République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abou Bakr Belkaid – Tlemcen

Faculté de Technologie

Département de Génie Electrique et Electronique

Mémoire de Fin d'Etudes

Pour l'obtention du diplôme de Master en Génie industriel

Spécialité : Productique

Thème:

Ordonnancement d'un atelier Flow shop par Métaheuristique récente

Préparé par :

LAIEB Ahmed

GRINE Amir

Soutenu en 15 Juin 2017 devant le jury composé de :

Président : MALIKI Fouad	MAA	EPST Tlemcen
Examinateur : BESSENOUCI Hakim Nadhir	MAA	Univ de Tlemcen
Examinatrice :MEKAMCHA Khalid	MAA	Univ de Tlemcen
Encadreur : HOUBAD Yamina	MAA	Univ de Tlemcen
Co-encadreur : BOUMEDIENE Fatima Zohra	Doctorante	Univ de Tlemcen

Année Universitaire : 2016/2017

Remerciements

D’abord, nous tenons à remercier **Allah**, notre créateur, pour le courage et la patience qu’il nous a donné pour accomplir ce travail.

Nous souhaitons exprimer notre plus sincère reconnaissance à mademoiselle **Yamina HOUBAD** et mademoiselle **BOUMADIENE Fatima Zohra**. Ses qualités humaines, ses qualités de chercheur, ainsi que leur patience nous ont permis de mener à terme cette mémoire. Nous tenons à leur faire part de toute notre gratitude pour nous avoir accordé tant de confiance.

Nous souhaitons remercier monsieur **MALIKI Fouad**, qui nous a fait l’honneur d’exercer les fonctions de président du jury et d’examinateur de thèse pour le temps qu’il a consacré à étudier notre travail et pour sa sympathie.

Nos remerciements s’adressent également aux examinateurs de notre mémoire du projet de fin d’étude monsieur **BESSENOUCI Hakim Nadhir** et MEKAMCHA Khalid pour avoir participer à ce jury de thèse. Nous leur exprimons toute notre reconnaissance pour l’intérêt porté à ce travail.

Nous tenons particulièrement à remercier tous les membres du laboratoire MELT Manufacturing Engineering Laboratory of Tlemcen et tous les enseignants pour leur sympathie et l'ambiance chaleureuse qu'ils ont su entretenir tout au long de notre cursus parmi eux et tout particulièrement monsieur le professeur **Zaki SARI**, monsieur le Dr **Ahmed HASSAM**, pour ses constantes disponibilités.

Nous souhaitons enfin remercier toutes nos familles de nous avoir soutenus lors de ces cinq années difficiles malgré la distance et notre manque de présence dans nos régions natales.

INTRODUCTION GENERALE

Une grande partie des problèmes liés aux systèmes de production sont ceux d'ordonnancement et sont souvent de grande complexité et pour les résoudre on fait appel aux méthodes dites approchées vue que ces techniques offre la possibilité de trouver des solutions, généralement, de bonne qualité en un temps raisonnable.

Les métaheuristiques offrent une alternative très intéressante pour la résolution des problèmes classés NP-Difficiles. Plusieurs métaheuristiques ont été étudiées dans la littérature allant de la plus simple recherche locale à des techniques de haut niveau.

A l'inverse de certaines métaheuristiques qui s'inspirent des phénomènes naturels, nous sommes intéressés dans ce travail à une métaheuristique qui s'appelle la recherche d'harmonie (RH) qui s'inspire du processus de recherche de la meilleure harmonie musicale dans un orchestre musical de Jazz, ou chaque musicien joue une note avec différents instruments musicales à la fois pour trouver l'harmonie parfaite. Cette métaheuristique ressemble dans son principe aux algorithmes génétiques inspirés de l'évolution des espèces.

Nous nous sommes intéressés à la résolution du problème de type Flow shop à l'aide de la recherche d'harmonie et de l'algorithme génétique pour pouvoir comparer les résultats obtenus vu la ressemblance des deux techniques.

Ce document, qui résume l'essentiel de notre travail, est organisé comme suit :

Le premier chapitre est consacré aux notions et définitions liées aux problèmes d'ordonnancement dans les systèmes de production.

Le second chapitre donne une présentation générale des méthodes d'optimisation approchées, suivie d'une section consacrée aux principes des métaheuristiques les plus répondues, ensuite nous présentons d'une manière détaillée les deux algorithmes, l'algorithme génétique, et la recherche d'harmonie, que nous avons adapté pour la résolution de notre problème.

Le troisième chapitre représente notre adaptation des deux algorithmes ainsi que leurs études de sensibilité et les résultats de simulations obtenus qui ont montré l'efficacité de la recherche d'harmonie par rapport à l'algorithme génétique.

Enfin, nous terminons par une conclusion qui fait la synthèse du travail présenté et propose d'autres idées comme perspectives.

CHAPITRE 1

L'ORDONNANCEMENT DES ACTIVITES DE PRODUCTION

L'objectif de ce chapitre introductif aux problèmes d'ordonnancement est de faire un cadrage de notre thème. Nous présentons alors, au travers ce chapitre les points essentiels concernant les problèmes d'ordonnancement d'une façon générale : la place de l'ordonnancement en gestion de production, les objectifs d'ordonnancement, définition et modélisation du problème d'ordonnancement, les critères d'optimisation, un formalisme de classification de ces problème et finalement les méthodes de résolution des problèmes d'ordonnancement.

1.1. Introduction

Dans ce chapitre introductif, nous nous intéressons aux problèmes d'ordonnancement d'une manière générale afin de situer la problématique de notre travail. Il s'agit de présenter les concepts et les base du problème d'ordonnancement.

Dans un premier temps, il est nécessaire de replacer l'ordonnancement dans le cadre général des systèmes de production. Nous rappelons alors, dans la deuxième section, des notions de base concernant la production et la gestion de production, en mettant l'accent sur le rôle de l'ordonnancement au sein de ces systèmes. La troisième section vise à présenter le problème de l'ordonnancement en précisant sa définition et les différents éléments qui le déterminent. La quatrième section du chapitre présente les critères d'optimisation comme élément de grande importance. Un formalisme de classification des problèmes d'ordonnancement est évoqué dans la cinquième section. La dernière section est consacrée à la description des différentes méthodes de résolution de problèmes d'ordonnancement.

1.2. Production et gestion de production

La production est le processus conduisant à la création de produits par l'utilisation et la transformation de ressources **[1].** Le processus de production est alors, constitué d'un ensemble d'opérations qui sont les activités conduisant à la création de biens et de services **[2].**

Le système de production, est l'ensemble de ressources réalisant une activité de production. C'est un ensemble de moyens divers : humains, matériels, informationnels et

d'autres, constituant un tout, dont l'objectif est la réalisation de biens ou de services **[3].**

Les systèmes de production ont été classés dans **[4],[5]** en trois grandes catégories :

- Les processus continus : tels que la production électrique ;
- Les processus discrets : tels que l'usinage et toutes les activités d'assemblage. ;
- Les processus discontinus : qui se situe à mi-chemin entre les processus continus et les processus discrets, les deux types de processus sont couplés : la production et continue mais il y a un conditionnement discret des produits.

Un système de production est soumis à une charge qui définit l'ensemble des biens ou des services que le système doit réaliser. Pour écouler cette charge, le système utilise une ou plusieurs ressources **[6].** Plus précisément, la charge d'un système de production est constituée de jobs. Un job suit la réalisation d'un produit dans toutes les étapes de production (de la matière première jusqu'au produit fini). Chaque job a une gamme qui décrit la suite des opérations qu'il doit réaliser. La gamme décrit les opérations à réaliser en donnant :

- Leur durée ;
- La ou les machines qui doivent réaliser cette opération ;
- Les ressources consommées (type de ressource et quantité utilisée) ;
- L'ordre entre les opérations à réaliser (facultatif) **[6].**

1.3. Décomposition du système de production

Classiquement, un système de production peut être en trois sous-systèmes, le système physique de production, le système de décision et le système d'information **[7] [2].**

Cette décomposition est structurée en fonction de la nature des flux qui traversent chaque système i.e. flux de décisions, flux d'informations et flux physique (figure 1.1).

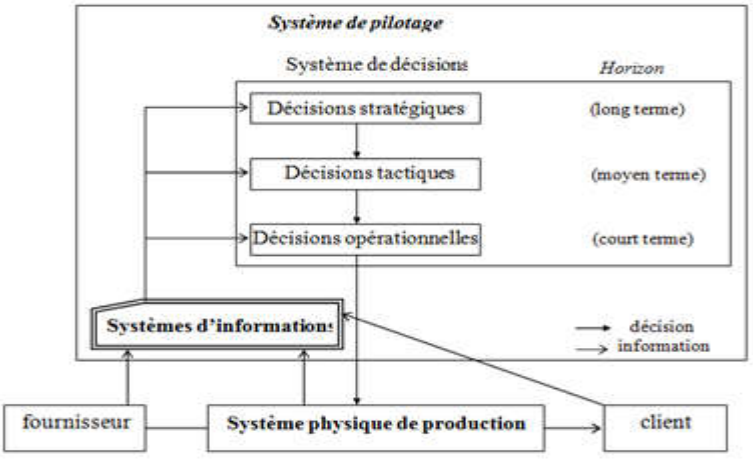


Figure 1.1 : Les sous systèmes constituant le système de production d'après **[3][7].**

1.3.1. Système physique de production : Transforme les matières premières ou composantes en produits finis. Il est constitué de ressources humaines et physiques.

1.3.1.1. Le système de décision : Contrôle le système physique de production. Il en coordonne et organise les activités en prenant des décisions basées sur les données transmises par le système d'information.

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

1.3.2. La gestion de production

Le système de décision "drainé" par le système d'information constitue ce qu'on appelle le système de gestion de production.

1.3.2.1. Définition et rôle de la gestion de production

La gestion de production est « un ensemble de processus qui permet de mener à bien la fabrication de produits à partir d'un ensemble de données et de prévisions » **[8].**

F. Blondel **[9],** définit la gestion de production comme étant « la fonction qui permet de réaliser les opérations de production en respectant les conditions de qualité, délai, coûts qui résultent des objectifs de l'entreprise ».

En fait, la gestion de production s'occupe d'un ensemble de problèmes liés à la production tels que la gestion des données, la planification, le contrôle (suivi) de la production, la gestion des stocks, la prévision, l'ordonnancement etc. (figure1.2).

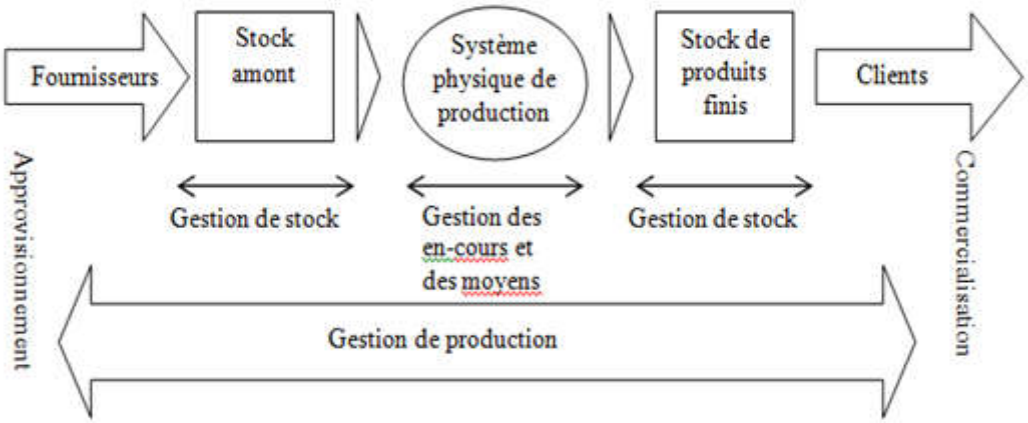


Figure 1.2 : Objectifs de la gestion de production .

1.3.2.2. Organisation hiérarchique de la gestion de production

Les niveaux hiérarchiques de la gestion de production couramment retenus sont en nombre de trois : stratégique, tactique et opérationnel **[1] [8] [3] [7].**

- ✓ **Le niveau stratégique:** Il s'agit de la formulation de la politique à long terme de l'entreprise (à un horizon de plus de deux ans). Elle porte essentiellement sur la gestion des ressources durables, afin que celles-ci soient en mesure d'assurer la pérennité de l'entreprise.
- ✓ **Le niveau tactique:** Il s'agit de décisions à moyen terme. Elles assurent la liaison entre le niveau stratégique et le niveau opérationnel. L'objectif est de produire au moindre coût pour satisfaire la demande prévisible, en s'inscrivant dans le cadre fixé par le plan stratégique de l'entreprise.
- ✓ **Le niveau opérationnel:** Il s'agit des décisions à court et à très court terme. C'est une gestion quotidienne pour faire face à la demande au jour le jour, dans le respect des décisions tactiques.

Parmi les décisions opérationnelles, on trouve : la gestion des stocks, la gestion de la main d'œuvre, la gestion des équipements **[9].**

1.4. Présentation du problème d'ordonnancement

En dépit de la complexité réelle des problèmes d'ordonnancement, il existe un modèle standard sur lequel est basée la définition du problème. Dans ce qui suit, nous nous intéressons à décrire ce modèle théorique du problème d'ordonnancement.

1.4.1. Définition du problème d'ordonnancement

L'ordonnancement de la production a fait ces dernières années l'objet de recherches très approfondies, pour de nombreux problèmes identifiés dont la résolution repose sur une politique d'ordonnancement qui soit la plus optimale possible, chose qui a permis de mettre au point de nombreuses méthodes de résolution.

L'ordonnancement trouve son application dans différents domaines allant du plus simple au complexe, dont on peut citer :

_ Emplois du temps ;
_ Les problèmes d'affectation, de transport ;
_ Organisation des prises de vue d'un satellite ;
_ Les systèmes distribués et les systèmes embarqués ;
_ Dans les ateliers de production.

