibre relatif total et il a utilisé l'algorithme de temps de traitement le plus long LPT pour trouver la solution initiale. Ils ont suggéré un algorithme pour améliorer la solution initiale pour la résolution approchée du problème. Les résultats ont démontré que la modélisation mathématique et les algorithmes sont des outils puissants et plus efficaces pour ce type de problèmes, par rapport aux méthodes traditionnelles.

5.5 Formulation du problème

Ce chapitre examine le problème d'ordonnancement de tâches indépendantes sur des machines parallèles identiques. Les répartitions des tâches vers les machines et les séquences des tâches pour chaque machine sont les deux problèmes essentiels dans le problème classique d'ordonnancement des machines parallèles **[Chaudhry et al. 2010]**. L'objectif d'ordonnancement des machines est d'assigner des tâches aux machines suivant la fonction objective correspondante afin de minimiser les temps de fonctionnement et d'augmenter la productivité **[Akyol 2006]**.

Le problème peut être décrit comme suit : Nous avons n jobs, on doit déterminer quand chaque opération doit commencer et finir sur chacune des m machines parallèles identiques et comment utiliser les ressources disponibles de manière efficace pour exécuter (assigner) des jobs ou des tâches sur des machines pendant un temps de traitement fixe sans préemption.

La représentation schématique d'un environnement de traitement parallèle identique typique impliquant trois machines identiques, nécessaire pour produire huit travaux, est illustrée à la figure 5.2.

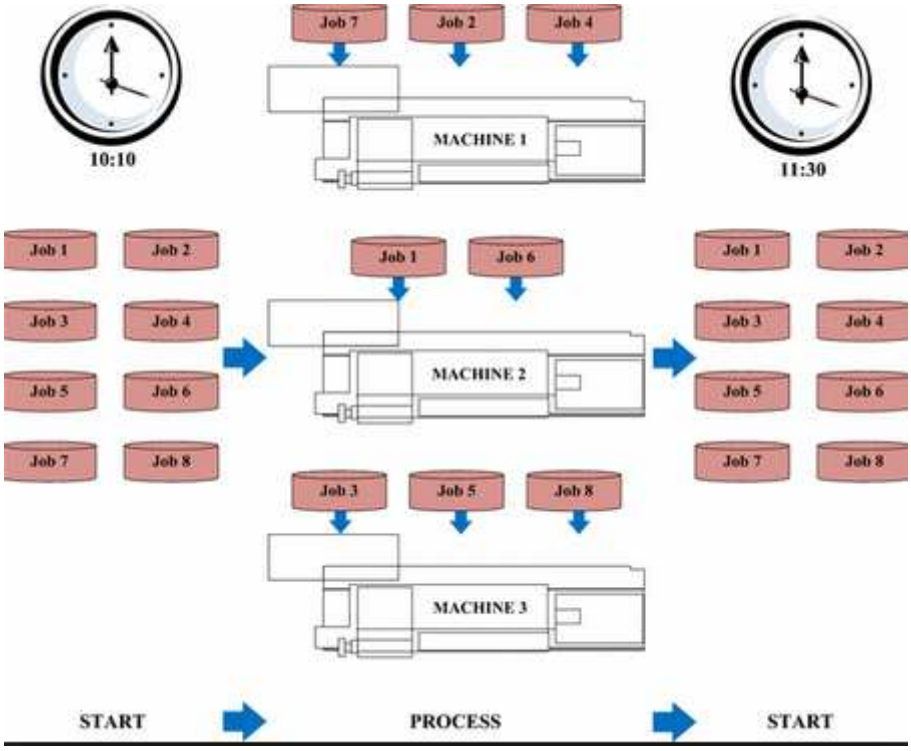


Figure 5.2. Structure du processus de machine parallèle**.**

L'objectif est de trouver une allocation appropriée des jobs aux machines qui optimiserai le critère de performance représenté dans notre travail par la minimisation du makespan qui est défini comme le temps total d'achèvement de toutes les tâches, pour un problème d'ordonnancement déterministe classique de machines parallèles identiques. Les résultats trouvés de l'heuristique MMIPMH seront comparées à ceux de l'approche heuristique LPT, Hu **[Hu 2004]**. Nous utiliserons cette notation dans ce qui suit :

n: nombre des tâches.

m: nombre de machines.

J: tâche (Job)j, j Є N = {1, ..., n}

Mi: machine i, i Є M = {1, ..., m}

P_j: temps d'exécution du job J.

C_i: temps total d'exécution des jobssur la machine i.

C_{max}: makespan, le délai d'exécution maximal de tous les travaux

$$C_{max} = max\{c_1, c_2, c_3, \dots, c_m\}$$

C_{min}: Minimum de charges des Jobs.

X_{ij}: Coefficient d'affectation du job j sur la machine i (X_{ij} =1 si le job j est affecté à la machine i, 0 si non).

Le problème en question est noté : P‖C_{max}, où le premier champ α=P qui représente le problème

d'ordonnancement de machines parallèles identiques, le deuxième champ β est vide cela veut dire qu'il n'y a pas de contraintes, le troisième champ γ nous donne la fonction objectif à optimiser représenté ici représentée par la minimisation du makespan (C_{max}).

La fonction objectif :

$$\text{Min } C_{max} \tag{5.1}$$

L'équation (5.1) montre la fonction objective Cmax makespan, qui doit être minimisé.

Nous avons les contraintes suivantes ;

$$C_{max} - \sum_{j=1}^{n} P_j \, x_{ij} \geq 0 \quad j = 1, \dots, n \; ; \; i = 1, \dots, m. \tag{5.2}$$

Cette contrainte assure que la charge sur n'importe quelle machine est égale ou inférieure à C_{max}.

$$\sum_{i=0}^{m} x_{ij} = 1 \quad i = 1, \dots\dots, m \; ; j = 1, \dots, n. \tag{5.3}$$

La contrainte (5.3) montre que chaque tâche est affectée à une seule machine.

5.6 Présentation et explication des algorithmes

Dans ce travail, la solution peut être obtenue en deux étapes : L'algorithme LPT est utilisé pour trouver (générer) la solution initiale, tandis que l'algorithme développé est utilisé pour améliorer la solution initiale

La règle LPT de Graham **[Graham et al. 1979]**, l'une des heuristiques les plus populaires, consiste à répartir les tâches entre des machines identiques de manière à minimiser le makespan. Cette règle place toujours les tâches les plus petites vers la fin de la planification, de sorte que $P_1 \geq P_2 \geq P_3 \geq \cdots \geq P_n$ puis assigne successivement les travaux à la machine la moins chargée.

Selon l'algorithme (LPT), chaque fois que l'une des machines (m) est libérée, le job sélectionné sera celui qui a le plus long temps d'exécution parmi les jobs en attentes. Le prochain job j sera planifié sur la machine i selon l'équation :

$$C_i = C_i + P_j, (i = 1, \dots, m \; ; \; j = 1, \dots, n) \tag{5.4}$$

Présentation de l'algorithme LPT :

L'algorithme LPT est résumé comme suit :

- **Étape 1 :** C_i=0 au départ pour toutes les machines.
- **Étape 2 :** Trier les P_j des tâches selon la règle LPT et les mettre dans une liste N (le premier job de la liste sera celui qui a le temps d'exécution le plus long).

- **Étape 3 :** Affecter le job j (le job sélectionné sera le premier de la liste de l'étape1) à la machine libre i.
- **Etape 4 :** faire le calcul de C_i selon (5).
- **Étape 5 :** Supprimer le job j sélectionné de la liste N.
- **Étape 6 :** Si j = n (Tous les jobs sont affectés) aller à l'étape suivante (Étape 7), sinon aller à l'étape 3.
- **Étape 7** : Calculer :$C_{max}^{LPT} = max\{C_{i,}, i = 1, \dots, m\}$ (5.5)

Présentation de l'algorithme MMIPMH :

L'algorithme, d'équilibrage de charge MMIPMH, résout le problème d'ordonnancement P ‖ C_{max} en utilisant la règle de priorité LPT.
Cette méthode est inspirée du réseau de neurones de Hopfild que nous avons simplifié par l'élimination de toutes les charges des machines qui ne sont pas utiles pour diminuer le temps de calcul.

Exemple explicatif1 :

Pour un système de machines parallèles identiques qui contient 5 jobs et 3 machines

- On ne va pas prendre le cas où tous les jobs sont affectés à la machin3: $\begin{vmatrix} m_1 \\ m_2 \\ m_3 \end{vmatrix} = \begin{vmatrix} 00000 \\ 00000 \\ 11111 \end{vmatrix}$
- On ne va pas prendre le cas où aucun job n'est affecté à la machine 1: $\begin{vmatrix} m_1 \\ m_2 \\ m_3 \end{vmatrix} = \begin{vmatrix} 00000 \\ 00011 \\ 11100 \end{vmatrix}$
- On ne va pas prendre le cas où un job est affecté à plus d'une machine. $\begin{vmatrix} m_1 \\ m_2 \\ m_3 \end{vmatrix} = \begin{vmatrix} 00100 \\ 10000 \\ 11010 \end{vmatrix}$

Algorithme MMIPMH :

Avant de commencer l'exécution des étapes de cet algorithme, il faut d'abord calculer $C_{min}^{LPT}, C_{max}^{LPT}$ à partir de l'algorithme LPT présenté précédemment.