Etablir un ordonnancement revient donc à coordonner l'exécution de toutes les tâches, en utilisant au mieux les ressources disponibles **[10].** En d'autres termes, il s'agit de :« Déterminer ce qui va être fait, quand, où et avec quels moyens, étant donné un ensemble de taches à accomplir, le problème d'ordonnancement consiste à déterminer quelles tâches doivent être exécutées et à assigner des dates et des ressources à ces tâches de façon à ce que les tâches soient, dans la mesure du possible, accomplies en temps utile, au moindre coût et dans les meilleurs conditions » **[11].**

Donc l'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu de contraintes temporelles et de contraintes liées à la disponibilité des ressources, afin de satisfaire un ou plusieurs critères d'optimisation.

D'après cette définition, on remarque que dans un problème d'ordonnancement quatre éléments fondamentaux interviennent : les tâches, les ressources, les contraintes et les objectifs.

1.4.2. Les tâches

Une tâche est un travail mobilisant des ressources et réalisant un progrès significatif dans l'état d'avancement du projet compte tenu du niveau de détail retenu dans l'analyse du problème **[1].**

D'une façon plus schématique : une tâche qu'on note i est une entité élémentaire de travail localisée dans le temps par une date de début t_i ou de fin c_i, dont la réalisation nécessite une durée p_i telle que $p_i = c_i - t_i$ **[12].**

Généralement trois paramètres caractérisent une tâche **[13]** :

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

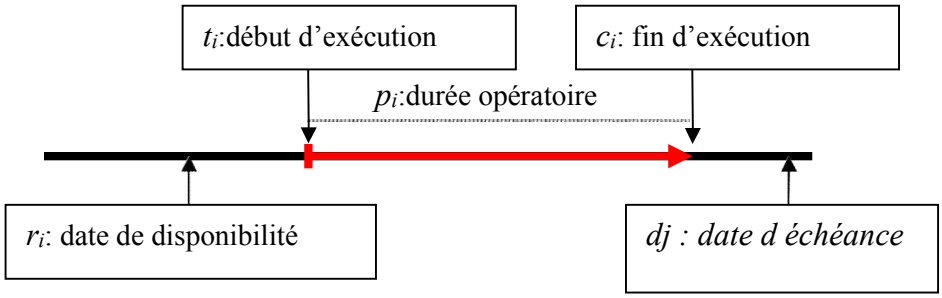


Figure 1.3 : Caractéristiques d'une tâche i.

1.4.3. Les Ressources

Une ressource est un moyen technique ou humain utilisé pour permettre la réalisation des tâches. 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 :

1.4.3.1. Les ressources renouvelables: qui, après avoir été allouées à une tâche, redeviennent disponibles et qui peuvent être réutilisées (machines, personnel, etc.).

1.4.3.2. Les ressources consommables: qui, après avoir été allouées à une tâche, ne sont plus disponibles, et sont donc épuisées (argent, matières premières, etc.).

1.4.3.3. 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 **[14].**

1.4.4. Les contraintes

Les contraintes représentent les conditions à respecter lors de la construction de l'ordonnancement pour qu'il soit réalisable. Elles rendent les problèmes d'ordonnancement plus difficiles car il faut les respecter lors de la résolution de ces problèmes. Ces contraintes peuvent être classées en deux types : endogène et exogène.

1.4.4.1. Les contraintes endogènes

Elles constituent des contraintes liées directement au système de production et à ses performances telles que :

- ✓ Les dates de disponibilité des machines et des moyens de transport ;
- ✓ Les capacités des machines et des moyens de transport ;
- ✓ Les séquences des actions à effectuer ou les gammes des produits.

1.4.4.2. Les contraintes exogènes

Ces contraintes sont imposées extérieurement. Elles sont indépendantes du système de production ; on distingue :

- ✓ Les dates de fin de fabrication au plus tard du produit imposées généralement par les commandes ;
- ✓ Les priorités de quelques commandes et de quelques clients ;
- ✓ Les retards possibles accordés pour certains produits.

Une autre classification consiste à distinguer :

- ✓ Les contraintes de gamme ou de précédence, caractérisant l'ordre d'exécution des tâches selon la gamme ;
- ✓ Les contraintes de capacités, caractérisées par un ensemble de conflits pour l'utilisation des ressources, pouvant se diviser en ;
- ✓ Contraintes disjonctives où une seule ressource est partagée par deux ou plusieurs tâches à la fois ;
- ✓ Contraintes cumulatives où il y'a partage de plusieurs ressources par plusieurs tâches ;
- ✓ Les contraintes liées directement aux produits finis (dates de livraison des commandes, priorités, dates de péremption, etc.) ;

1.4.5. Les objectifs

Tout ordonnancement est guidé par un ou plusieurs objectifs à optimiser. Les objectifs que doit satisfaire un ordonnancement sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement donné **[12]** :

Les objectifs liés au temps : on trouve par exemple, la minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales de réglage ou des retards par rapport aux dates de livraison.

- ✓ Les objectifs liés aux ressources : par exemple, maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches ;
- ✓ Les objectifs liés au coût : ces objectifs sont généralement de minimiser les coûts, de lancement, de production, de stockage, de transport, etc ;
- ✓ Les objectifs liés à l'énergie ou au débit.

Dans la section suivante nous reviendrons avec une description plus poussée sur les différents objectifs que doit satisfaire l'ordonnancement.

1.5. Les problèmes d'atelier multi-machines

Les problèmes d'ordonnancement d'ateliers se décomposent en trois grandes classes de problèmes.

1.5.1. Le type flow-shop

Les ateliers de type « flow-shop » pour lesquels la ligne de fabrication est constituée de plusieurs machines en série ; toutes les opérations de toutes les tâches passent par toutes les machines dans le même et unique ordre. Ce type d'atelier est dit à cheminement unique. Dans les ateliers de type « flow shop hybride », une machine peut exister en plusieurs exemplaires identiques et parallèles.

Dans ce type d'atelier (Figure 1.4), on dispose de *n* pièces qui doivent s'exécuter suivant le même ordre sur les *m* machines qui composent l'atelier.

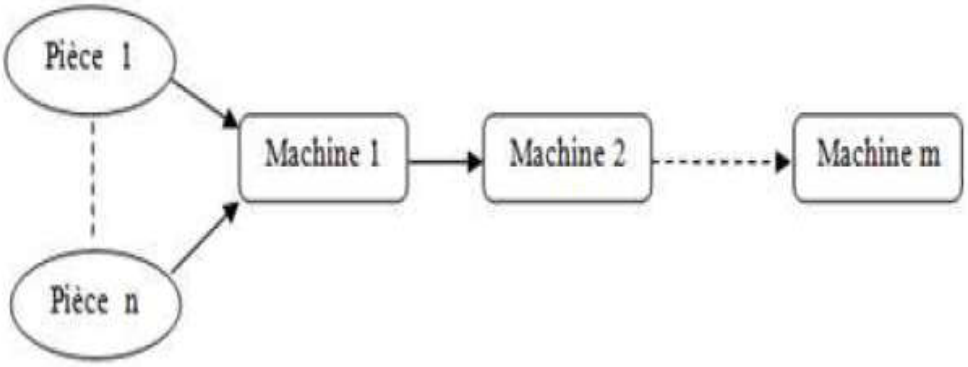


Figure1.4: Exemple d'un Atelier Flow shop.

On distingue quatre types de Flow shop :

- ✓ Flow shop pur : tous les temps opératoires sont positifs ;
- ✓ Flow shop généralisé : les temps opératoires peuvent être nuls si une tâche ne doit pas subir un traitement sur une machine particulière ;
- ✓ Flow shop de permutation : toutes les tâches sont disponibles à l'instant 0, les tâches s'exécutent dans l'ordre défini à l'instant 0, le dépassement n'est pas autorisé ;
- ✓ Flow shop hybride : c'est un atelier Flow shop dans lequel chaque machine est remplacée par un étage composé de plusieurs machines qui ne sont pas forcément identiques et disposées en parallèle. Chaque travail visite successivement chaque étage et ne doit passer que par une seule machine par étage.

1.5.2. Le type job-shop

Dans les ateliers de type « job-shop », les opérations sont réalisées selon un ordre total bien déterminé, variant selon la tâche à exécuter. Ce type d'ateliers est nommé aussi atelier à cheminements multiples. Dans ce cas, plusieurs changements d'outils sont à envisager.

Le job shop flexible est une extension du modèle job shop classique. Sa particularité essentielle réside dans le fait que plusieurs machines sont potentiellement capables de réaliser

un sous-ensemble d'opérations. Plus précisément, une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération.

Dans un atelier Job shop, aucune séquence d'opérations n'est fixe (Figure 1.5), et une pièce peut circuler plusieurs fois sur la même machine pour des tâches différentes.

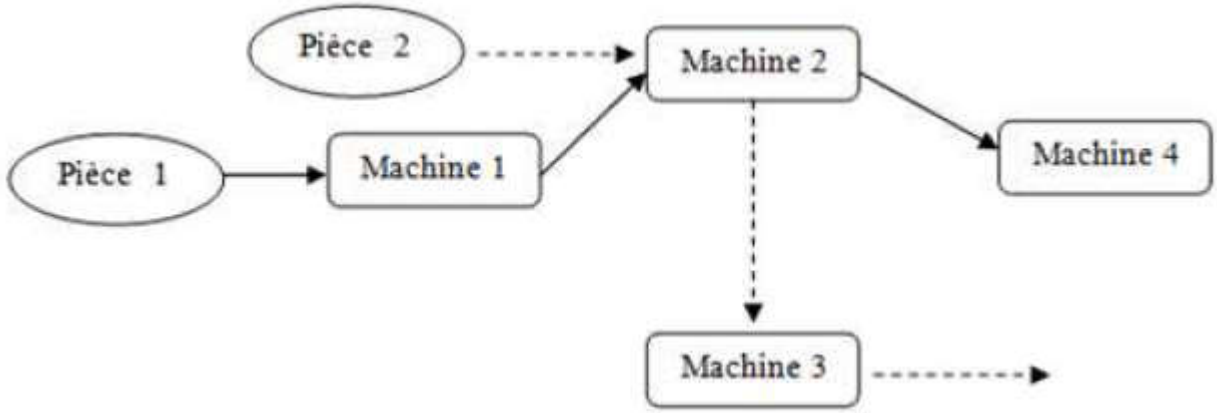


Figure 1.5: Exemple d'un Atelier Job shop

1.5.3. Le type open-shop

Aucun ordre de fabrication n'est imposé, dans ce cas ; le cheminement de toutes les opérations est multiple et libre; ces opérations peuvent être exécutées dans n'importe quel ordre. Des extensions à ces types de base ont été définies dans le but de se rapprocher des ateliers réels de production. Par exemple, la flexibilité des machines qui consiste à associer à chaque opération un ensemble de machines c'est-à-dire que cette opération peut être traitée par une machine quelconque de cet ensemble. Un problème d'affectation est alors ajouté au problème initial d'ordonnancement. Les types suivants sont distingués :

- ✓ Les problèmes à « machines parallèles » où les opérations sont indépendantes et sont traitées par le même ensemble de machines ;
- ✓ Les problèmes de type «flow-shop hybride » constituant une extension du problème du type « flow-shop », où la première opération de chaque produit est traitée par le premier ensemble, la deuxième opération par le deuxième ensemble, et ainsi de suite

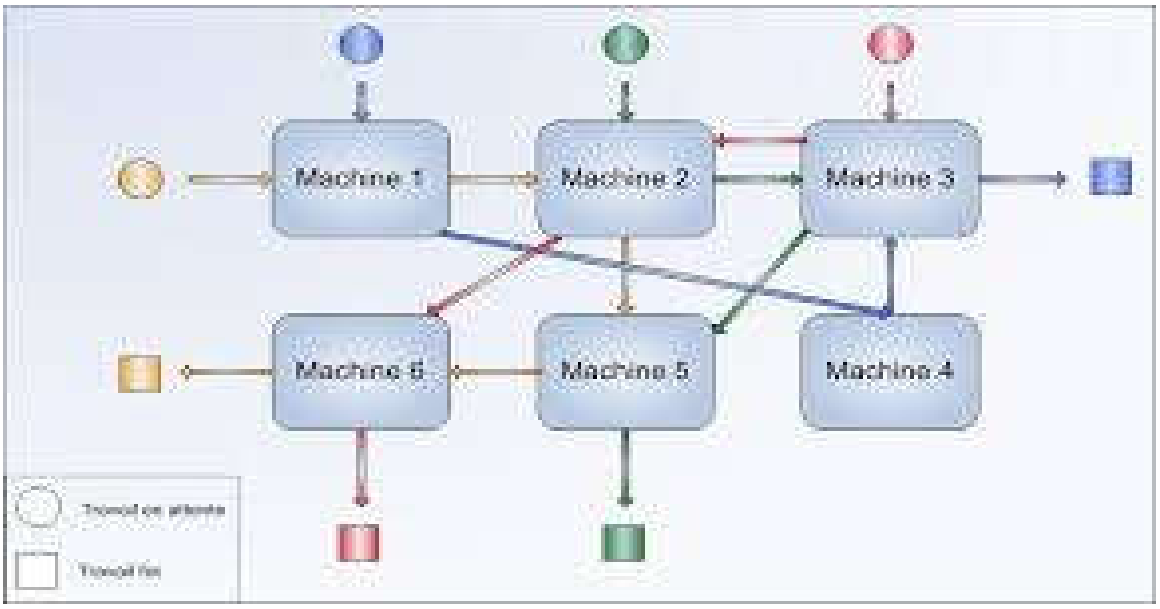


Figure 1.6: Exemple d'un Atelier open shop

1.5.2. Les critères d'optimisation

Dans la résolution d'un problème d'ordonnancement, on peut choisir entre deux grands types d'objectifs, visant respectivement à l'optimalité des solutions (trouver la meilleure solution) ou plus simplement à leur admissibilité (trouver une solution).