- **Etape1 :** 1 <= le nombre de jobs exécutés sur chaque machine<=n-m + 1 (5.7)
- **Etape2 :** Calcul de :$\propto = C_{max}^{LPT} - C_{min}^{LPT}$
- **Etape 3 :** Calcul deC_{max}^{*}

  $C_{max}^{*} = C_{min}^{LPT} + \propto/2$ (5.8)
- **Etape 4 :** Chercher le C_{max} qui représente le minimum des $C_i$et appartenant à l'intervalle de (9) pour les différentes séquences de jobs possibles.

  $C_{min}^{LPT} < C_{max} < C_{max}^{*}$ (5.9)

La figure 5.3 illustre comment calculer α :

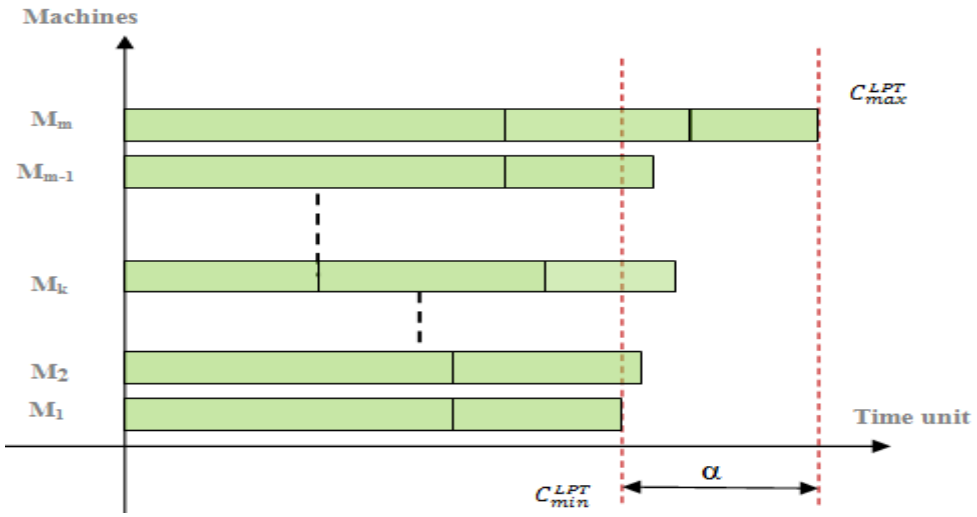


Figure 5.3. Illustration de la valeur α

Exemple explicatif 2 :

Soit le tableau suivant contenant 9 jobs avec leur temps d'exécution :

Job (j)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
P(j)	**4**	**5**	**7**	**4**	**6**	**7**	**6**	**4**	**5**

Dans cet exemple on va utiliser l'algorithme LPT pour déterminer les valeurs$C_{min}^{LPT} et C_{max}^{LPT}$.

- La liste N des jobs après utilisation de LPT : N= {7, 7, 6, 6, 5, 5, 4, 4, 4}.
- Le calcul de $C_{min}^{LPT} et C_{max}^{LPT}$ a donné :

  La charge sur les machines :

  $C_1 = P_1 + P_8 + P_9 = 7 + 4 + 4 = 15$

  $C_2 = P_2 + P_7 = 7 + 4 = 11$

  $C_3 = P_3 + P_6 = 6 + 5 = 11$

  $C_4 = P_4 + P_5 = 6 + 5 = 11$

$$C_{max} = max\{C_i : i = 1,2,3,4\} \rightarrow C_{max} = max\{C_1, C_2, C_3, C_4\}$$

$$C_{max} = max\{11,11,11,15\} \rightarrow \boldsymbol{C_{min}^{LPT} = 11} \; ; \; \boldsymbol{C_{max}^{LPT} = 15}$$

- Utilisation de l'algorithme **MMIPMH** pour améliorer la solution initiale trouvée par l'algorithme LPT.
- On calcul la paramètre α :

$$\propto = C_{max}^{LPT} - C_{min}^{LPT} = 15 - 11 = 4$$

- On Calcul C_{max}^{*}:

$$C^*_{max} = C^{LPT}_{min} + \propto \frac{1}{2} = 11 + \frac{4}{2} = 13$$

$$11 \leq C_{max} \leq 13$$

- Après on va l'algorithme chercher la valeur C_{max}appartenant à l'intervalle [11,13]. La figure 5.4 montre le diagramme de Gantt pour la solution et la charge sur les quatre machines pour les deux algorithmes **LPT** et **MMIPMH**.

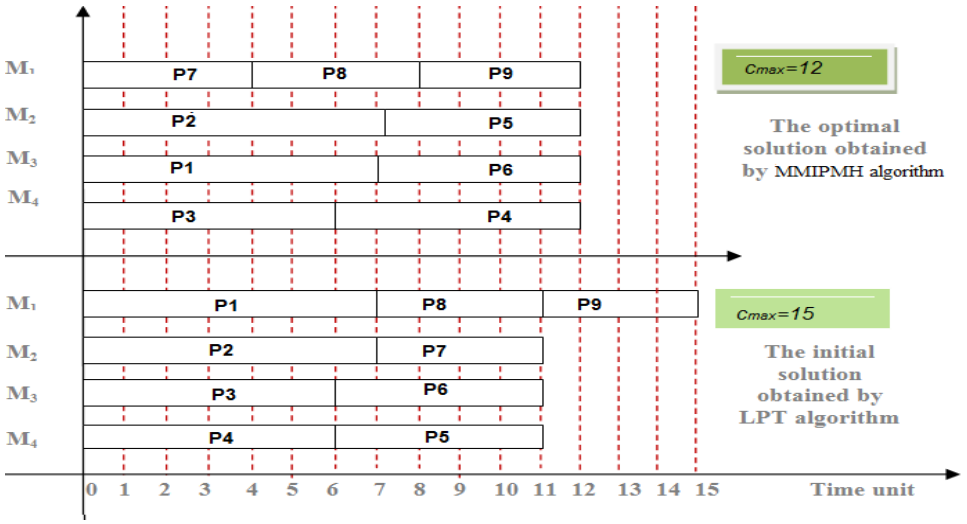


Figure 5.4. Diagramme de Gantt pour quatre machines et neuf emplois

5.7 Programme informatique

Les algorithmes développés ont été codés en Delphi et exécutés sur un ordinateur doté d'un processeur Intel i5 avec une fréquence cadencée à 2,30 GHz et une mémoire RAM de 6,00 Go. Sous le système d'exploitation Windows 7 (64bit). Le programme est conçu pour résoudre diverses instances du problème d'ordonnancement $P||C_{max}$, ces instances dépendent du nombre de machines (m) et du nombre de tâches (n).

De nombreuses instances avec différents niveaux de difficultés ont été réalisées pour évaluer les performances de l'algorithme MMIPMH. Pour évaluer les performances de notre algorithme, nous allons fait une comparaison entre celui-là et l'algorithme LPT précédemment utilisé pour résoudre le même problème.

5.7.1 Interface du Programme informatique

Dans cette section, nous donnons un aperçu de l'interface graphique de notre application.

Le bouton **LPT** nous permet de calculer le C_{max} , le bouton **Clear** nous permet d'effacer les listes N des temps d'exécution (P_j) et la liste des charges sur les machines (C_i). Elle représente l'algorithme LPT à travers lequel on peut extraire directement C_{max}^{LPT}et C_{min}^{LPT}.

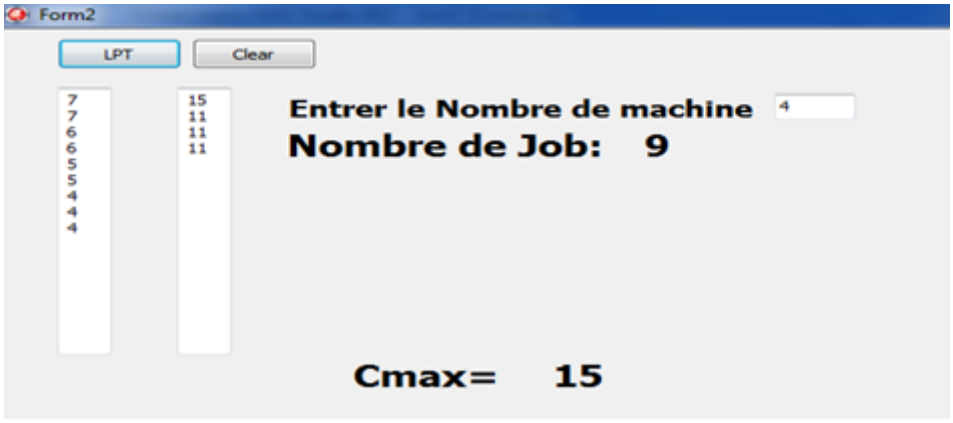


Figure 5.5. Interface pour l'algorithme LPT

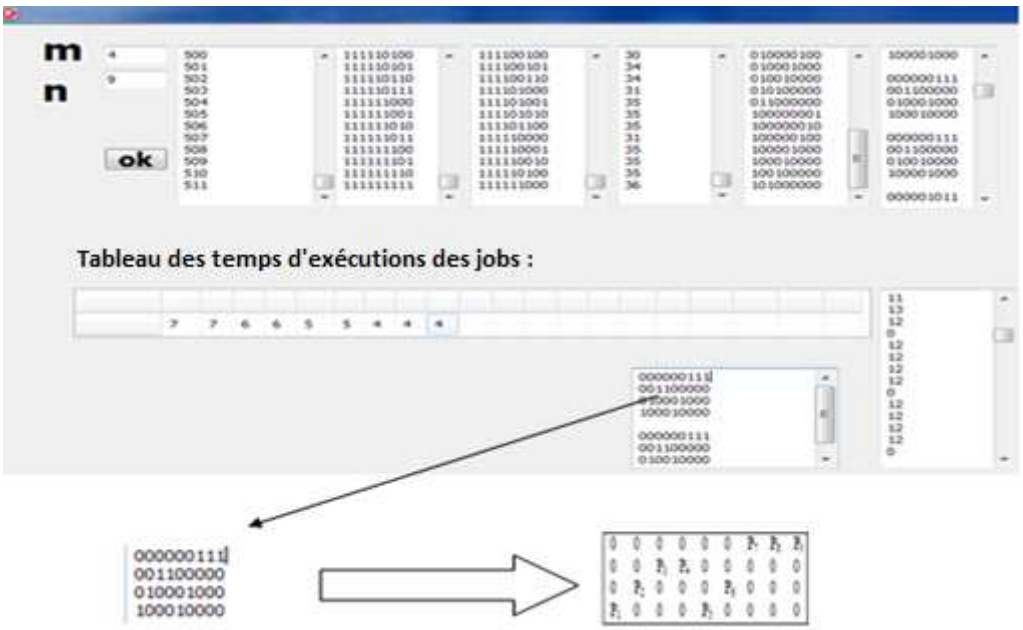


Figure 5.6. Interface pour l'heuristique MMIPMH

La deuxième interface représente l'algorithme **MMIPMH** qui cherche la meilleure séquence pour trouver le C_{max} parmi un ensemble de séquences possibles.

5.8 Algorithme MMIPMH pour le ré-ordonnancement

Dans cette partie du chapitre nous allons faire un ré-ordonnancement en utilisant l'algorithme MMIPMH. On utilise le ré-ordonnancement des jobs dans le cas où le nombre de

machine dans système de production est grand, afin d'améliorer le C_{max} qu'on trouve avec l'algorithme MMIPMH sans le ré-ordonnancement.

- **Etape 1 :** On commence par trouver les C_{max}^{LPT}et C_{min}^{LPT} , par l'application de de la règle LPT.
- **Etape 2 :** On supprime la séquence (la liste des jobs) de la machine qui contient le C_{min}^{LPT} .
- **Etape 3 :** On applique l'algorithme MMIPMH pour ordonnancer les jobs qui reste et trouver le C_{max}.

- **Exemple explicatif 3 :**

Soit un système contenant 4machines parallèles identiques et 12 jobs, chaque machine ne peut traiter qu'un seul job à la fois. Les temps d'exécution des jobs sont donnés comme suit :

job (j)	1	2	3	4	5	6	7	8	9	10	11	12
P(j)	3	6	3	15	6	7	9	7	8	9	4	5

- **Utilisation de l'algorithme LPT** :

Après utilisation de LPT On trouve :

$C_1= P_4 + P_6 + P_{10} + P_{12}= 8+7+4+3 = 22$

$C_2= P_3 + P_5+ P_{11}= 9+ 7+3 =19$

$C_3= P_2 + P_8 + P_9= 9+ 6+5 = 20$

$C_4=P_1+P_7=15+6=21$

$$C_{max} = max\{C_i : i = 1,2,3,4\}$$

$$C_{max} = max\{C_1, C_2, C_3, C_4\}$$

$$C_{max} = max\{22,19,20,21\} \rightarrow C_{max}^{LPT} = 22 \; ; C_{min}^{LPT} = 19$$

- **En utilisant l'algorithme MMIPMH :**

Pour trouver l'intervalle de recherche du C_{max}, on va appliquer l'algorithme MMIPMH :

On calcule le paramètre α : $\propto = C_{max}^{LPT} - C_{min}^{LPT} = 22 - 19 = 3$

Donc:$C_{max}^{*} = C_{min}^{LPT} + \propto \frac{1}{2} = 19 + \frac{3}{2} = 21{,}5$

Puisque on ne peut pas diviser le temps d'exécution des jobs, on va prendre la partie entière de C_{max}^{*}. $Donc : C_{max}^{*}$=21.

Cela veut dire que l'algorithme va chercherC_{max} dans l'intervalle [19,21].

$$19 \leq C_{max} \leq 21.$$

Les résultats trouvés sont illustrés sur la figure 5.7.

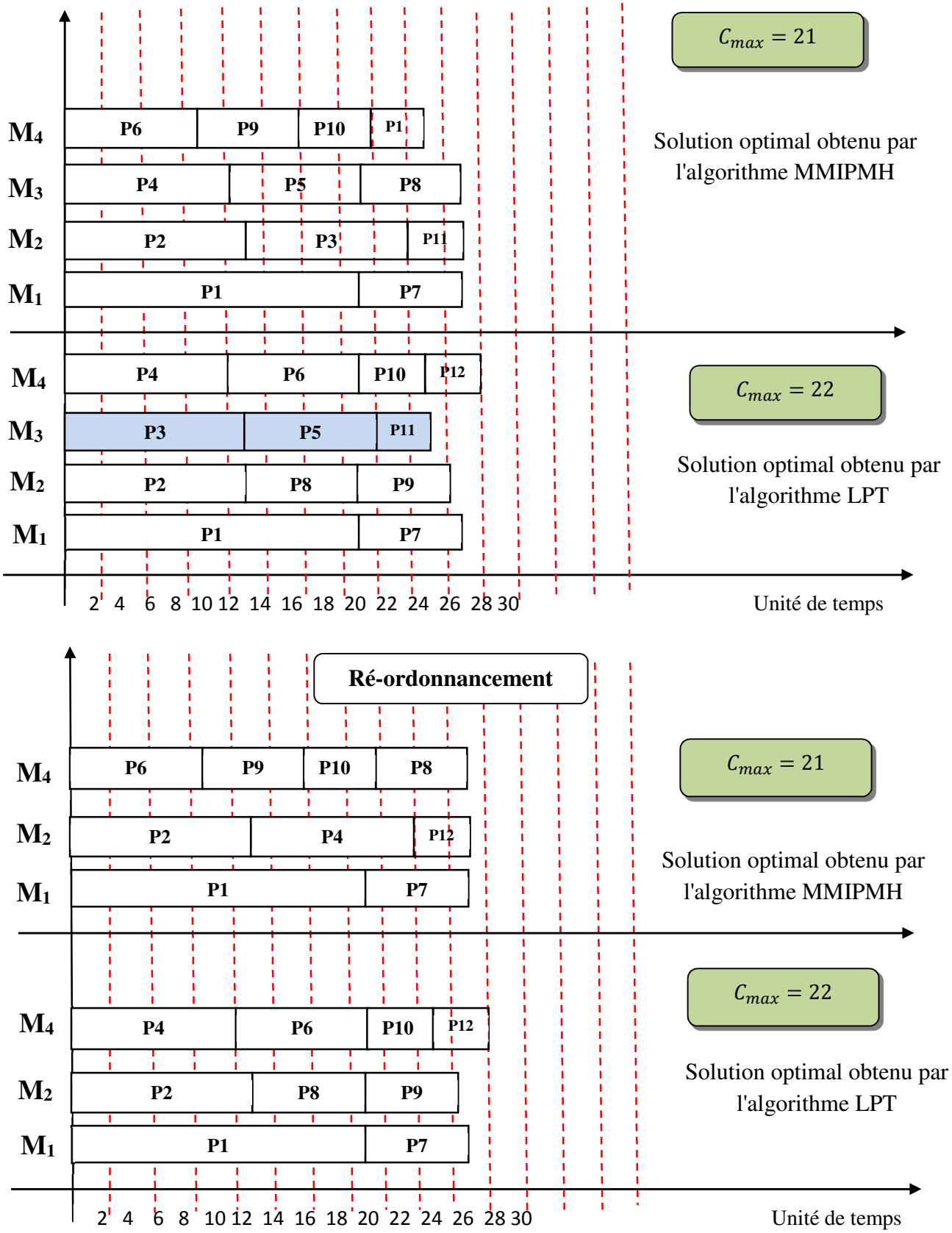


Figure 5.7. Diagramme de Gantt pour 4 machines et 9jobs sans et avec ré-ordonnancement**.**

- **Exemple explicatif 4 :**

Soit un système contenant 5 machines parallèles identiques et 15 jobs, chaque machine ne peut traiter qu'un seul job à la fois. Les temps d'exécution des jobs sont donnés comme suit :

Pj= {4,4,4,5,5,6,6,7,7,8,9,9,11,15,19}.