La recherche de l'optimalité impose à mesurer la qualité d'une solution de répondre aux exigences du problème étudié. Dans ce but, des critères d'optimisation sont définis et utilisés.

1.5.3. Classification des critères d'optimisation

Les critères que doit satisfaire un ordonnancement sont variés. D'une manière générale, on peut distinguer trois catégories importantes **[15]** :

- ✓ Les critères liés aux dates de fin et de livraison ;
- ✓ Les critères liés aux volumes des encours ;
- ✓ Les critères liés à l'utilisation des ressources.

Avant de dénombrer les critères importants de chacune de ces catégories, rappelons-nous les notations suivantes :

c_i(completion time) : la date d'achèvement de la dernière opération de la tâche i,

d_i(due date) : la date de livraison de$i$,

r_i(release time) : la date de disponibilité de$i$,

p_{ij}: la durée de l'opération j de la tâche i,

$F_i = c_i - r_i$(flow time) : la durée de séjour de la tâche i depuis sa disponibilité jusqu'à la fin de

son exécution,

$L_i = c_i - d_i$(lateness) : le retard algébrique, qui détermine le retard d'exécution de la tâche *i* par rapport à sa date d'échéance,

$T_i = \max(L_i, 0)$ (tardiness) : le retard absolu de la tâche *i*,

$E_i = \max(-L_i, 0)$ (earliness) : l'avancement de la tâche *i*.

1.6. Conclusion

Nous avons abordé dans ce chapitre l'aspect commun des problèmes d'ordonnancement en production, en exposant notamment les notions de base concernant ces problèmes. D'abord, le cadre général de l'ordonnancement comme fonction originale du système de gestion de production a été évoqué, en présentant succinctement les systèmes de production, la gestion de production et le rôle de l'ordonnancement au sein de ces systèmes.

Le deuxième point exposé dans ce chapitre, est la caractérisation du problème d'ordonnancement d'atelier en présentant sa définition, et précisant notamment les différents éléments qui le déterminent : les tâches, les ressources, les contraintes et les critères d'optimisation.

CHAPITRE 2

LES METAHEURISTIQUES

Nous nous intéressons dans ce chapitre aux métaheuristiques, méthodes formant une famille d'algorithmes d'optimisation combinatoire. L'objectif est de présenter les principes généraux de ces algorithmes visant à résoudre les problèmes de type NP-difficiles pour lesquels on ne connait pas de méthodes classiques plus efficaces. Nous commençons par une présentation générale, suivie d'une section consacrée aux principes des métaheuristiques les plus répondues. Enfin nous présentons d'une manière détaillée les deux algorithmes, l'algorithme génétique, et la recherche d'harmonie, que nous avons adapté pour la résolution de notre problème.

2.1. Introduction

Au fil des années, de nombreuses méthodes de résolution de problèmes ont été proposées. Ainsi, une grande variété de concept et principe, de la stratégie et des performances ont été discernées. Cette variété et ces différences ont permis de regrouper les différentes méthodes de résolution de problèmes NP-difficiles en deux classes principales : la classe de méthodes exactes et la classe des méthodes approchées **[16].**

La résolution d'un problème d'ordonnancement, de type NP-difficiles (taille comparable à ceux rencontrés dans la pratique), se heurte à des tailles de mémoire et de temps de calcul trop importants. L'objectif n'est plus alors d'obtenir systématiquement l'optimum mais plutôt d'obtenir une solution proche de l'optimum ou de « bonne qualité » en un temps minimal. Ainsi, au lieu d'effectuer une recherche exhaustive, les méthodes approchées échantillonnent l'espace de recherche et n'en considèrent qu'une partie, et fournissent ainsi, en un temps raisonnable, la meilleure configuration rencontrée. On distingue deux types de méthodes : les heuristiques et les métaheuristiques.

Contrairement à une méthode exacte, qui vise à l'obtention d'une solution optimale, l'objectif d'une heuristique est de trouver une « bonne » solution en un temps raisonnable. Les heuristiques sont beaucoup plus adaptées au contexte industriel, où il ne s'agit pas tant de satisfaire complètement tel ou tel objectif que de fournir une solution d'ordonnancement réaliste, qui procure une satisfaction jugée acceptable d'un ensemble d'objectifs. Pour cette raison, les logiciels d'ordonnancement disponibles sur le marché font tous appel à des méthodes heuristiques **[17].**

Les Métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonne qualité en un temps raisonnable à des problèmes combinatoires réputés difficiles pour lesquels on ne connait pas de méthode classique plus efficace **[18].**

Ce chapitre est consacré aux métaheuristiques, nous allons présenter celles les plus connues, leurs origines, leurs principes de base, et leurs algorithmes, ainsi que les deux approches que nous avons adapté dans notre travail, à savoir, l'algorithme génétique, et la recherche d'harmonie.

2.2. Généralités sur les méthodes approchées

Le but d'une méthode approchée est de trouver une solution réalisable en tenant compte de la fonction objectif mais sans garantie d'optimalité. Son utilisation offre de multiples avantages par rapport à une méthode exacte, citons :

Elles sont plus simples et plus rapides à mettre en œuvre quand la qualité de la solution n'est pas trop importante **[19].**

Elles sont plus souples dans la résolution des problèmes réels. En pratique, il arrive souvent que l'on doive prendre en considération de nouvelles contraintes qu'on ne pouvait pas formuler dès le départ. Ceci peut être fatal pour une méthode exacte si les nouvelles contraintes changent les propriétés sur lesquelles s'appuyait la méthode **[19].**

Elles fournissent des solutions et des bornes qui peuvent être utiles dans la conception de méthodes exactes **[19].**

Depuis toujours, les chercheurs ont tenté de résoudre les problèmes NP-difficiles le plus efficacement possible. Pendant longtemps, la recherche s'est orientée vers la proposition d'algorithmes exacts pour des cas particuliers polynomiaux **[20].**

2.3. Les Heuristiques

Nombreuse sont les définitions proposées aux heuristiques.On peut citer quelques-unes :

- Pour FEIGENBAUM « Une méthode heuristique est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillibles de ce qui est la meilleure chose à faire» **[21]** ;
- Pour SLAGLE « Une heuristique est une règle d'estimation, une stratégie, une méthode ou astuce utilisée pour améliorer l'efficacité d'un système qui tente de découvrir les solutions des problèmes complexes**.»[22]** ;
- Pour PEARL « Les heuristiques sont des critères, des méthodes ou des principes pour décider qui, parmi plusieurs d'autres plans d'action promet d'être le plus efficace pour atteindre un certain but » **[23]** ;

- Pour NEWELL « Les heuristiques sont des règles empiriques et des morceaux de connaissances, utiles (mais non garanties) pour effectuer des sélections différentes et des évaluations» **[24]** ;
- Pour SOLSO « Les heuristiques sont des ensembles de règles empiriques ou des stratégies qui fonctionnent, en effet, comme des règles d'estimation » **[25].**

Les heuristiques sont donc en optimisation combinatoire, un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée.

Les heuristiques peuvent être classées en deux catégories : des méthodes constructives qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue et des méthodes de fouilles locales qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaye d'améliorer cette solution en explorant son voisinage **[26].**

De nombreuses méthodes heuristiques ont été proposées dans la littérature pour résoudre les problèmes d'ordonnancement d'atelier. On distingue **[27][28][29]** :

- ✓ FIFO (First In First Out) : la première tâche qui vient est la première tâche ordonnancée ;
- ✓ SPT (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu**[16]** ;
- ✓ LPT (Longest Processing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu **[16]** ;
- ✓ EDD (Earliest Due Date) : cet algorithme choisit parmi les tâches exécutables celle dont le délai est échu le plus tôt. Si aucune tâche n'est disponible, alors un temps libre est généré**[16]** ;
- ✓ SRPT (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs**[16]** ;
- ✓ ST (Slack Time): à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative **[16].**

2.3. Les métaheuristiques

Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, les métaheuristiques ont fait leur apparition. Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage**[26].**

Le rapport entre le temps d'exécution et la qualité de la solution trouvée d'une Métaheuristique reste alors dans la majorité des cas très intéressant par rapport aux différents types d'approches de résolution **[26].**

La plupart des métaheuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à des problèmes comme l'explosion combinatoire **[26].**

Une métaheuristique peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné **[26].**

Plusieurs d'entre elles sont souvent inspirées par des systèmes naturels dans de nombreux domaines tels que : la biologie (algorithmes évolutionnaires et génétiques) la physique (recuit simulé), et aussi l'éthologie (algorithmes de colonies de fourmis) **[26].**

Un des enjeux de la conception des métaheuristiques est donc de faciliter le choix d'une méthode et le réglage des paramètres pour les adapter à un problème donné **[26].**

Les métaheuristiques peuvent être classées de nombreuses façons. On peut distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois. Les méthodes qui tentent itérativement d'améliorer une solution sont appelées méthodes de recherche locale ou méthodes de trajectoire. Ces méthodes construisent une trajectoire dans l'espace des solutions en tentant de se diriger vers des solutions optimales. Les exemples les plus connus de ces méthodes sont : La recherche Tabou et le Recuit Simulé. Les algorithmes génétiques, l'optimisation par essaim de particules et Les algorithmes de colonies de fourmis présentent les exemples les plus connus des méthodes qui travaillent avec une population **[26].**

Donc les problèmes d'optimisation combinatoire comme le problème de flow shop, sont pour la majorité NP-difficiles, et ne possèdent pas à ce jour de solutions algorithmiques efficaces valables pour toutes les données **[30].** Le recours à des méthodes approchées ou

heuristiques les plus populaires, et également les plus efficaces **[31],** destinées à la résolution des problèmes d'optimisation combinatoire, sont des techniques générales, appelées Métaheuristiques, qu'il s'agit d'adapter à chaque problème particulier.

2.3.1. Définition de Métaheuristique

Le mot métaheuristique est dérivé de la composition de deux mots grecs: Heuristique qui vient du verbe heuriskein (euriskein) et qui signifie 'trouver', META qui est un suffixe signifiant 'au-delà', 'dans un niveau supérieur'.

Selon la définition proposée par Osman « Une Métaheuristique est un processus itératif qui subordonne et guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales » **[32].**

Pour résumer ces définitions, on peut dire que les propriétés fondamentales des Métaheuristiques sont les suivantes.

- Les Métaheuristiques sont des stratégies qui permettent de guider la recherche d'une solution optimale ;
- Le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions (presque) optimales ;
- Les techniques qui constituent des algorithmes de type métaheuristique vont de la simple procédure de recherche locale à des processus d'apprentissage complexes ;
- Les métaheuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité ;
- Les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche ;
- Les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, sans faire appel à un problème spécifique ;
- Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur ;
- Les Métaheuristiques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche **[33].**

L'ensemble des métaheuristiques proposées dans la littérature sont partagées en deux catégories : des métaheuristiques à base de solution unique et des métaheuristiques à base de

population de solutions. Nous présentons dans ce qui suit quelques métaheuristiques des deux catégories **[34].**

2.3.2. Les Métaheuristiques à base de solution unique

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

De nombreuses méthodes à base de solution unique ont été proposées dans la littérature. Parmi lesquelles : la descente, le recuit simulé, la recherche tabou, la recherche à voisinage variable (VNS: Variable Neighbourhood Search), la recherche locale réitérée (ILS: Iterated Local Search), la recherche locale guidée (GLS: Guided Local Search)…etc **[34].**

2.3.2.1. Le Recuit Simulé (SimulatedAnnealing)

Le recuit simulé (SA) a été introduit par (Kirkpatrik et al. 1983) et (Cerný 1985) comme une méthode de recherche locale normale, utilisant une stratégie pour éviter les minima locaux.

Cette Métaheuristique est basée sur une technique utilisée depuis longtemps par les métallurgistes qui, pour obtenir un alliage sans défaut, faisant alterner les cycles de réchauffage (ou de recuit) et de refroidissement lent des métaux. Le recuit simulé s'appuie sur des travaux faits par (Metropolis et al. 1953), qui ont pu décrire l'évolution d'un système en thermodynamique.