Soit le tableau suivant contenant 9 jobs avec leur temps d'exécution :

Job (j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(j)	15	5	4	4	6	7	6	4	5	8	11	7	9	9	19

- **Utilisation de l'algorithme LPT** :

Après utilisation de LPT On trouve :

C_1= P_1 + P_{13} = 19+4= 23

C_2= P_2 + P_9+ P_{14}= 15+ 6+4 =25

C_3= P_3+ P_7 + P_{11}= 11+7+5 = 23

C_4=P_4+P_8+P_{10}=9+7+6=22

C_5=P_5+P_6+P_{12}+P_{13}=26

$$C_{max} = max\{C_i : i = 1,2,3,4,5\}$$

$$C_{max} = max\{C_1, C_2, C_3, C_4, C_5\}$$

$$C_{max} = max\{22,25,23,22,26\} \rightarrow C_{max}^{LPT} = 26\ ;\ C_{min}^{LPT} = 22$$

- **En utilisant l'algorithme MMIPMH :**

Pour trouver l'intervalle de recherche du C_{max}, on va appliquer l'algorithme MMIPMH :

On calcule le paramètre α : $\propto = C_{max}^{LPT} - C_{min}^{LPT} = 26 - 22 = 4$

Donc : $C_{max}^{*} = C_{min}^{LPT} + \propto \frac{1}{2} = 22 + \frac{4}{2} = 24$

Donc : C_{max}^{*}=24.

Cela veut dire que l'algorithme va chercherC_{max} dans l'intervalle [22,24].

$$22 \leq C_{max} \leq 24$$

La figure 5.8 nous montre les résultats trouvés avec les algorithmes LPT et MMIPMH sans ré-ordonnancement où on remarque que la valeur du C_{max} sera trouvée dans l'intervalle [22,24]. Après l'utilisation de l'algorithme MMIPMH pour le ré-ordonnancement on remarque que la valeur de C_{max}sera trouvée dans l'intervalle [23,24].

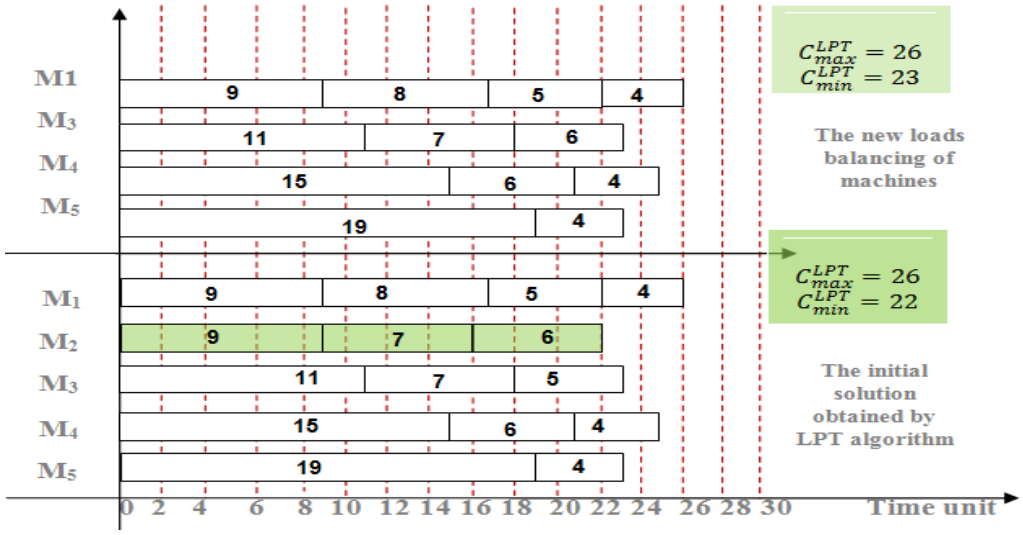


Figure 5.8. Diagramme de Gantt pour 5 machines et 15jobs sans et avec ré-ordonnancement

5.8.1 Etude comparative et interprétationdes résultats

Dans la section qui suit on va montrer et étudier les différents résultats obtenus pour trouver le C_{max} en utilisant les deux algorithmes LPT et MMIPMH. Ces résultats seront présentés sous forme de tableaux.

Tableau 5-2 Cmax pour un problème P||Cmax avec 4 machines et 9 jobs

Algorithme MMIPMH					
Machines	**Séquences de Jobs**		**C_i**	**C_{ma}**	**Itérations**
M_1	$P_8P_7P_9$	$P_8P_7P_9$	12	12	4
M_2	P_3 P_4	P_3P_4	12		
M_3	P_2 P_5	P_2P_6	12		
M_4	P_1P_6	P_1P_5	12		
Algorithme LPT					
Machines	**Séquences de Jobs**		**C_i**	**C_{ma}**	**Itérations**
M_1	$P_1P_8P_9$		15	15	1
M_2	P_2 P_7		11		
M_3	P_3 P_6		11		
M_4	P_4P_5		11		

Le tableau 5.2 nous montre les résultats obtenus du C_{max} pour le problème P||C_{max} avec 4 Machines et 9 Jobs, nous remarquons que le C_{max} obtenue par l'algorithme LPT est égale à 15, tandis que celui trouvé par l'algorithme MMIPMH est égale à 12. Ces résultats montrent que la méthode que nous avons développée(MMIPMH) donne de meilleurs résultats par rapport à ceux donnés par la méthode qui utilise la règle LPT. Cette amélioration des résultats est due à la répartition bien équilibrée des jobs sur l'ensemble des machines qui existe dans le système.

Le tableau 5.3 nous donne un résumé des résultats de simulations de plusieurs instances (plusieurs configurations d'un même système) de machines parallèles identiques, nous avons pris un système qui contient 4 machines parallèles identiques et nous avons varier le nombre de jobs pour le même système. Les résultats obtenus pour tous les cas de figures montrent que la méthode développé MMIPMH donne toujours les meilleures valeurs de C_{max}, ce qui montre la puissance de la méthode MMIPMH qui se base sur le principe d'équilibre des charges sur les machines qui constitues le système et la recherche du meilleur C_{max} dans un espace de recherche très réduit afin minimiser le temps de calcul au maximum et de se rapprocher à la solution optimale de C_{max} de n'importe quel système.

Tableau 5-3 C_{max} pour 4 machines et N jobs

Algorithme LPT et MMIPMH					
N°	**Machines**	**Jobs**	**C_{max}**		**Itérations**
	(m)	(N)	LPT	MMIPMH	
1	4	9	15	12	4
2		10	18	17	
3		11	22	20	
4		12	23	21	
5		15	31	29	
6		18	47	46	
7		20	50	48	

Le tableau 5-4 nous illustre les résultats trouvés lors de la simulation de plusieurs systèmes de machines parallèles identiques avec à chaque fois une variation dans le nombre de jobs. Cela pour vérifier est ce que les performances trouvées dans les tableaux précédents sont valables même en cas de variation de la taille du système avec la variation du nombre de jobs. Nous pouvons confirmer que la méthode que nous avons développée est soit plus performante que la méthode qui utilise la règle LPT ou dans le pire des cas égale à cette dernière et ceci quel que soit la taille ou le nombre de jobs utilisés.

Tableau 5-4 C_{max} pour M machines et N jobs.

Algorithme LPT et MMIPMH					
N°	**Machines**	**Jobs**	**C_{max}**		**Itérations**
	(m)	(N)	LPT	MMIPMH	
1	2	8	24	24	5
2	3	11	24	23	5
3	4	9	15	12	4
4	5	15	26	23	3

5.9 Conclusion

Dans ce chapitre nous avons parlé des problèmes d'ordonnancement de machines simple dans la littérature avant de parler des problèmes d'ordonnancement de machines parallèles identique, qui a été étudié par plusieurs chercheurs dans le passé. Parmi les problèmes d'ordonnancement de machines parallèles identiques on trouve le problème $P\|C_{max}$ qui reste toujours un problème d'ordonnancement NP-Complet. Pour cela nous avons développé un algorithme pour résoudre de problème.

L'algorithme que nous avons développé noté MMIPMH a été inspiré du modèle de réseaux de neurones d'Hopfield qui utilise une matrice de connexion symétrique et utiliser la règle de priorité LPT pour sélectionner ordonnancer les jobs au départ. Notre algorithme va trouver le meilleur équilibre des charges sur les machines parallèles identiques qui forment les systèmes étudiés. Cela va nous permettre aussi de trouver un intervalle de calcul pour trouver le meilleur C_{max} et éliminer toutes les séquences qui sont en dehors de cet intervalle, cet approche va réduire considérablement le temps de calcul pour trouver le meilleur C_{max} .
Après la présentation de notre méthode nous avons montré plusieurs exemples explicatifs du fonctionnement de cette dernière. En fin du chapitre nous avons montrés plusieurs résultats qui confirme la puissance de notre algorithme pour trouver le meilleur C_{max} quelques soit les variations dans la taille du système de production ou le nombre de jobs dans le système.