Le principe du recuit simulé est de parcourir de manière itérative l'espace des solutions. On part avec une solution notée s_0 initialement générée de manière aléatoire dont correspond une énergie initiale E_0, et une température initiale T_0 généralement élevée. A chaque itération de l'algorithme, un changement élémentaire est effectué sur la solution, cette modification fait varier l'énergie du système. Si cette variation est négative (la nouvelle solution améliore la fonction objective, et permet de diminuer l'énergie du système), elle est acceptée. Si la solution trouvée est moins bonne que la précédente alors elle sera acceptée avec une probabilité P calculée suivant la distribution de Boltzmann suivante **[26]**donner par l'équation 2.1:

$$P(E, T) = exp^{\frac{-\Delta E}{T}} \qquad 2.1$$

Au début de la recherche, la température élevée permet au recuit d'explorer l'espace de

recherche sans être piégé par les optima locaux. A mesure que la température décroît, le recuit est de moins au moins capable d'explorer l'espace de recherche, et la recherche se concentre sur une zone déterminée : l'exploration prend le dessus sur l'exploration. Par conséquent la vitesse de décroissance de la température détermine la vitesse à laquelle le recuit « converge » vers une solution **[35].**

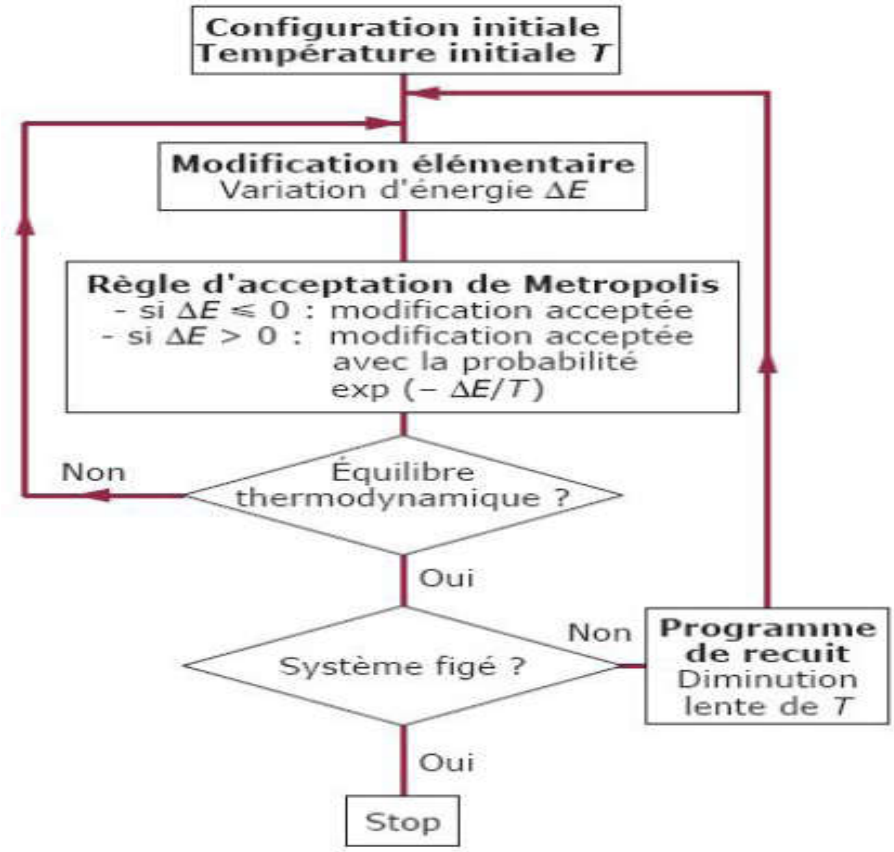


Figure 2.1 : Organigramme de l'algorithme recuit simulé

Le choix de la température est primordial pour garantir l'équilibre entre l'intensification et la diversification des solutions dans l'espace de recherche. Premièrement, le choix de la température initiale dépend de la qualité de la solution de départ. Si cette solution est choisie aléatoirement, Il faut prendre une température relativement élevée.

On utilise souvent la règle suivante :

$T_{k+1} \leftarrow T_k . \alpha$ où $\alpha \in [0, 1]$, un paramètre qui exprime la diminution de la température de l'itération k à k + 1.

La décroissance de la température peut également être réalisée par paliers (en d'autres termes, elle ne change qu'après un certain nombre d'itérations).

Certains préconisent l'utilisation de stratégies non monotones. On peut ainsi rehausser la température lorsque la recherche semble bloquée dans une région de l'espace de recherche.

On peut alors considérer une grande augmentation de la température comme un processus de diversification alors que la décroissance de la température correspond à un processus d'intensification **[34].**

La méthode du recuit simulé a l'avantage d'être souple vis-à-vis des évolutions du problème et facile à implémenter et contrairement aux méthodes de descente elle évite le piège des optima locaux mais en contre partie elle nécessite de nombreux tests nécessaires pour trouver les bons paramètres et la définition de voisinages permettant un calcul efficace de ΔE **[34].**

2.3.2.2. La recherche Tabou (Tabu Search)

La recherche tabou (TS) est une méthode de recherche locale combinée avec un ensemble de techniques permettant d'éviter d'être piégé dans un minimum local ou la répétition d'un cycle. La recherche tabou est introduite principalement par Glover (Glover 1986), Hansen (Hansen 1986),Glover et Laguna dans (Glover et Laguna 1997). Cette méthode a montré une grande efficacité pour la résolution des problèmes d'optimisation difficiles. En effet, à partir d'une solution initiale *s* dans un ensemble de solutions local S, des sous-ensembles de solution N(s) appartenant au voisinage S sont générés. Par l'intermédiaire de la fonction d'évaluation nous retenons la solution qui améliore la valeur de f, choisie parmi l'ensemble de solutions voisines N(s).

L'algorithme accepte parfois des solutions qui n'améliorent pas toujours la solution courante. Nous mettons en œuvre une liste tabou (tabulist) T de longueur k contenant les k dernières solutions visitées, ce qui ne donne pas la possibilité a une solution déjà trouvée d'être acceptée et stockée dans la liste tabou. Alors le choix de la prochaine solution est effectué sur un ensemble de solutions voisines en dehors des éléments de cette liste tabou **[26].**

Quand le nombre k est atteint, chaque nouvelle solution sélectionnée remplace la plus ancienne dans la liste. La construction de la liste tabou est basée sur le principe FIFO, c'est-à-dire le premier entré est le premier sorti. Comme critère d'arrêt on peut par exemple fixer un nombre maximum d'itérations sans amélioration de s*, ou bien fixer un temps limite après lequel la recherche doit s'arrêter **[26].**

La recherche tabou est une méthode de recherche locale, et la structure de son algorithme de base est proche de celle du recuit simulé, avec l'avantage d'avoir un paramétrage simplifié : le paramétrage consistera d'abord à trouver une valeur indicative t d'itérations pendant lesquelles les mouvements sont interdits. Il faudra également choisir une

stratégie de mémorisation. En revanche, la méthode tabou exige une gestion de la mémoire de plus en plus lourde en mettant des stratégies de mémorisation complexe. L'efficacité de la méthode tabou offre son utilisation dans plusieurs problèmes d'optimisation combinatoire classiques tels que le problème de voyageur de commerce, le problème d'ordonnancement, le problème de tournées de véhicules, etc **[26].**

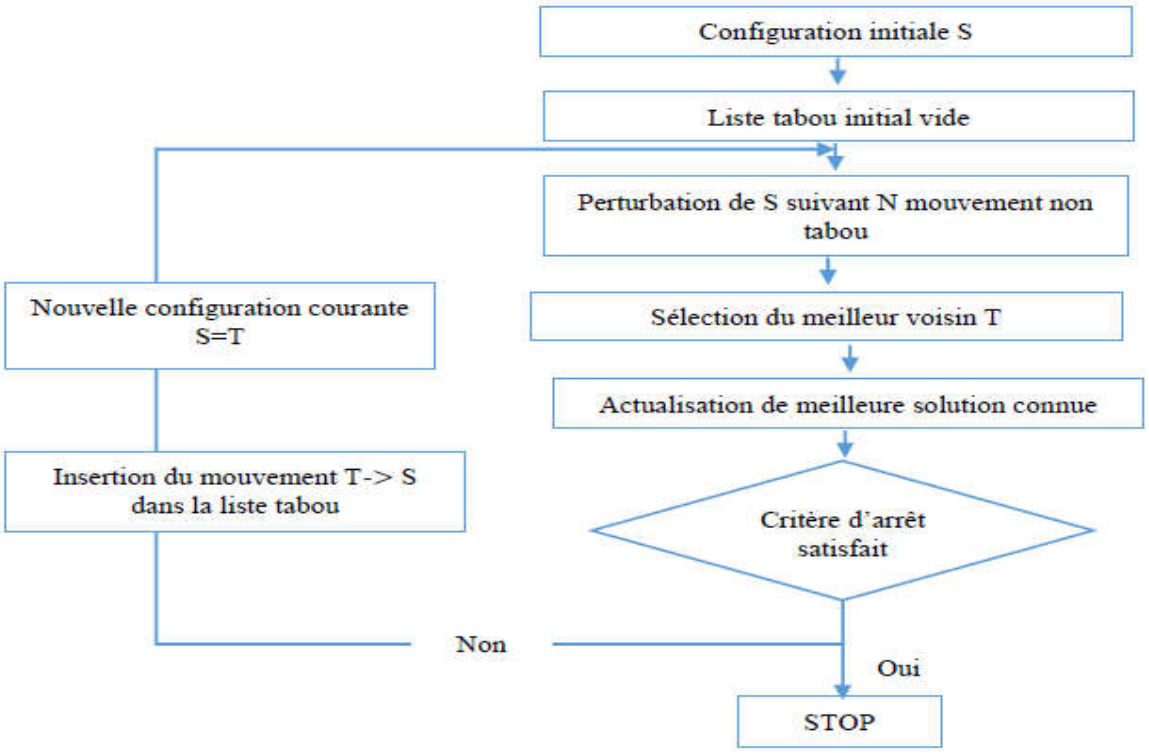


Figure 2.2 : Organigramme de l'algorithme recherche tabou

2.3.3. Les métaheuristiques à base de population de solutions

Les métaheuristiques à base de population de solutions débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure (l'optimum global) qui représentera la solution du problème traité. L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité **[34].**

2.3.3.1. Les Colonies de Fourmis (Ant Colony Optimization)

L'optimisation par l'approche colonie de fourmis est relativement récente. Le principe de l'optimisation est apparu au début des années 90. Il est le fruit des travaux de recherche de M. Dorigo, V. Maniezzo et A. Colorni. **[36].**

Les fourmis sont capables de résoudre collectivement des problèmes complexes. Pour cela, elles communiquent entre elles de façon locale et indirecte. Au cours de leur

progression, les fourmis déposent une trace de phéromone ; elles choisissent ensuite leur chemin, selon une probabilité dépendant de la quantité de phéromone précédemment déposée par les autres fourmis.

L'exemple présenté dans la figure, montre la capacité d'adaptation des fournis. **[37].** Au départ, les fourmis disposent d'un chemin pour aller de leur nid, la fourmilière, à la source de nourriture

Un obstacle est placé entre le nid et la source de nourriture, les fourmis vont alors commencer par se séparer de part et d'autre de l'obstacle en déposant de la phéromone sur leur passage, figure 2.3. (b). Les fourmis les plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui ont emprunté les deux branches les plus courtes à l'aller et au retour. Ainsi la quantité de phéromone présente sur le plus court trajet est plus importante que celle présente sur le chemin le plus long.

Finalement la quantité de phéromone sur le chemin le plus court fait que ce chemin finit par être emprunté par la plupart les fourmis.

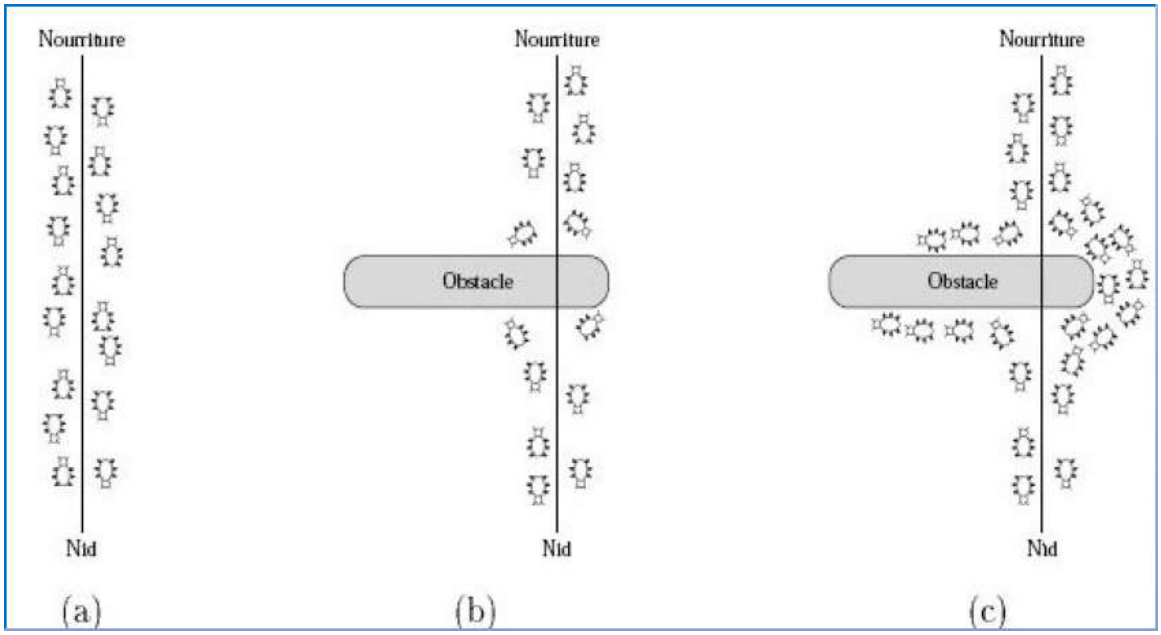


Figure 2.3 : Schéma de comportement des fourmis

Ant system (AS) est le premier algorithme de fourmi reposant sur le comportement de fourragement des fourmis **[36].** Cet algorithme a pour but de reproduire le comportement naturel des fourmis pour retrouver le meilleur chemin possible vers un objectif donné. On peut résumer ce mécanisme par l'algorithme suivant :

Algorithme : Algorithme de colonie de fourmis
(1) **Initialiser**
(2) **Répéter**
(3) **Pour** chaque fourmi **faire**
(4)**Construire** une solution basée sue une procédure de construction, dépendant de la trace de phéromone
Mettre à jour la trace de phéromone en se basant sur la qualité de la solution trouvée
(6) Fin pour
(7) **Jusqu'à** satisfaction d'un critère d'arrêt

Algorithme 2.1 : Algorithme de colonie de fourmis

Beaucoup de travaux ont été proposés pour résoudre le problème d'ordonnancement d'atelier nous citons : pour une seule machine, machines parallèle ,atelier type flow shop atelier type job shop, atelier type open shop, atelier flow shop flexible et le job shop flexible.

2.3.3.2. Les essaims particulaires (Particle Swarms Optimization)

L'Optimisation par essaim particulaire (OEP) est une méthode née aux Etats –Unis en 1995 sous le nom de Particle Swarms Optimization (PSO).Ces concepteurs à l'origine cherchaient à modéliser des interactions sociales entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information. La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Ce comportement fait rappeler particulièrement celui des essaims d'abeilles, du fait qu'une abeille ayant trouvé un site promoteur sait informer certaine de ses 'consœurs' et que celles –ci vont tenir compte de cette information pour leur prochain déplacement **[38][39][40]**.