Conclusion générale

Ce travail fait partie de résoudre les problèmes d'ordonnancement des systèmes flexibles de production. Nous nous sommes particulièrement intéressés à ceux avec flexibilité de routage. Ce type de flexibilité qui peut être trouvé, dans chaque système qui comprend des machines parallèle identiques, alternatives ou redondantes offre plusieurs avantages en balançant mieux les charges des machines, en permettant au système de continuer à fonctionner et de maintenir ses performances élevées malgré l'occurrence de certains événements inattendus tels que les pannes de machines,… .Ces biens faits sont poussé plusieurs chercheurs à le prendre en compte lors de l'implémentation et la configuration des systèmes, ce qui explique l'existence de nombreux travaux qui traitent les problèmes d'ordonnancement dans les systèmes manufacturiers avec présence de flexibilité.

Tous d'abord nous avons parlé dans les chapitres un et deux des généralités sur l'ordonnancement des systèmes de production, nous nous somme s'intéressés à l'étude de l'ordonnancement dans sa globalité, à savoir ses différentes composantes (tâches, ressources, contraintes, critères), ses différents types d'ateliers(job-shop, flow-shop et open-shop) et les méthodes d'optimisation les plus utilisées pour sa résolution, allant des méthodes exactes aux méthodes approchées puis nous avons parlé d'intelligence artificielle et ses méthodes ensuite nous avons présenté les algorithme évolutionnaires qui s'inspirent de la nature et du comportement social des individus.

Dans le chapitre trois nous sommes entrés dans le vif du sujet où nous avons présenté une étude bibliographique et un état de l'art sur les méthodes de résolutions du problème d'ordonnancement job shop, où nous avons décrit les problèmes d'optimisation combinatoire de manière générale. Ce chapitre compris les notions principales des Métaheuristiques et leur classification. Nous avons cité les différentes algorithmes évolutionnaires (AE). Puis nous avons parlé d'une analyse du comportement de l'intensification et de la diversification pour des variantes d'algorithmes génétiques et GA. Nous avons présenté une comparaison entre ces variantes. Les résultats obtenus sont présentés dans deux chapitres qui suivent

Conclusion générale

Dans le chapitre 4 nous avons présenté les résultats en présence de pannes sur les machines du système étudié. Nous avons commencé par donner une introduction, puis une description complète du modèle FMS étudié décrite par le nombre maximal de machines identiques pour chaque tâche. Les algorithmes évolutionnaires (EA), en particulier les variantes des algorithmes génétiques ont été développés pour fournir de bonnes approximations de la solution optimale. En raison de leur simplicité et de leur facilité de mise en œuvre, nous avons présenté une adaptation de deux algorithmes à base des variantes d'algorithmes génétiques (un algorithme mémétique avec gestion de population et approche de recherche de dispersion) pour les décisions de routages de pièces afin améliorer les performances des systèmes de production flexibles dans un problème combinatoire NP-difficile avec introduction des pannes. Notre contribution dans ces aspects touche la formulation du problème, la proposition d'un codage, d'opérateurs de croisement et de mutation ainsi que d'un algorithme spécifique à la résolution de problèmes d'ordonnancement FMS.

Les résultats montrent la performance de l'algorithme mémétique avec la gestion de population et la recherche de dispersion avec introduction des pannes. Nous avons présenté une analyse de sensibilité de ces algorithmes, en étudiant l'influence de variation des paramètres internes sur ses performances, pour pouvoir tirer des informations sur les meilleures valeurs des paramètres de la méthode.

L'idée d'un meilleur balancement des charges sur les machines du job shop, nous a poussé de l'essayer sur un système de machines parallèles qui contient une flexibilité dans la sélection des machines. Pour cela nous avons développé un algorithme qu'on a appelé MMIPMH pour sélectionner le meilleur makespan pour un système de machines parallèles identiques. Ce travail a été présenter dans le dernier chapitre, où nous avons présenté et expliqué l'algorithmes en donnant des exemples explicatifs.

Les résultats de cet algorithme on étét comparer à ceux de la règle LPT qui est l'une des méthodes qui donne les meilleurs résultats pour le problème d'ordonnancement $P||C_{max}$ (la minimisation du Cmax pour le problèmede machines paralleles identiques sans contraintes) qui est un problème NP-complet. Ces résultats ont confirmé que la méthode MMIMPH que nous avons développée donne pratiquement toujours des résultats meilleurs que ceux données par la règle LPT et des résultats égaux à ceux de la LPT dans le pire des cas. La puissance de cette méthode est dans la manière avec la quelle on fait l'équilibre des charges sur les différentes machines du systèmes. On note aussi que cette méthode donne des résultats dans un temps de

calcul assez raisonnable par rapport à d'autres méthodes à cause de la réduction de l'espace de recherche du makespan.

Enfin, avec les résultats encourageants obtenus, il serait intéressant de pouvoir développer certains autres aspects dans nos travaux futurs envisagés en perspectives comme :

- Adapté la méthode MMIMPH dans les systemes job shop et flow shop.
- Etudier l'impact de l'hybridation de MMIPMH avec des méthodes approchées.

Références bibliographiques

[Abada 1997] Abada A, Contribution a la résolution des problèmes d'ordonnancement par réseaux de neurones. Thèse de doctorat Laboratoire d'Automatique de Grenoble - L.A.G -I.N.P.G, 1997.

[Andrea et al. 2007] R. Andrea, and D. Gino, "Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method", Rob. Comput-Integr Manuf., vol. 23, pp. 503-516, 2007.

[Akyo et al. 2006] Derya Eren Akyol and Mirac Bayhan , G.”Minimizing Makespan on Identical Parallel Machines Using Neural Networks “. I. King et al. (Eds.): ICONIP, Part III, LNCS 4234, pp. 553, 2006.

[Akyol et al. 2007] Akyol, D.E. and Araz, O.U. A Neural Network Based Decision Support System for Real-Time Scheduling of Flexible Manufacturing Systems. Book chapter in: Operations Research Proceedings 2007, book series: Operations Research Proceedings Volume .Part IV, 83-88, 2007.

[Bachelet 1999] Bachelet, V. Métaheuristiques parallèles hybrides : application au QAP. Habilitation à Diriger des Recherche, USTL LIFL France, 1999.

[Bel et al 1988] Bel .G, Bensana .E, Dubois .D, "Construction d'ordonnancements prévisionnels : un compromis entre approches classiques et systèmes experts", Revue d'Automatique, Productique, Informatique Industrielle, vol.22, n°5, 1988, pp. 509-536, 1988.

[Blum et Roli 2003] C.Blum, A.Roli., ‘’Metaheuristics in combinatorial optimization’,’Overview and conceptual comparison. ACM Computing Surveys 35(3), 268–308, 2003.

[Benalia 2005] Benalia M., Parallèlisation de la recherche dispersée Étude du modèle des îles : Application au problème MAX-W-SAT. Mémoire de magister en informatique soutenue à l’institut national d’informatique-INI-, 2005.

[Benatchba 2005] Benatchba K., Modèle d’exécution pour l’aide à la solution problème MAX-SAT. Thèse de doctoratsoutenue àl’institut national d’informatique Alger, 2005.

[Bensana et al 1988] Bensana.E, Bel .G, Dubois .D, "OPAL: a multi-knowledge based system for industrial job-shop scheduling", International Journal of Production Research, vol.26, n°3, 1988, pp. 795-815, 1988.

[Bénassy 1987] Bénassy,Q. La gestion de production, Hermes,paris, 1987.

[Boisson 2008] Boisson, Jean-Charles. "Modélisation et résolution par métaheuristiques coopératives: de l'atome à la séquence protéique". Rapport de thèse de doctorat. Université Lille 1, 2008.

[Boukef 2009] Boukef .H. Sur l’ordonnancement d’ateliers job-shop flexibles et flow-shop en industries pharmaceutiques : optimisation par algorithmes g´en´etiques et essaims particulaires, these doctorat, 2009

Références bibliographiques

[Bracker 1988] Bracker P: An efficient algorithm for the job-shop problem with two jobs, Computing, 40(4): pp.353-359, 1988.

[Browne et al. 2016] J. Browne, D. Dubois, K. Rathmill, S. P. Sethi, and K. E. Stecke, "Classification of flexible manufacturing systems," The FMS magazine, vol. 2, no. 2, pp. 114–117,1984.[xx]José ELOUNDOUTHÈSE de doctoratModélisation Multi-contraintesd'un Système de Production Flexible.Thèse soutenue le 11 juillet 2016.

[Caprihan et al. 1997] R. Caprihan, and S. Wadhwa, "Impact of routing flexibility on the performance of an FMS-A simulation study", Int. J. Flex. Manuf. Syst., vol.9, no. 3, pp. 273-298,.[http://dx.doi.org/10.1023/A:1007917429815], 1997

[Caprihan et al. 2006] R. Caprihan, A. Kumar, and K.E. Stecke, "A fuzzy dispatching strategy for due date scheduling of FMSs with information delays", Int. JFlex. Manuf. Syst., vol. 18, no. 1, pp. 29-53, [http://dx.doi.org/10.1007/s10696-006-9002-4], 2006 .

[Cavique et al. 2001] Cavique L., Rego C. and Themdo I., A Scatter Search Algorithm For the Maximum Clique Problem , In Ribeiro, C.C, Hansen, P. (Eds.), Essays and Surveys in Metaheuristics,. Kluwer-Academic Publishers, 2001.

[Chang 1986] D. Chang, "Using SLAM to Design the Material Handling System of a Flexible Manufacturing System", Int. J. Prod. Res., vol. 24, no. 1, pp.15-26, 1986.

[Chang 2007] A.Y. Chang, "On the measurement of routing flexibility: A multiple attribute approach", Int. J. Prod. Econ., vol. 109, pp. 122-136, [http://dx.doi.org/10.1016/j.ijpe.2006.11.011], 2007.

[Chaudhry 2010] Chaudhry. I.A, Sultan. M and Riaz.A ." Minimizing Makespan for Machine Scheduling and Worker Assignment Problem in Identical Parallel Machine Models Using GA" Proceedings of the World Congress on Engineering Vol III WCE 2010, June 30 - July 2, 2010, London, U.K, 2010.

[Chrétienne 1988] Chrétienne, P and Carlier, J . Problèmes d'ordonnancement : modélisation, complexité, algorithmes. Masson, 1988.

[Chunwel et al 2001] Chunwel, and W. Zhining, ""A Genetic Algorithm Approach to the Scheduling of FMSs with Multiple Routes", The Shanghai Jiaotong University International", J. Flexible Manuf. Syst., vol. 13, pp. 71-88, [http://dx.doi.org/10.1023/A:1008148313360], 2001.

[Conway et al. 1967] Conway, R W, W L Maxwell and L W Miller. Theory of Scheduling, Addison-Wesley, Reading Massachusetts, 1967.

[Cook et al 1998] Cook, W., Cunningham, W., Pulleyblank, W. R., et Schrijver, A. Combinatorial Optimization. New York: Wiley and Son, 1998.

[Darwin 1859] Darwin. C, On the Origin of Species by Means of Natural Selection. Londres, John Murray, 1859.

[Dogramaci et al. 1979] Dogramaci A and Surkis J., "Evaluation of a heuristic for scheduling independent jobs on parallel identical processors", Management Science, Volume 231208-1216, 1979.

[Dreyfus et al. 2004] Dreyfus G., Martinez J.M., Samuelides M., Reseaux de neurones, Methodilogie et applications. April 2004.

[E Eiben 1998] A.E Eiben , C.A Schippers,'' On Evolutionary Exploration and Exploitation''. Fundamenta Informaticae 35(1-4), 35–50,1998.

Références bibliographiques

[ElMekkawy et al. 2003] T.Y. ElMekkawy, and H.A. ElMaraghy, "Real-time scheduling with deadlock avoidance in flexible manufacturing systems", Int. J. Adv Manuf. Technol., vol. 22, pp. 259-270 [http://dx.doi.org/10.1007/s00170-002-1468-y], 2003.

[Farbe 2009] Farbe F., Conduite orientée ordonnancement d'un simulateur dynamique hybride : application aux procédés discontinus. Thèse de doctorat soutenue à l'Institut National Polytechnique de Toulouse, 2009.

[Fogel 1993] Fogel D.B., On the philosophical differences between evolutionary algorithms and genetic algorithms. In D.B. Fogel and W. Atmar, editors, Proceedings of the SecondAnnual Conference on EvolutionaryProgramming, pages 23–29. Evolutionary Programming Society, La Jolla, 1993.

[Gao et al. 2009] J. Gao, G. He, & Y. Wang, "A new parallel genetic algorithm for solving multiobjective scheduling problems subjected to special process constraint," International Journal Advanced Manufacturing Technology,Vol. 43pp. 151–160, 2009

[Garavelli 2001] A.C. Garavelli, "Performance analysis of a batch production system with limited flexibility", Int. J. Prod. Econ., vol. 69, pp. 39-48 [http://dx.doi.org/10.1016/S0925-5273(00)00043-8] , 2001.

[Garey1979] Garey, M.R and Johson, D.S. Computers and intractability a guide of the theoryof NP- completeness. W.H. Freeman and company, San Fransisc, 1979.

[Ghomri 2007] L. Ghomri, "Influence des contraintes et des perturbations sur les performances des règles de routages dans un FMS", La conference l internationale Conception et Production Intégrées, 2007.

[German et al 2016] German.Y, Badi.I, Bakir.A and Shetwan.A." Scheduling to Minimize Makespan on IdenticalParallel Machines". International Journal of Scientific & Engineering Research, Volume 7, Issue 3, ISSN 2229-5518, March-2016.

[Giard 1988] Giard, V., Gestion de production,3^{e}édition, 1230 pages Economica, 1988.

[Goldberg 1989] Goldberg D.E., Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, 1989.

[Glover 1995] Glover F., Scatter search and star-paths: beyond the genetic metaphor. OR Spektrum, 17:125–137, 1995.

[Graham et al 1979] Graham, R., Lawer, E., Lenstra, J., & Rinnooy Ken, A. Optimization and approximation in deterministic sequencing and scheduling. Annals of Discrete Mathematics, 5, 287–326, 1979.

[Grinsted 1995] S. Grinsted, "Alternative Routing: Some Rules for the "Where-Next" Decision", In: Proceedings of the Second International Conference of European Operations Management Association, Enschede, the Netherlands , pp. 171–177,1995.

[Haq et al. 2003] N.A. Haq, T. Karthikeyan, and M. Dinesh, "Scheduling decisions in FMS using a heuristic approach", Int. J. Adv. Manuf. Technol., vol. 22 no. 5–6, pp. 374-379, [http://dx.doi.org/10.1007/s00170-002-1474-0], 2003.

[Hao et al. 2008] Hao Shao and al."Minimizing Makespan for Parallel Batch Processing Machines with Non-identical Job Sizes Using Neural Nets Approach", 978-1-4244-1718-6/08. IEEE,2008.

[Hansheng 1999] L.Hansheng, K .Lishan,''Balance between exploration and exploitation in genetic search'', Wuhan University Journal of Natural Sciences 4(1), 28–32, 1999.

Références bibliographiques

[Harun et al. 2009] Harun Resit Yazgan, Semra Boran and Kerim Goztepe , "Selection of dispatching rules in FMS: ANP model based on BOCR with choquet integral", Int. J. Adv. Manuf. Technol., vol. 49, no. 5-8, pp.785-801, 2009.

[Hasan et al. 2008] S.M.K, Hasan, R. Sarker, D. Essam, & D.Cornforth, "Memeticalgorithms for solving job-shop scheduling problems," Memetic Comp,Vol. 1, pp.69–83, 2008.

[Hashemian 2012] Hashemian N. Claver Diallo · Béla Vizvári Makespan minimization for parallel machines scheduling with multiple availability constraints". Published online: 21 January 2012© Springer Science+Business Media, LLC 2012. Ann Oper Res 213:173–186, 2014.

[Hassam et Sari 2010] Hassam .A & Sari .Z, "Selection of alternative routings in real time: DMM and modified DMM rules", International Journal of Product Development, Vol. 10, Nos. 1/2/3, 2010.

[Hassam 2012] Hassam .A . Thèse de Doctorat Es Sciences en Productique Intitulée :Développement et analyse de méthodes d'ordonnancement temps réel pour les systèmes flexibles de production, 2012.

[Hebb 1949] Hebb, D. The Organization of Behavior, New York: Wiley, 1949.

[Heinrich 1995] Heinrich, K." A heuristic algorithm for the loading problem in flexible manufacturing systems. Int. J. Flex. Manuf. Syst. 7, 229–254", 1995.

[Hoffmeister et al. 1991] Hoffmeister F.and BackT.. Genetic algorithms and evolution strategies: Similarities and differences. In H.-P. Schwefel and R. M¨anner, editors, Parallel ProblemSolving from Nature – PPSN I, Lecture Notes in Computer Science 496, pages 455–469. Springer, Berlin, 1991.

[Hoos et al. 2004] Hoos H.H. and Stützle T., Stochastic local search: Foundations and applications. Morgan Kaufmann, 2004

[Holland 1975] Holland J.H., Adaptation in natural and artificial systems .PhD, MichiganPress Univ., Ann Arbor, MI, 1975.

[Houbad 2011] H. Houbad, "Sul'ordonnancement d'ateliers job-shop:,'' Optimisation par memetic et essaims particulaires". Thèse de magister soutenue à l'Université de Tlemcen, 2011.

[Hu 2004] Hu P-C., "Minimizing total tardiness for the worker assignment scheduling problem in identical parallel-machine models", ,International Journal of Advanced Manufacturing Technology, vol. 23(5–6), pp. 383-388, 2004.

[Joseph et al. 2011] O.A. Joseph, and R. Sridharan, "Effects of routing flexibility: sequencing flexibility and scheduling decision rule on the performance of aflexible manufacturing system", In: Int J Adv Manuf Technol, vol. 56. Springer-Verlag London Limited, no. 1-4, pp. 291-306.[http://dx.doi.org/10.1007/s00170-011-3158-0], 2011.

[Joseph 2017]O.A. Joseph, "Evaluation of routing flexibility of a flexible manufacturing system using simulation modelling and analysis", Int. J. Adv.Manuf. Technol., vol. 56, no. 1-4, pp. 273-289, 2011.[http://dx.doi.org/10.1007/s00170-011-3153-5]Adaptation of Memetic Algorithm with Population The Open Automation and Control Systems Journal, Volume 9 13, 2017,

[Kaufmann 1992] Kaufmann A, Introduction à la logique floue. Techniques de l'Ingénieur, A 120, Mathématiques pour l'ingénieur, 1992.