Le fonctionnement de l'OEP fait que c'est une méthode qui peut être rangée dans les méthodes itératives (on approche peu à peu de la solution) et stochastiques (un fait appel au hasard) **[34].**

Le principe de l'algorithme est : au départ un essaim est reparti au hasard de l'espace de recherche chaque particule a une vitesse aléatoire également, ensuite à chaque pas de temps :

Chaque particule est capable d'évaluer la qualité de la position qu'elle a atteind jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser) **[34].**

Chaque particule est capable d'interroger un certain nombre de ses congénères (ses informatrices, dont elle- même) et d'obtenir de chacune d'entre elles sa propre meilleure performance (et la qualité afférente).

Chaque particule choisit la meilleure des meilleures performances dont elle a connaissance, modifie sa vitesse en fonction de cette information et de ces propres données et se déplace en conséquence**[34].**

Figure 2.4 : Schéma de principe du déplacement d'une particule

Les principales étapes de l'algorithme d'optimisation par essaims particulaires sont données par la figure 2.5 **[34].**

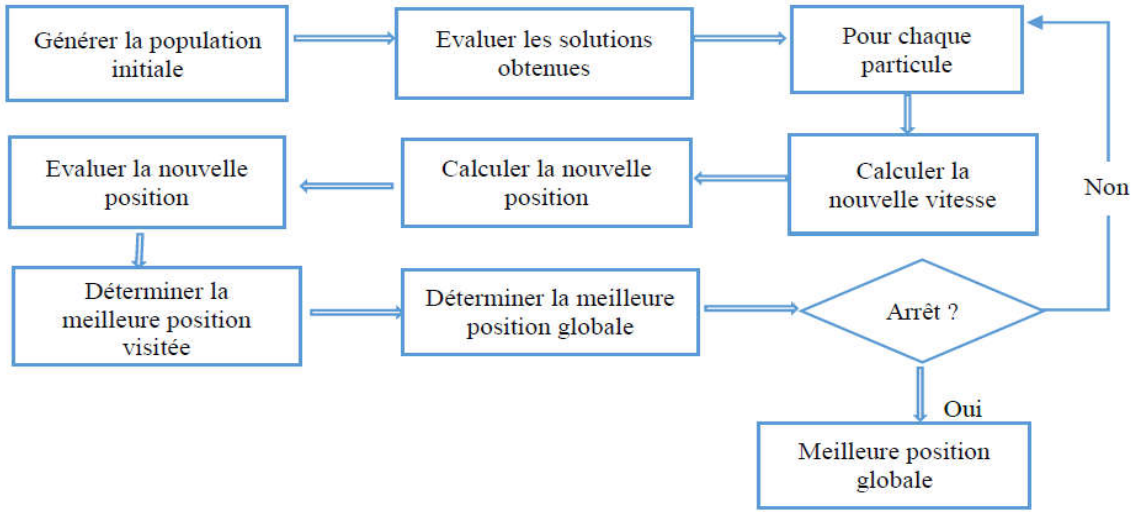


Figure 2.5 : Organigramme décrit l'algorithme d'optimisation par essaims particulaires

2.3.3.3. L'algorithme de la recherche coucou

La recherche coucou a été proposée en 2009 par Yang et Deb sous le nom de CuckooSearch (CS) **[35].** Elle s'inspire du comportement de reproduction d'une espèce spéciale d'oiseaux parasites de nids appelés « Coucous ».

Dans l'algorithme de la recherche coucou, une solution possible est appelée « nid » ou « coucou ». En fait, la recherche coucou part du principe que chaque nid comporte un seul coucou. Au cours du processus de la recherche de l'algorithme CS, chaque coucou crée son propre poussin en fonction de sa représentation actuelle (le coucou lui-même) et du vol de Lévy. L'évaluation de la qualité du poussin et de son père permet de sélectionner lequel d'entre eux survivra et subira quelques modifications dans le but d'améliorer sa qualité.

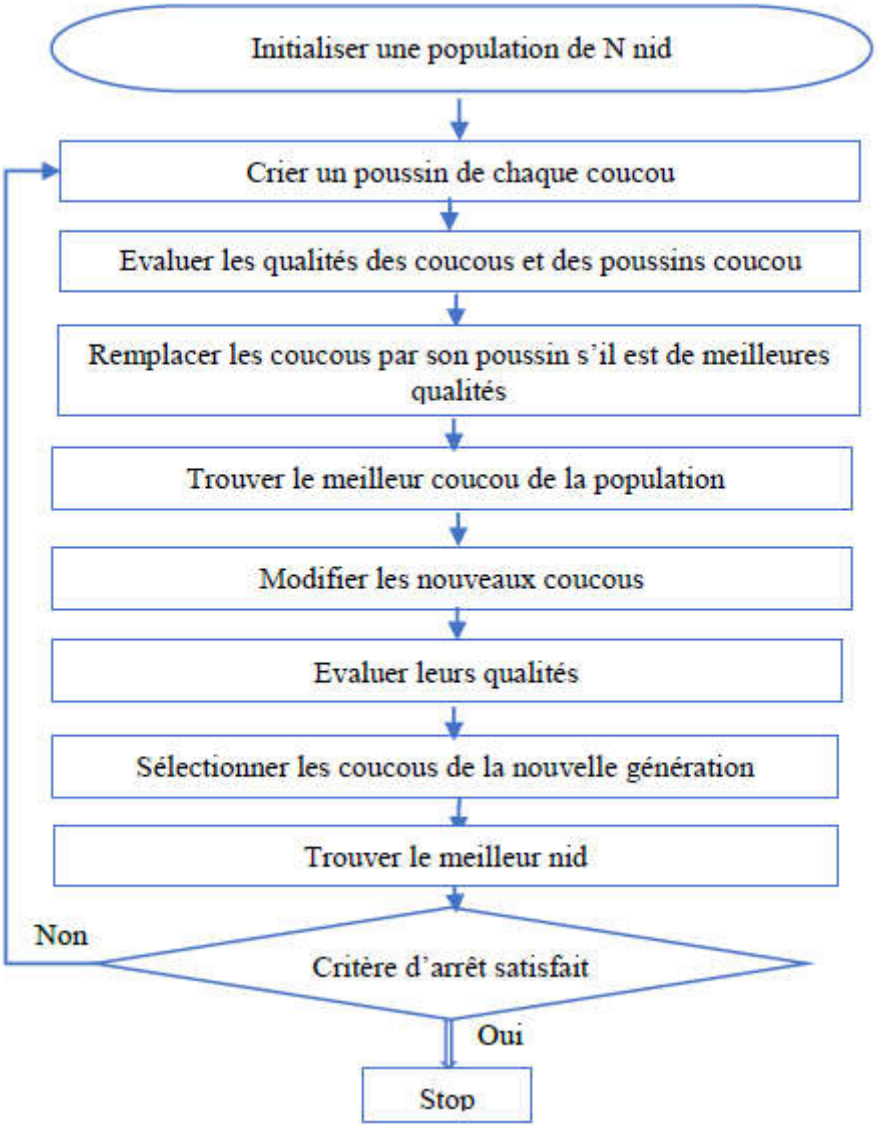


Figure 2.6 : Organigramme décrit l'algorithme de la recherche coucou

2.4. Les algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils ont été adaptés à l'optimisation par John Holland (Holland 1975), également les travaux de David Goldberg ont largement contribué à les enrichir **[41]**

Le vocabulaire utilisé est le même que celui de la théorie de l'évolution et de la génétique, on emploie le terme individu (solution potentielle), population (ensemble de solutions), génotype (une représentation de la solution), gène (une partie du génotype), parent, enfant, reproduction, croisement, mutation, génération, etc.

Leur fonctionnement est extrêmement simple, on part d'une population de solutions potentielles (chromosomes) initiales, arbitrairement choisies.

On évalue leur performance (Fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre. On recommence ce cycle jusqu'à ce qu'on trouve une solution satisfaisante. En effet, l'héritage génétique à travers les générations permet à la population d'être adaptée et donc répondre au critère d'optimisation, illustre les principales étapes d'un algorithme génétique. Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Son mise en œuvre nécessite :

2.4.1. Le codage des données

La première étape est de définir et coder convenablement le problème. Cette étape associe à chaque point de l'espace de recherche une structure de données spécifique, appelée génotype ou ensemble de chromosomes, qui caractérisera chaque individu de la population.

Le codage de chaque individu en séquence est essentiel dans l'élaboration d'un algorithme génétique dont dépend notamment l'implémentation des opérateurs de transformations. Ainsi, cette phase détermine la structure de données qui sera utilisée pour coder le génotype des individus de la population. Le codage doit donc être adapté au problème traité.

Plusieurs types de codages sont utilisés dans la littérature, les premiers résultats théoriques sur les algorithmes génétiques ont opté pour un codage par une séquence binaire de longueur fixe à travers la notion de schéma (Goldberg 1989a). L'efficacité de l'algorithme génétique dépend donc du choix convenable du type de codage.

2.4.2. Génération de la population initiale

La génération de la population initiale, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer, ce choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme.

Néanmoins, une initialisation aléatoire est plus simple à réaliser : les valeurs des gènes sont tirées au hasard selon une distribution uniforme. Toutefois, il peut être utile de guider la génération initiale vers des sous domaines intéressants de l'espace de recherche.

Par exemple lors d'une recherche d'optima dans un problème d'optimisation sous contraintes, il est préférable de produire des éléments satisfaisant les contraintes. La population initiale doit être suffisamment diversifiée et de taille assez importante pour que la recherche puisse parcourir l'espace d'état dans un temps limité.

2.4.3. Fonction d'adaptation (Fitness)

L'évaluation de la Fitness est généralement l'étape dans laquelle on mesure la performance de chaque individu. Pour pouvoir juger la qualité d'un individu et ainsi le comparer aux autres, il faut établir une mesure commune d'évaluation. Aucune règle n'existe pour définir cette fonction, son calcul peut ainsi être quelconque, que ce soit une simple équation ou une fonction affine. La manière la plus simple est de poser la fonction d'adaptation comme la formalisation du critère d'optimisation.

2.4.4. Sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais, pendant le passage d'une génération à une autre, ce processus est basé sur la performance de l'individu. L'opérateur de sélection doit être conçu pour donner également une chance aux mauvais éléments, car ces éléments peuvent, par croisement ou mutation, engendrer une descendance pertinente par rapport au critère d'optimisation. Il existe différentes techniques de sélection, on propose quelques-unes.

2.4.4.1. Sélection uniforme

On ne s'intéresse pas à la valeur d'adaptation fitness et la sélection s'effectue d'une manière aléatoire et uniforme telle que chaque individu i a la même probabilité Prob(i) = $1/T_{pop}$ comme tous les autres individus (T_{pop} est la taille de la population).

2.4.4.2. Sélection par tournoi

Deux individus sont choisis au hasard, on compare leurs fonctions d'adaptation et le mieux adapté est sélectionné.

2.4.4.3. Élitisme

Cette méthode de sélection permet de favoriser les meilleurs individus de la population. Ce sont donc les individus les plus prometteurs qui vont participer à l'amélioration de notre population.

On peut constater que cette méthode induisait une convergence prématurée de l'algorithme.

2.4.5 .Croisement

L'opérateur de croisement favorise l'exploration de l'espace de recherche et enrichit la diversité de la population en manipulant la structure des chromosomes, le croisement fait avec deux parents et génère deux enfants, en espérant qu'un des deux enfants au moins héritera de bons gènes des deux parents et sera mieux adapté qu'eux.

Il existe plusieurs méthodes de croisement par exemple le croisement en un point, ou en multiples points voir les figure

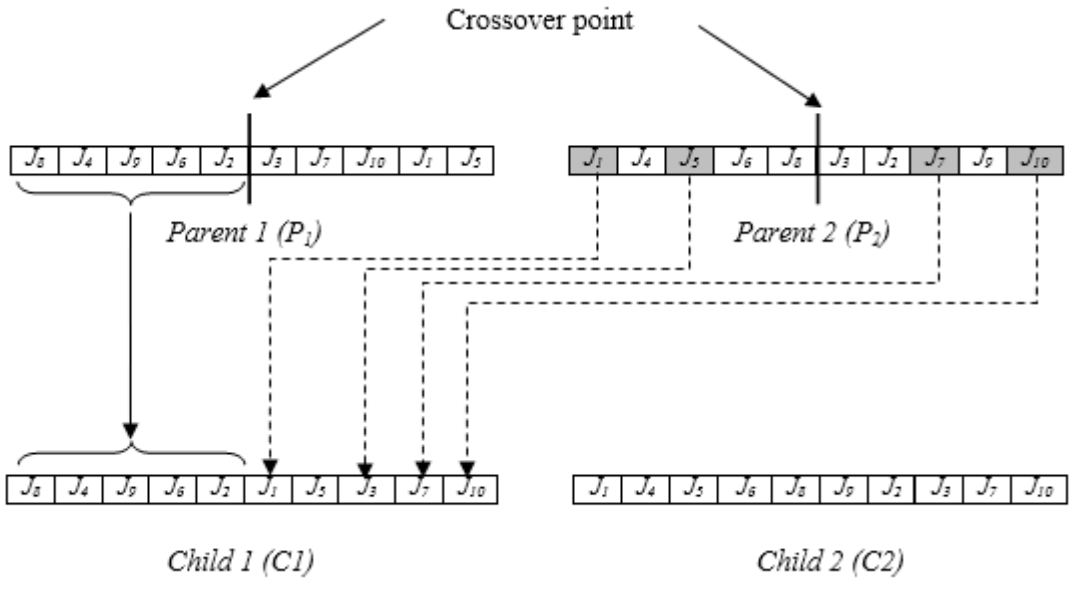


Figure 2.7 : Schéma représetative de croisement

2.4.5. Mutation

La mutation permet de transformer au hasard le codage d'un individu afin d'apporter une certaine diversité dans la population et empêcher que celle-ci converge trop vite vers un seul type d'individu parfait, incapable de sortir d'un minimum local. La mutation est réalisée en modifiant un gène d'un individu pris au hasard. Dans les Algorithmes Génétiques, la mutation est considérée comme un opérateur secondaire par rapport au croisement. Parmi les stratégies de mutation utilisées en pratique:

a. **La mutation uni-point :**
Cette mutation se fait par altération d'une seule valeur sur le chromosome.

b. **La mutation bi-points et multi-points :**
Cette mutation se fait par altération de plusieurs valeurs sur le chromosome.

c. **La mutation par valeurs :**
Avec ce type, la mutation se fait par transformation d'une valeur donnée en une autre valeur déterminée, sur tous les gènes de chromosome.