Références bibliographiques

[Kedia 1971] Kedia SK., “A job scheduling problem with parallel processors”, Unpublished report, Department of Industrial and Operations Engineering, University of Michigan, AnnArbor, MI, 1971.

[Kötzing et al. 2011] T. Kötzing, D. Sudholt, M .Theile,’’How crossover helps in pseudo-boolean optimization’’. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 989–996. ACM, GECCO, 2011.

[Kovàcs et al 1994] Kovàcs, G. L., Mezgàr, I., Kopàcsi, S., Gavalcovà, D. and Nacsa, J. . Application of artificial intelligence to problems in advanced manufacturing systems. International Journal of Computer Integrated Manufacturing Systems, 7(3), 153-160, 1994.

[Kim 2007] Y.W. Kim, "FMS scheduling based on timed Petri Net model and reactive graph search", Appl. Math. Model., vol. 31, no. 6, pp. 955-970,.[http://dx.doi.org/10.1016/j.apm.2006.10.023], 2007.

[Kim et Roand 1991] J. Kim I.Roand, "Multi-criteria Operational Control Rules in Flexible Manufacturing Systems", Int. J. Prod. Res., vol. 28, no. 1, pp. 47-63, 1991.

[Kuruvilla et al 2015] Abey Kuruvilla and al.”Minimizing Makespan on Identical Parallel Machines”. International Journal of Operations Research and Information Systems, 6(1), 19-29, January-March 2015.

[Lawler et al 1989] Lawler E.L., Lenstra J.L., Rinnooy Kan, A.H.G., Schmoys, D.B.. Sequencing and scheduling: algorithms and complexity. Rapport BS-R8909, Centre for Mathematics and Computer Science, Amsterdam, 1989.

[Le Moal et Tarondeau 1979] Le Moal, P et Tarondeau, JC.: "Un défi à la fonction production". Revue française de gestion, pp. 9-14. 1979.

[Lee In et al. 2000] Lee In, Shaw M.J., A neural-net approach to real time flow-shop sequencing. Computers and Industrial Engineering, 2000.

[Li et al 2006] Li, D.C., Wu, C., and Chang, F.M. Using data continualization and expansion to improve small data set learning accuracy for early flexible manufacturing system (FMS) scheduling. International Journal of Production Research, 44(21), 4491-4509, 2006.

[Liu et al. 1999] Liu, M., Wu, C.”A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines”,. Artif. Intell. Eng. 13(4), 399–403, 1999.

[Mahmoodi et al. 1999] F. Mahmoodi, and C.T. Mosier, "The Effects of Scheduling Rules and Routing Flexibility on the Performance of Random Flexible Manufacturing System", In: International Journal of Flexible Manufacturing Systems, vol. 11. Academic Publishers: Boston, no. 3, pp.271-289, 1999.

[McNaughton 1959]McNaughton R., «scheduling with deadlines and loss functions», Management Sci. 6, 1-12, 1959.

[Meignan 2004] Meignan D., une approche organisationnelle et multi- agent pour la modélisation et l’implantation métaheuristiques : Application aux problèmes d’optimisation de réseaux de transports. Thèse de doctoratsoutenue àl’Université de Franche-Comté, 2004.

[Meignan 2008] Meignan, D. , Une approche organisationnelle et multi-agent pour la modélisation et l'implantation de métaheuristiques, Application aux problèmes

d'optimisation de réseaux de transports. Thèse de doctorat, Université de Technologie de Belfort-Montbeliard, France, 2008.

[Min-Chun et al 2009] Min-Chun Yu, and Greene J.Timothy, "ShopAn operational measure of routing flexibility in a multi-stage multi-product production system", Int J Adv Manuf Technol, vol. 43, pp. 357-364, Springer-Verlag London Limited, 2009.

[Min et al. 1998]Min, H. S., Yih, Y. and Kim, C. O. A competitive neural network approach to multi-objective FMS scheduling. International Journal of Production Research, 36(7), 1749-1765, 1998.

[Michalewicz 1996] Michalewicz Z., "Genetic algorithms + data structures = evolution programs", thirdedition, London: Springer Verlag, 1996.

[Mitchell 1998] Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, 1998.

[Mokotoff 2001] Mokotoff,E., "Parallel machine scheduling problems: A survey",Asia-Pacific Journal of Operational Research, vol. 18, pp.193-242, 2001.

[Moscato 1993] Moscato P., An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. Annals ofOperations Research, 41:85–121, 1993.

[Moscato 1989] Moscato P., On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. Technical report, Caltech Concurrent Compulation Program, 1989.

[O'kane 2000] O'kane, J. F. A knowledge-based system for reactive scheduling decision-making in FMS. Journal of Intelligent Manufacturing, 11(5), 461-474, 2000.

[OULD-SAADI 2012] Youcef, OULD-SAADI. Conception d'une Métaheuristique Réactive. Rapport de thèse de doctorat. Université Mohamed Boudiaf des sciences et de la technologie d'Oran, 2012.

[Ow et al 1988] Ow .P.S, Smith S.F, "Viewing scheduling as an opportunistic problem solving process", Annals of Operations Research, vol.12, n°, 1988, pp. 85-108, 1988.

[Pei et al. 2008] C.C, Pei, H.C, Shih, Y.F, Chin, & L.C, Chien "Genetic algorithmintegrated with artificial chromosomes for multi-objective flow shop scheduling problems," Applied Mathematics and Computation, Vol.205,pp.550–565, 2008.

[Petrovic 2005] S. Petrovic, & C. Fayad, "A genetic algorithm for job shop schedulingwith load balancing," School of Computer Science and Information Technology, University of Nottingham, Nottingham, 2005.

[Pezzella et al. 2008] F. Pezzella, G. Morganti, & G. Ciaschetti, "A genetic algorithm for the flexible jobshop scheduling problem," Computers and Operations Researc,Vol. 35 (10), pp. 3202-3212, 2008.

[Pinedo 2008] Pinedo, M. Scheduling:Theory,algorithmsandsystems(2ndedition). Prentice Hall, 2008.

[Prodhon et al. 2006] Prodhon C., et Wol-er-Calvo R., A memetic algorithm with population management (MA/PM) for the capacitated location-routing problem. In J.Gottlieb et G. R. Raidl (Eds.), Lecture Notes in Computer Science, 2006

[Qian 2008] B. Qian, L. Wang, D.X, Huang, & X. Wang, "Schedulingmultiobjective job shops using a memetic algorithm based ondifferential evolution", International Journal of Advanced Manufacturing Technology, Vol. 35, pp.1014–1027, 2008.

Références bibliographiques

[Ouazene et al. 2016] Yassine Ouazene, Farouk Yalaoui, Alice Yalaoui and Hicham Chehade." Theoretical Analysis of Workload Imbalance Minimization Problem on Identical Parallel Machines". N.T. Nguyen et al. (Eds.): ACIIDS 2016, Part II, LNAI 9622, pp. 296–303, 2016.

[Raghavendra et al. 2011] Raghavendra, B.V., Murthy, A.N.N."Workload balancing in identical parallel machine scheduling while planning in flexible manufacturing system using genetic algorithm", ARPN J. Eng. Appl. Sci. 6(1), 49–55, 2011.

[Raghavendra et al. 2010] Raghavendra, B.V., Murthy, A.N.N.: Some solution approaches to reduce the imbalance of workload in parallel machines while planning in flexible manufacturing system through genetic algorithm", Int. J. Eng. Sci. Technol. 2(5), 724–730, 2010.

[Rajakumar et al. 2007] Rajakumar S, Arunachalam VP and Selladurai V., "Workflow balancing in parallel machines through genetic algorithm", International Journal of Advanced ManufacturingTechnology, Volume 33, Issue 11–12, 1212–1221, 2007

[Rajamani et al. 1990] D. Rajamani, N. Singh, and Y.P. Aneja, "Integrated design of cellular manufacturing systems in the presence of alternative process plans", Int.J. Prod. Res., vol. 28, no. 8, pp. 1541-1553, [http://dx.doi.org/10.1080/00207549008942811], 1990.

[Rahul et al. 1997] C. Rahul, and W. Subhash, "Impact of Routing Flexibility on the Performance of an FMS—A Simulation Study", Int. J. Flex. Manuf. Syst.,vol. 9, pp. 273-298, [http://dx.doi.org/10.1023/A:1007917429815],1997.

[Roubellat 2001] Roubellat F., Ordonnancement de la production. Hermès Sciences publications, 2001.

[Ruiz et al 2001] Ruiz, D., Canton, J., Mara, N. J., Espuna, A. et Puigjaner, L. On-line faultdiagnosis system support for reactive scheduling in multipurpose batch chemical plants. Computers and Chemical Engineering, 25(4), 829-837, 2001.

[Saka 1984] Sakarovitch,M., « Optomisation combinatoire : Méthode mathématique et algorithme », 1984.

[Sabuncuoglu 1998] Sabuncuoglu. I, Scheduling with neural networks: a review of literature and new research directions, Production Planning and Control, vol.9, n°1, pp. 2-12, 1998.