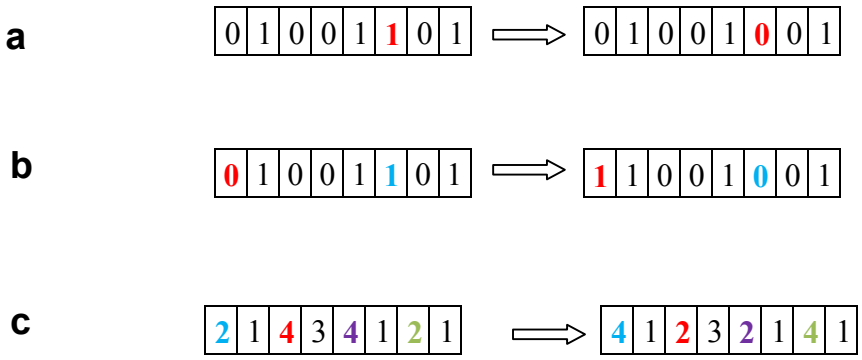


Figure 2.8 : Diféférentes possibilités de mutation

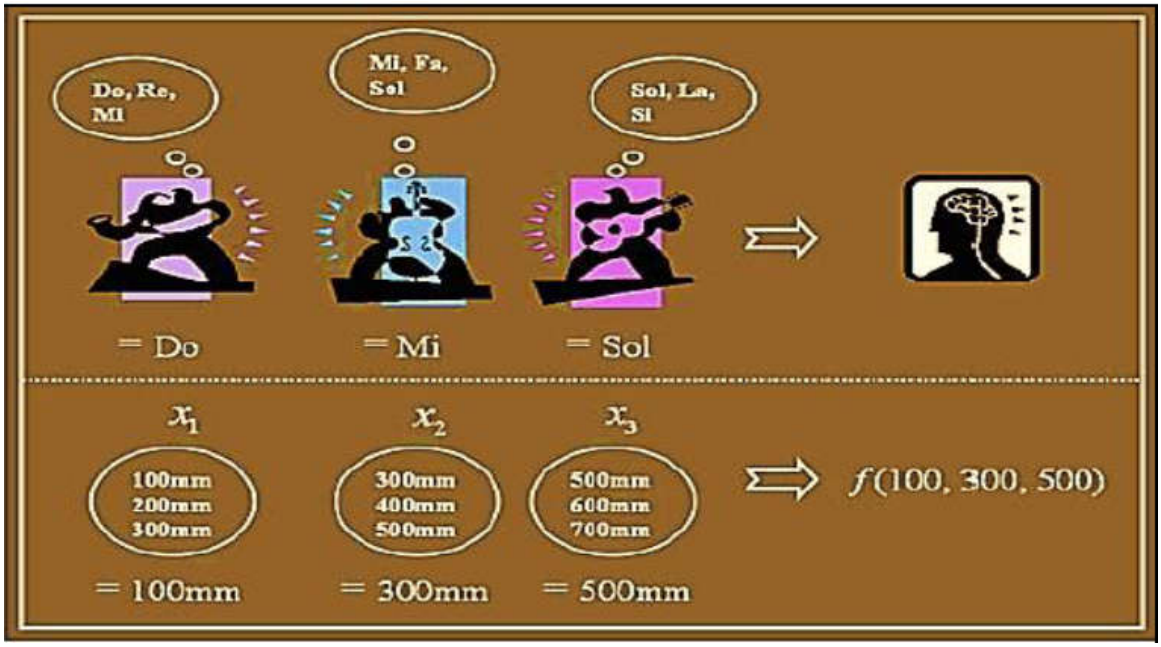
2.5. L'algorithme de la recherche d'harmonie (Harmony search)

A l'inverse de certaines métaheuristiques qui s'inspirent des phénomènes naturels, nous sommes intéressés à une métaheuristique qui s'appelle la recherche d'harmonie (RH), en anglais (Harmony Search), proposée par Geem et ses collègues 2001, cette recherche s'inspire du processus de recherche de la meilleure harmonie musicale dans un orchestre musical de Jazz, ou chaque musicien joue une note avec des différents instruments musicales à la fois pour trouver l'harmonie parfaite. L'algorithme RH a été appliqué avec succès sur multiples problèmes comme le problème du voyageur de commerce, le problème de tournée de véhicule , la conception de réseau d'eau et le problème de conception des Sudoku puzzle, le carré magique, l'imagerie médicale, l'analyse de données astronomiques nature**[42]**.

2.5.1. Analogie entre la musique d'improvisation et l'optimisation

La figure 2.9 montre les détails de l'analogie entre l'improvisation de musique et l'optimisation technique, ou chaque musicien guitariste, contrebassiste, saxophoniste peut correspondre à chaque variable de décision dont la gamme de chaque instrument de musique sont le saxophone (DO, Ré , Mi); le contrebasse (Mi, Fa, SOL); et la guitare (Sol, La, Si). Si le saxophoniste joue la note de Do, le bassiste cueille Mi, et le guitariste cueille Sol, leurs notes forment ensemble une nouvelle harmonie (Do, Mi, Sol). Si cette nouvelle harmonie est mieux que l'harmonie existante, la nouvelle harmonie est stockée dans la mémoire de chaque joueur. Le nouveau vecteur de solution est maintenu s'il est meilleur que l'harmonie existante en termes de valeurs de la fonction objectif, la qualité de l'harmonie est améliorée par la pratique après l'entraînement

De même dans l'optimisation de l'ingénierie, chaque variable de décision x_i correspond à une note qui sort de l'instrument i, une harmonie correspond à un vecteur de solution, et l'esthétique musicale correspond à la fonction objective à maximiser ou à minimiser.



La figure 2.9 : Montre les détails de l'analogie entre l'improvisation de musique

L'algorithme comprend six étapes essentielles **[42] :**

Etape 1 : initialisation des paramètres

Dans cette étape, les paramètres de l'algorithme sont initialisés:

La taille de la mémoire d'harmonies (population de solutions) notée par HMS (de l'anglais : Harmony Memory Size), elle varie généralement de 1 à 100.

Le taux de sélection ou de considération de la mémoire harmonique notée par HMCR (de l'anglais: Harmony Memory Considering Rate) avec HMCR ∈ [0, 1], c'est le taux d'élire ou de sélectionner une valeur de la mémoire d'harmonie, elle varie généralement de 0,7 à 0,99**[42]**.

Le taux d'ajustement notée par PAR (de l'anglais: Pitch Adjusting Rate), représentant la probabilité d'apporter quelques modifications à un élément de la mémoire d'harmonie avec PAR ∈[0,1].

Le critère d'arrêt NI (nombre d'improvisations ou itérations) généralement représente un nombre maximum de recherches **[42]**

Etape 2 : génération des solutions initiales (appelé mémoire de l'harmonie HM)

La mémoire d'harmonie est une matrice de solutions de taille HMS ou chaque vecteur représente une solution du problème traité, c'est l'équivalent d'une harmonie dans la music. Dans cette étape la matrice HM est remplie aléatoirement, les valeurs générées seront stockées dans cette matrice, et pour chaque solution i (i=1,.., HMS) la fonction objectif fi est calculée

La structure générale de la mémoire d'harmonie est représente par la matrice suivante :

$$HM = \begin{pmatrix} X_1^1 & \cdots & X_n^1 \\ \vdots & \ddots & \vdots \\ X_1^{HMS} & \cdots & X_n^{HMS} \end{pmatrix}$$

Etape 3 : improvisation d'une nouvelle harmonie (solution) à partir de la HM

C'est l'étape la plus importante de l'algorithme ou un nouveau vecteur est généré en s'appuyant sur deux règles idéalisée : la considération de la mémoire HM, l'ajustement de lancement et le choix aléatoire

a) La considération de la mémoire

En utilisant le paramètre HMCR qui est défini comme la probabilité de choisir une valeur historique stockée dans HM, chaque variable x_i(i=1,.., n) du nouveau vecteur est choisi aléatoirement du vecteur ((y_1) i, …, (yHMS) i) de la matrice HM, alors que
(1-HMCR) est déterminé comme la probabilité d'élire aléatoirement une valeur à partir de la plage possible des valeurs, c'est l'équivalent du choix aléatoire.

Si ce taux est trop faible, seulement quelques meilleures harmonies seront choisies pour la nouvelle improvisation et peuvent converger trop lentement et si c'est extrêmement haut (près de 1), pratiquement tous les harmonies sont utilisées dans HM.
Par exemple si le HMCR est égal à 0.90, ça indique que l'algorithme choisira une valeur d'une variable avec une probabilité de 90% à partir de la HM, alors que (1-HMCR) qui est

égal à (100%-90%) est défini comme la probabilité de choisir aléatoirement une valeur à partir de la gamme possible des valeurs.

b) L'ajustement de pitch

Tout élément du nouveau vecteur d'harmonie obtenu à partir de l'étape de la considération de mémoire est testé pour déterminer s'il devrait être modifié et ajusté pour le lancer; cette opération utilise le taux PAR qui représente la probabilité d'ajustement de lancement.

Le nouvel élément du nouveau vecteur X nouveau (i) égal au élément i du vecteur élu est qui présente le vecteur sélectionné à partir de la mémoire d'harmonie plus un pourcentage ∈ € [0,1] multiplié par l'élément i du vecteur voisin est qui présente un voisin dans le rang i du élément élu.

D'où X nouveau (i)=X élu (i) + ∈* X voisin (i).

Etape 4 : la mise à jour de mémoire d'harmonie ou le remplacement

Si la nouvelle solution (nouveau vecteur) est faisable et meilleure que la plus mauvaise solution dans la matrice HM en termes de valeur de fonction objectif alors le nouveau vecteur (nouvelle harmonie) est inclus dans HM et la plus mauvaise harmonie existante est exclue de HM.

La Figure (2.10) montre la notion de remplacement dans un processus d'improvisation.

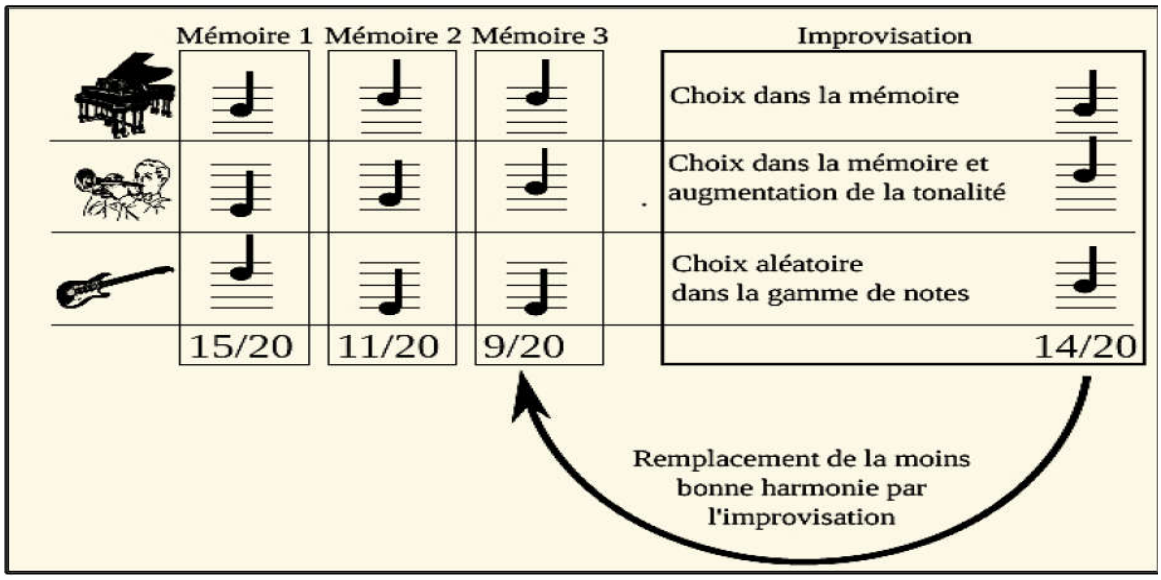


Figure 2.10 : La notion de remplacement dans un processus d'improvisation

Etape 5 : vérification du critère d'arrêt

L'amélioration et la mise à jour seront refaites jusqu'à la satisfaction du critère d'arrêt sinon l'étape de création de la population initiale et l'étape de génération de nouveaux solutions seront répétées

L'algorithme de la recherche d'harmonie est présenté ci-dessous sous forme d'un organigramme

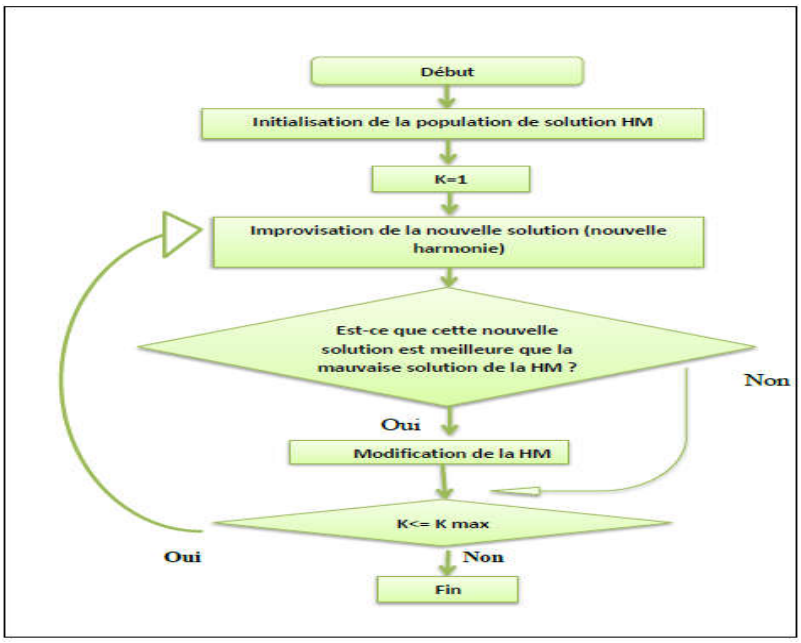


Figure 2.11:L'organigramme de la recherche d'harmonie

2.5.2. Les avantages de la recherche d'Harmonie [42]

L'algorithme de recherche d'harmonie a prouvé sa capacité dans de nombreuses applications grâce à multiples avantages parmi eux celle cités dans **[42]** :

- Facile à mettre en œuvre ;
- Possède une grande capacité de trouver l'optimum global ;
- Il peut échapper les optimums locaux ;
- Il peut gérer des variables discrètes ainsi que les variables continues ;
- Les utilisateurs novices peuvent facilement utiliser l'algorithme ;
- Certaines variantes du HS ne nécessitent pas de paramètres de l'algorithme tels que HMCR et PAR.

2.6. Conclusion

Dans ce chapitre nous avons décrit les principales métaheuristiques, leurs origines, principes de fonctionnement, et leurs algorithmes de bases. De plus nous avons présenté les deux méthodes que nous avons adaptés pour la résolution de notre problème, à savoir, l'algorithme génétique et l'algorithme et l'algorithme de la recherche d'harmonie.

Le chapitre suivant est consacré à l'adaptation de ces deux algorithmes pour la résolution d'un problème d'ordonnancement de type Flow Shop, à la présentation d'une étude de sensibilité du deuxième algorithme, et aux résultats obtenus et leurs interprétations.

CHAPITRE 3

Adaptation de l'algorithme génétique et la recherche d'harmonies et résultats de simulation

Ce chapitre a pour objectif la résolution d'un problème d'ordonnancement de type Flow Shop à l'aide de deux métaheuristiques proposées, à savoir l'algorithme génétique et la recherche d'harmonie. Nous présentons donc, le principe et l'adaptation de deux techniques proposées et une comparaison entre les deux méthodes en terme de la manière par laquelle chacune d'entre des deux explore l'espace de recherche. Pour valider notre adaptation nous proposons leurs applications et simulation sur différentes classes de problèmes Flow Shop avec différents tailles.

3.1. Introduction

Les ateliers de type flow shop sont une production linéaire, caractérisée par une séquence d'opérations identiques pour tous les produits. L'objectif de ce travail est d'optimiser le makespan Cmax (Cmax =max {Cj ; j=1,……., n}, où Cj est la dates de fin d'exécution du taches) par deux Métaheuristiques, à savoir l'algorithme génétique et la recherche d'harmonie pour avoir les meilleurs solutions.

Dans le but de résoudre le problème d'ordonnancement de type flow shop proposé, nous présentons dans ce chapitre, de manière synthétique le principe et l'adaptation de ces deux méthodes, les résultats obtenus et la comparaison entre ces deux techniques.

Ce chapitre est subdivisé en trois section, nous consacrons les deux premières à l'adaptation des deux méthodes retenues, nous présentons respectivement la résolution du problème par l'algorithme génétique et par la recherche d'harmonie ainsi que les différents paramètres de chaque méthode et nous terminons le chapitre par la présentation des résultats de simulation, leur interprétation.

3.2. Adaptation de l'algorithme génétique

Les Algorithmes Génétiques sont considérés par plusieurs chercheurs comme une méthode bien adaptée au problème de type Flow Shop, même si elle ne peut pas arriver à l'optimum dans certains cas difficiles.

Notre application porte uniquement sur l'Algorithme Génétique standard, c-à-d, la version de base caractérisé par une population d'individus générés aléatoirement, un opérateur de sélection, un opérateur de croisement appliqué à un point, et un opérateur de mutation.

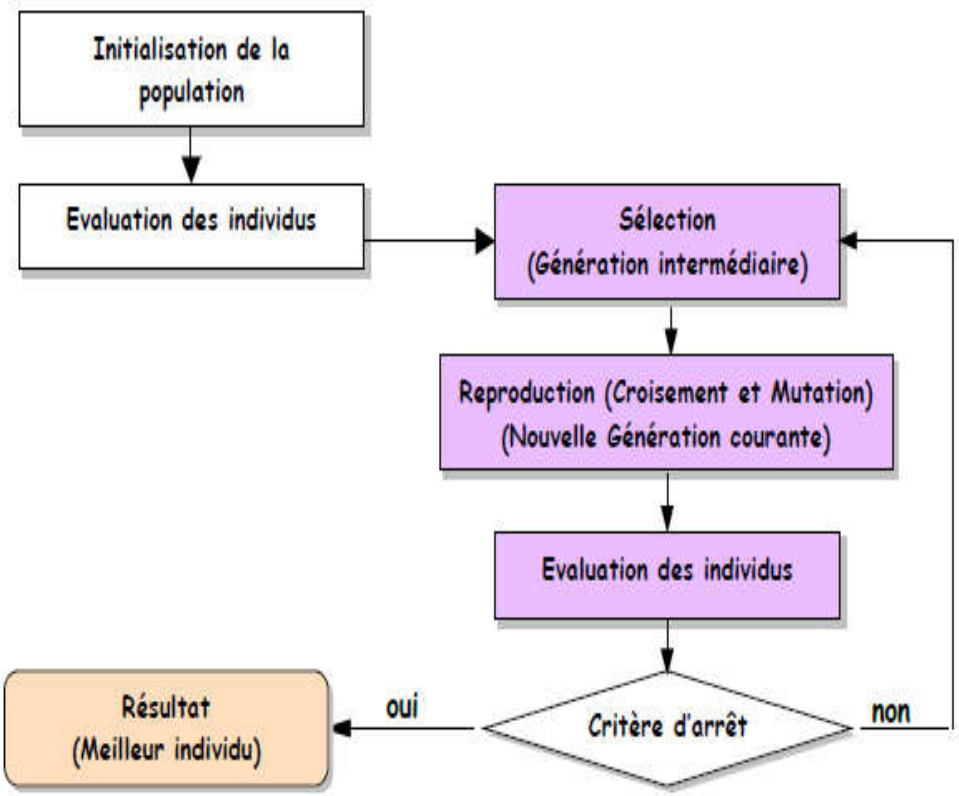


Figure 3.1 : Organigramme général de l'Algorithme Génétique implémenté.

3.2.1. Différents paramètres de l'algorithme génétique

La mise en œuvre de notre Algorithme Génétique passe par :

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

3.2.1.1. Le codage

Le codage est le déterminant important de l'efficacité de la méthode **[43][44].**Dans un problème d'ordonnancement il signifie la transcription d'un ordonnancement réel en représentation adéquate permettant la réalisation des différents opérateurs génétiques. Par conséquent, les deux mots "codage" et "représentation" seront employés dans cette section

dans le même sens.

En conséquence, chaque chromosome généré, qui représente en même temps une solution possible, correspond impérativement à un ordonnancement réel c'est-à-dire une séquence possible pour les différents jobs passant par les différentes machines.

Dans ce travail, nous adoptons le codage basé sur les tâches (Job Based). Dans ce codage, chaque gène dans un chromosome donné représente un job. La figure (3.1) donne un exemple de codage proposé dans notre travail.

J1	J3	J2	J4	J5

Figure 3.1: Exemple d'un codage d'un chromosome

Remarque : un chromosome dans un algorithme génétique représente une solution possible du problème, il est nommé aussi « individu ».

3.2.1.2. La sélection

La sélection est appliquée afin de favoriser au cours du temps les individus les mieux adaptés, à les faire se produire (duplication). Elle consiste à choisir deux individus (solutions) dits « parents » en but de créer deux autres individus dits « enfants ».

La sélection peut être effectuée de différentes façons (citées dans le chapitre2), dans notre approche, nous avons opté pour la sélection par roulette, c'est-à-dire les individus sont choisi deux à deux successivement.

3.2.1.3. Le croisement

Le croisement est un opérateur très important dans les Algorithmes Génétiques. Cet opérateur est appliqué sur chaque deux parents sélectionnés. Il peut être appliqué de différentes façons, on trouve le croisement à un point et à deux points. Nous utilisons le croisement à un point, qui consiste à subdiviser chacun des chromosomes considérés (les deux parents) en deux en vue de construire deux nouveaux enfants.

La manière dont on applique l'opérateur de croisement peut différer d'un problème à un autre. La figure (3.2) représente la technique de croisement utilisée généralement dans les problèmes de types Flow shop avec un codage basé sur tâches.

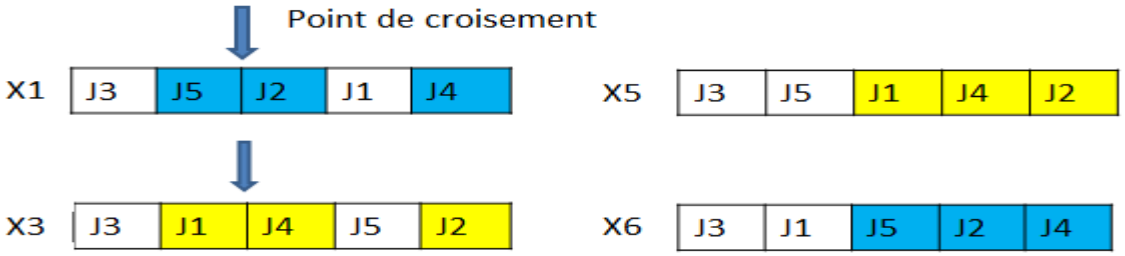


Figure 3.2: Exemple de l'approche de croisement utilisé dans notre cas.

3.2.1.4. La mutation

Le rôle de la mutation est d'apporter une certaine diversité à la population et d'empêcher que celle-ci converge trop vite vers un seul type d'individu. Généralement, la mutation suit deux grands principes : mutation de valeur et mutation de position.

Il consiste à faire, généralement, une petite modification au niveau de chaque chromosome, dans la plupart des cas cette modification est un inter-changement de deux ou plusieurs gènes. Dans notre étude nous avons choisi le cas standard de mutation basé sur la permutation entre deux gènes choisis aléatoirement le long du chromosome considéré.

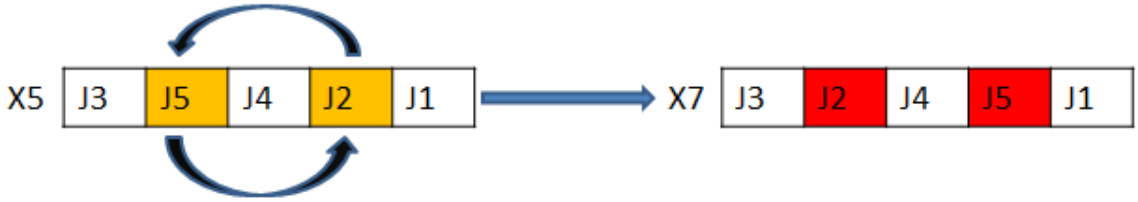


Figure 3.3: Principe de mutation par permutation de tâches.

3.2.2. La population initiale

Avant de lancer l'Algorithme Génétique, une population initiale doit être générée. Cet ensemble d'individus qui servira de base pour les générations ultérieures doit être non homogène. Afin de satisfaire cette non homogénéité, la génération de la population initiale dans notre application se fait aléatoirement.

En générant des chromosomes aléatoires au lieu d'ordonnancements réels. Cette approche est exploitable pour tous les codages implantés.

3.2.3. L'évaluation des individus

Le seul critère que nous avons utilisé pour l'évaluation des individus est le makespan. Sur ce critère, s'appuie l'estimation de fitness des individus. L'évaluation de fitness des chromosomes se fait en calculant le makespan qui représente la fonction objectif (Cmax).

3.3. Adaptation de la recherche d'harmonie

A l'opposé des autres métaheuristiques qui s'inspirent des phénomènes naturels, la recherche par harmonies s'inspire du processus de recherche de la meilleure harmonie musicale. Les étapes du processus de recherche de l'algorithme de recherche d'harmonies sont résumées dans l'algorithme 3.4.

Début:

Initialiser les paramètres nécessaires;

Initialiser la mémoire HM (une population d'harmonies);

Tant que la condition d'arrêt n'est pas satisfaite faire

Produire une nouvelle solution en améliorant la solution *i x*;

Fin Tant que

Retourner la ou les meilleures solutions;

Fin

Algorithme.3.1 :L'algorithme général de la recherche d'harmonies

L'algorithme HS commence par une étape d'initialisation des paramètres nécessaires et de la mémoire d'harmonies (population de solution) composée d'un ensemble de 1 à HMS harmonies (i.e. solutions) aléatoires et des paramètres nécessaires pour le fonctionnement du HS qui sont:

la taille de la mémoire d'harmonies (i.e. la population), notée par HMS (de l'anglais: Harmony Memory Size). Dans notre étude elle représente le nombre de solutions générées.

Le taux de considération de la mémoire harmonique, noté par HMCR (de l'anglais: Harmony Memory Considering Rate), dont le rôle est de décider si la mémoire HM sera utilisée ou non, il est compris entre 0.1 et 0.9.

Le paramètre PAR (de l'anglais: Pitch Adjusting Rate), représentant la probabilité d'apporter quelques modifications à un élément de la HM, nous l'avons fixé à 0.1 pour avoir une certaine diversité mais ne pas diverger.

Le critère d'arrêt (généralement un nombre maximum d'itérations).