[Saygin et al.1999] C. Saygin, and S.E. Kilic, "Integrating flexible manufacturing systems with scheduling in flexible manufacturing system", Int. j. adv. Manuf.Technol., vol. 15, no. 4, pp. 268-280, 1999.

[Schwefel et al. 4998] Schwefel H.-P and Bäck T., Artificial evolution: How and why? In D. Quagliarella, J.P´eriaux, C. Poloni, and G. Winter, editors, Genetic Algorithms and EvolutionStrategyin Engineering and Computer Science: Recent Advances and IndustrialApplications, pages 1–19. Wiley, Chichester, 1998

[Senties 2007] Senties B., Methodologie D'aide a La Decision Multicritère Pour L'ordonnancement D'ateliers Discontinus. Thèse de doctorat soutenue àL'institut National Polytechnique De Toulouse, 2007.

[Sethi et al. 1990] A. Sethi and S. Sethi, "Flexibility in manufacturing : A survey," International Journal of Flexible Manufacturing Systems, vol. 2, pp. 289–328, July 1990.

[Sevaux 2004] Sevaux M., Métaheuristiques Stratégie pour l'optimisation de la production de biens et de services. Habilitation à diriger desrecherches, Laboratoire d'Automatique, de

Mécanique d'informatiqueIndustrielles et Humaines du CNRS (UMR CNRS 8530) dans l'équipesystèmes de production, 2004.

[Shu-Chen et al. 2009] Shu-Chen Cheng et al. « Dynamic hard-real-time scheduling using genetic algorithm for multiprocessor task with resource and timing constraints," Expert Systems with Applications, Vol. 36, pp.852, 2009.

[Shukla et al. 2001]C.S. Shukla, and F.F. Chen, "An intelligent decision support system for part launching in a flexible manufacturing system", Int. J. Adv. Manuf. Technol., vol. 18, pp. 422-433 [http://dx.doi.org/10.1007/s001700170052] , 2001.

[Sik et Yih 2003] Sik H., Yih Y., Development of a real-time multi-objective scheduler for a semiconductor fabrication system, Taylor & Francois group, 2003.

[Singhet al. 1991] N. Singh, and B.K. Mohanty, "A Fuzzy Approach to Multi-objective Routing Problem with Applications to Process Planning in Manufacturing Systems", Int. J. Prod. Res., vol. 29, no. 6, pp. 1161-1170, [http://dx.doi.org/10.1080/00207549108930125], 1991.

[Sörensena et al. 2004] Sörensena[a] K., Sevauxb[b] M., MA|PM: memetic algorithms with population management, [a]University of Antwerp, Faculty of Applied Economics, Prinsstraat 13, B-2000 Antwerp, Belgium [b]University of Valenciennes,CNRS, UMR 8530, LAMIH-SP, Le Mont Houy-Bat Jonas 2, F-59313 Valenciennescedex 9, France , 2004.

[Souier et al 2010] Souier,M, Hassam A, and Sari, Z. "Meta-heuristics for real time routing selection in FMS", In: Lyes Benyoucef and Bernard Grabot (Ed.),Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management: Springer-Verlag, London, pp. 221-247, 2010.

[Souier 2012] M. Souier . Thèse de Doctorat en Sciences en Productique Intitulée : Investigations sur la sélection de routages alternatifs en temps réel basées sur les métaheuristiques -les essaims particulaires- ,2012.

[Smith et al 1992] Smith, S. F., Knowledge-Based production management: approaches, results and prospects, Production Planning and Control, vol.3, n°4, 350-380, 1992

[Talbi 2009] Talbi. E, Metaheuristics: From Design to Implementation. JohnWiley and Sons, Inc, 2009.

[Tavakkoli et al. 2008] M.R, Tavakkoli, K.Y, Gholipour, & R. Cheraghalizadeh, "A genetic andmemetic algorithm approach to sequencing and scheduling of cellular manufacturing systems," International Journal of Management Science and Engineering Management, Vol.3(2), pp.119–130, 2008.

[Tseng et al. 2009] Tseng, L. Y. and Lin, Y. T., "A hybrid genetic local search algorithm for the permutationflow-shop scheduling problem", European Journal of Operational Research, Volume 198,Issue 1, 84-92, 2009.

[Ümit et al. 2008] B. Ümit, and F. Murat, "Erinç, A parametric fuzzy logic approach to dynamic part routing under full routing flexibility", Comput. Ind. Eng. vol. 55, pp. 15-33, 2008.[http://dx.doi.org/10.1016/j.cie.2007.11.013]

[Wang et al 1995] Wang, L. C., Chen, H. M. and Liu, C. M.. Intelligent scheduling of FMSs with inductive learning capability using neural networks. International Journal of Flexible Manufacturing Systems, 7(2), 147-175, 1995.

Références bibliographiques

[Warren 1994] T. Warren Liao, "Design of line-type cellular manufacturing systems for minimum operating and material-handlin g costs", Int. J. Prod. Res.,vol. 32, no. 2, pp. 387-397,.[http://dx.doi.org/10.1080/00207549408956939], 1994

[Widmer et al. 2001] Widmer M., Hertz A., and Costa D., Les métaheuristiques, chapter 3, pages 55-93. hermesedition, 2001.

[Xia et al. 2005] Xia W and Wu Z., "An effective hybrid optimization approach for multi-objectiveflexible job-shop scheduling problems", Computers and Industrial Engineering, Volume 48, Issue 2, 409-425, 2005.

[Yang et al. 2014] Yang, Xin-She, Suash Deb, and Simon Fong. "Metaheuristic algorithms: optimal balance of intensification and diversification." Applied Mathematics & Information Sciences 8.3, 977, 2014

[Yang 2008] X. S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, UK, 2008.

[Yang 2010] X. S .Yang,''Engineering Optimization: An Introduction with Metaheuristic Applications'', John Wiley and Sons, USA, 2010.

[Yeh et al. 2014] Yeh, W.-C., Lai, P.-J., Lee, W.-C., & Chuang M.-C., " Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects", Information Sciences, Vol. 269, pp. 142-158, 2014.

[Zhao et al. 2001] C. Zhao, and W. Zhiming, "A Genetic Algorithm Approach to the Scheduling of FMSs with Multiple Routes", Int. J. Flex. Manuf. Syst., vol.13, pp. 71-88, 2001.[http://dx.doi.org/10.1023/A:1008148313360], 2001.

Résumé

Les systèmes de fabrication et de production moderne offrent une grande flexibilité et des avantages divers tels que l'augmentation de la productivité, l'augmentation de l'utilisation des ressources, la réduction des-en- cours etc. Dans de tels systèmes, l'ordonnancement est un domaine très important et très utilisé. Il existe plusieurs techniques et méthodes pour résoudre les problèmes d'ordonnancement, parmi ces approches et techniques de résolution des problèmes d'ordonnancement nous pouvons citer les méthodes de sélection de routages alternatifs qui sont des techniques récentes et très peu étudiées. Dans notre thèse nous allons étudier l'approche de résolution de problèmes d'ordonnancement avec deux méthodes approchées qui s'inspirent de la nature et du comportement social des individus et nous avons développé un algorithme qu'on a appelé MMIPMH pour sélectionner le meilleur makespan pour un système de machines parallèles identiques et améliorer les performances données par les méthodes existantes.

Mots clé : Ordonnancement, méthodes approchées, Sélection de routages, machine parallèle

Abstract

Modern manufacturing and production systems offer great flexibility and various advantages such as increased productivity, increased use of resources, reduction of courses, etc. In such systems, scheduling is a very important and widely used area. There are several techniques and methods to solve the scheduling problems, among these approaches and techniques for solving scheduling problems we can mention the alternative routing selection methods that are recent techniques and very little studied. In our thesis we will study the problem-solving approach of scheduling with two approximate methods that are inspired by the nature and social behavior of individuals and we developed an algorithm that was called MMIPMH to select the best makespan for a system of identical parallel machines and improve the performance given by existing methods.

Keywords: Scheduling, approximate methods, Routing selection, parallel machine

ملخص

انظمة التصنيع والانتاج الحديثة توفر مرونة عالية وفوائد مختلفة مثل زيادة الإنتاجية، وزيادة استخدام الموارد، وخفض دورات أثناءها الخ في مثل هذه الأنظمة، جدولة هو مجال مهم جدا وتستخدم على نطاق واسع. هناك العديد من التقنيات والأساليب من أجل حل المشاكل جدولة بين هذه الطرق والتقنيات من أجل حل المشاكل جدولة يمكن أن نذكر طرق اختيار طرق بديلة والتي هي من تقنية حديثة ودرس قليلا جدا. في حالتنا سندرس نهج لحل مشاكل الجدولة مع اثنين من الطرق التقريبية مستوحاة من الطبيعة والسلوك الاجتماعي للأفراد وضعنا وقد دعا خوارزمية MMIPMH لاختيار الأفضل makespan لنظام من آلات موازية متطابقة وتحسين الأداء نظرا للطرق القائمة.

كلمات البحث: جدولة، وأساليب تقريبية، واختيار التوجيه، آلة موازية