Ensuite, l'algorithme passe à l'étape d'amélioration des harmonies. Cette étape consiste

à améliorer une solution *X'i=X'1,X'2,….X'n*en se basant sur trois règles: la considération de la mémoire HM, l'ajustement des valeurs des variables de la solution et la sélection aléatoire.

Après génération du nouveau vecteur (nouvelle solution)*X'i=X'1,X'2,….X'n*, les composantes obtenues par considération de la mémoire HM sont examinées pour décider s'ils devront être ajustées ou non. Suit à l'étape d'amélioration de solutions, le HS passe à l'étape de mise à jour de la mémoire HM. Cette étape consiste à remplacer la mauvaise solution de la matrice HM par la nouvelle solution trouvée si cette dernière est de meilleure qualité (comparée avec la mauvaise solution). Les étapes d'amélioration et de mise à jour seront répétées jusqu'à la satisfaction du critère d'arrêt.

3.3.1. Différents paramètres de la recherche d'harmonie

3.3.1.1. Initialisation de la population

Dans cette étape, une population de solutions est générée aléatoirement afin d'avoir un espace large de solution. Nous avons opté pour la génération aléatoire des solutions pour avoir une diversité dans l'espace de recherche.

3.3.1.2. Génération de nouvelles solutions

Une fois la population de solutions est généré, l'algorithme calcule pour chaque vecteur (solution)la valeur de Cmax, et la stocke. Les solutions sont ensuite triées dans un ordre croissant. Ensuite, un nouvel individu sera créé en se basant sur 2 étapes intéressantes : la première étape appelée « sélection par le taux HMCR », si ce taux est faible donc le nombre d'individus sélectionnés de HM est faible, sinon si ce taux est trop grand (près de 1) alors un nombre important d'individus de cette population sera choisi d'une manière aléatoire.

3.3.1.3. Modification de la population

Si la nouvelle solution (le nouveau vecteur) crée est meilleure que la plus mauvaise solution dans la matrice HM en termes de valeur de fonction objectif alors le nouveau vecteur est inclus dans la HM et le plus mauvais individu existant est exclu du HM.

3.3.1.4. Vérification du critère d'arrêt

Dans notre contribution, le critère d'arrêt présente un nombre d'itérations fixé au début, tant que ce critère n'est pas atteint alors l'algorithme ne s'arrête pas.

3.5. Comparaison entre les principes des deux techniques

Nous remarquons qu'il y a une certaine ressemblance entre les deux méthodes en termes

de certains paramètres. Les deux algorithmes génèrent initialement un nombre de solutions d'une façon aléatoire. Les deux algorithmes utilisent un opérateur de diversité (l'opérateur de mutation dans l'AG et l'opérateur PAR dans la HS) en se basant sur le même principe qui est la modification d'une petite partie de la solution considérée (ce qui revient à la modification d'un job dans la séquence considérée dans notre cas).

La manière dont les deux méthodes examinent l'espace de recherche diffère dans le fait que l'AG fait une interaction entre les individus en appliquent l'opérateur de croisement tandis que la HS examine l'espace de recherche d'une manière différente, en faisant un classement de solutions selon leur fitness et en les décomposant en deux sous populations à l'aide du paramètre HMCR.

Les critères d'insertion utilisés par les deux méthodes diffèrent. Dans l'AG un enfant est inséré s'il est meilleur que son parent c'est-à-dire la comparaison se fait entre les enfants et leurs parents tandis que dans la HS l'insertion se fait si la solution obtenue est meilleur que la plus mauvaise solution dans la HM.

3.6. Résultats de simulation

Pour valider notre adaptation nous avons effectué plusieurs simulations sur un nombre de classes de problèmes Flow Shop.

Les simulations sont effectuées sur 4 classes de problèmes en faisant varier dans chaque classe le nombre de jobs et de machines (10jobs/10machines, 20jobs/30machines, 50jobs/50machines, 100jobs/20machines) et pour chaque classe nous considérons 3 exemples différents (les temps de traitement de pièces diffèrent d'un exemple à l'autre et sont pris dans un intervalle de [1 :30]).

3.6.1. Etude de sensibilité de l'algorithme génétique et la recherche d'harmonie

L'analyse de sensibilité consiste à étudier la sensibilité de l'algorithme suite à la variation de ses paramètres. Nous effectuons cette étude pour 3 exemples de classe 10 jobs/ 10 machines.

L'étude de sensibilité de l'AG est effectuée par rapport à sa taille de population, tandis que pour la HS elle est effectuée par rapport à la HMCR.

La taille de population de la HS est fixée à la meilleure valeur obtenue pour calle de l'AG.

Algorithme génétique					**Recherche d'harmonie**				
Exemple 1					**Exemple 1**		**Taille de population= 30**		
Taille de population	**30**	**40**	**50**	**60**	**HCMR**	**0.2**	**0.5**	**0.7**	**0.9**
Cmax	**320**	326	325	331	**Cmax**	334	326	**320**	334
Exemple 2					**Exemple 2**		**Taille de population= 60**		
Cmax	391	370	376	**368**	**Cmax**	386	384	**374**	381
Exemple 3					**Exemple 3**		**Taille de population= 30**		
Cmax	**342**	353	355	365	**Cmax**	**357**	362	376	360

Tableaux 3.1 : Analyse de sensibilité des deux algorithmes pour 3 exemples de 10 machines 10 jobs

Le tableau (3.1) montre les résultats de simulation de l'AG et ceux de la HS. Nous avons fixé le taux de mutation, pour l'AG, à 0,1 et en faisant varier la taille de population. Les résultats montrent que le Cmax minimum est obtenu pour une taille de population égale à 30 pour les exemples 1 et 3 et pour une taille de population égale à 60 pour l'exemple 2, c'est ce qui nous a conduit à fixer la taille de population de l'algorithme à 30 pour la suite de notre étude (nous n'avons pas pris la taille 60 vu que cette classe de problème n'est pas NP-difficile, c'est-à-dire on peut calculer la solution minimale en balayant tout l'espace de recherche, donc si on considère une taille de population très grande ceci peut conduire à des duplications des solutions initiales générés aléatoirement et donc une mauvaise qualité des solutions obtenus) .

En ce qui concerne l'algorithme HS, nous avons fixé la taille de population à 30 pour pouvoir comparer les résultats de cette technique avec ceux de l'AG et l'étude de sensibilité a été faite sur le HCMR (en le variant de 0,2 à 0,9). Le tableau (3.1) montre que les meilleurs résultats sont obtenus pour un HCMR = 0,7 pour les exemples 1 et 2 et un HCMR égale à 0,2. Ceci peut être expliqué par le fait que plus le paramètre PAR est grand plus ça offre une possibilité de bien examiner l'espace de recherche, ceci est du au fait que l'opérateur PAR offre une diversité à l'espace de recherche.

3.6.2. Résultats de simulation pour les différentes classes de problèmes

Dans cette section nous présentons les différents résultats de simulation pour les différentes classes de problèmes considérées.

	10jobs×10machines		**20jobs×30machines**		**50jobs×50machines**		**100jobs×20machines**	
	AG	**RH**	**AG**	**RH**	**AG**	**RH**	**AG**	**RH**
Exemple 1	**320**	**320**	1025	**1016**	2145	**2143**	3488	**2300**
Exemple 2	391	**374**	1014	**963**	2143	**2127**	3428	**2390**
Exemple 3	**342**	376	1044	**993**	2212	**2169**	3444	**2386**

Tableau 3.2: Comparaison des résultats de simulation entre les deux techniques

Le tableau (3.2) illustre les résultats de simulation de l'AG et ceux de la HS pour les différents exemples des différentes classes. Il est clairement remarquable que les meilleurs résultats sont obtenus par la HS (sauf pour l'exemple 1 où les résultats sont les mêmes et l'exemple 3 ou l'AG donne le meilleur résultat pour la classe de problème 10 jobs×10 machines).

Nous constatons que la HS dépasse l'AG en terme de qualité de solution pour les cas considérés, ceci peut être expliqué par le fait que la HS exploite l'espace de recherche de la même façon que l'AG mais elle offre la possibilité de garder toujours une partie de la population conservée, c'est-à-dire elle décompose l'espace de recherche en deux sous populations et les examine de façons différentes.

3.4. Conclusion

Nous avons consacré ce chapitre à la présentation des deux algorithmes adaptés. Une étude de sensibilité a été faite pour déterminer les meilleurs paramètres des algorithmes. Les résultats obtenus montrent que la HS dépasse l'AG malgré que le principe des deux méthodes est similaire.

CONCLUSION GENERALE

Ce travail été consacré à l'étude, l'adaptation et la simulation de deux métaheuristiques à population de solutions très proches dans les opérateurs utilisés pour la recherche de solutions mais qui diffèrent dans le fait que l'une (qui est la recherche d'harmonies) procède à subdiviser l'espace de recherche en deux sous populations et les explorer de manières différentes.

Certains points communs ont été clairement visibles lors de l'adaptation tel que l'opérateur de « mutation » avec celui de « pitch ajusting ».

Pour la validation de notre adaptation nous avons appliqué les deux algorithmes sur différentes classes de problèmes de différentes tailles composées de plusieurs exemples.

Les résultats de simulation ont montré que la recherche d'harmonies dépasse l'algorithme génétique en termes de qualité des solutions obtenues.

Nous proposons comme perspectives de ce travail d'hybrider la recherche d'harmonies avec une autre technique de recherche locale au niveau des deux sous populations construites dans l'algorithme.

Références bibliographiques

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

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

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

[4] Sassine, C. Intégration des politiques de maintenance dans lessystèmes de production manufacturiers. Thèse de doctorat soutenue àl'INP de Grenoble (France). (1998)

[5] Amodeo, L., (1999). Contribution à la simplification et à la commande des réseaux de Petri stochastiques. Application aux systèmes de production, Thèse de doctorat en Productique soutenue à l'INP,Grenoble (France). (1999).

[6] Caumond, A., Lacomme, P., Moukrim, A., Tchernev, N.,
An MILP for scheduling problems in an FMS with one vehicle.European journal of operational research.(2006).

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

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

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

[10] Esquirol, P., Lopez, P., L'ordonnancement, Série: Production et techniques quantitatives appliquées à la gestion, collection Gestion, Economica.(1999).

[11] Parunak, H.V.D., Manufacturing experience with the contract net. In Proceedings of the Fith Workshop on Distributed Artificial Intelligence. (1985).

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

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

[14] Mesghouni, K, Application des algorithmes évolutionnistes dans les problems d'optimisation en ordonnancement de la production, Thèse de doctorat, Université des sciences et technologies de Lille1. (1999).

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

[16] **DRISS IMEN ,** Analyse D'un Système Job Shop AspecOrdonnancemetn,Thèse de Doctorat , Batna, 2016

[17] Talbi, E., Sélection et Réglage de Paramètres pour l'Optimisation de
Logiciels d'Ordonnancement Industriel, Thèse de doctorat, Institut National Polytechnique de Toulouse. (2004).

[18] Ayari, N., Métaheuristiques parallèles hybrides pour l'optimisation combinatoire : problème de règles de Golomb, Mémoire de Master, Université de Jendouba, Tunisie. (2010).

[19] Boumediene merouane hocine, les problemes d'ordonnancement a machines parallèles, de taches dépendantes. Magister .blida, juillet 2006

[20] Sevaux, M,Métaheuristiques Stratégie pour l'optimisation de la production de biens et de services, Habilitation à diriger des recherches, Laboratoire d'Automatique, de Mécanique d'informatique Industrielles et Humaines du CNRS (UMR CNRS 8530) dans l'équipe systèmes de production, (2004).

[21] L. Ferro, S. Khin, N. Salman .Résolution pratique de problèmes NP complets. 2005.

[22] J. R. Slagle. Artificial intelligence: The heuristic programming approach. McGraw-Hill.pp. 3. New York, 1971

[23] J. R. Slagle. Artificial intelligence: The heuristic programming approach. McGraw-Hill.pp. 3. New York, 1971

[24] J. Pearl. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Publ. Co, London, 1984.

[25] R. L. Solso. Cognitive psychology.Harcourt Brace Jovanovich, Inc., New York.436, 1979.

[26] R. L. Solso. Cognitive psychology.Harcourt Brace Jovanovich, Inc., New York.436, 1979.

[27] S.O. Shim, Y.D. Kim .A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property .Computing. Operational Research. 35 (2008) 863–875.

[28] P.Esquirol ,P.Lopez . L'Ordonnancement .Edition ParieEconomica ISBN 1999 2-7178-3798-1.

[29] Y Wang , L. Kwei . Implementing a general real time scheduling framework in the RED-Linux real-time kernel.In IEEE Real-Time Systems Symposium. (1999)246–255 .

[30] E. Hart , P. Ross .Evolutionary Scheduling: A Review. Spring Genetic Programming and Evolvable Machines, 6 (2005)191–220.

[31] I.H. Osman, G. Laporte .Metaheuristics : A bibliography. Operations Research. 63(1996) 513-623.

[32] Metaheuristics Network Websitehttp://www.metaheuristics.net visité en novembre 2004

[33]I.Driss, K.N.Mouss, A.Laggoun . A New genetic algorithm for flexible job shop scheduling problem .Journal of Mechanical Science and Technology . 29 (3) (2015) 1273-1281 .

[34] Duvivier, D., Etude de l'hybridation des métaheuristiques, application à un problème d'ordonnancement de type jobshop, Thèse de doctorat, Laboratoire d'Informatique du Littoral, Cote-d'Opale, Calais. (2000).

[35] M. Dorigo et L.M. Gambardella. Ant colony system: a cooperative learning approach to the travelling salesman problem. IEEE Transactions on Evolutionary Computation . 1(1997). 53-66.

[36] Hertz A., Les métaheuristiques : quelques conseils pour en faire bon usage, dans : Gestion de production et ressources humaines : méthodes de planification dans les systèmes productifs, Presse internationale Polytechnique, Montréal. p. 205-222.

[37] S.Khalouli . Métaheuristique à base de modéle : application à l'ordonnancement d'atelier flow shop hybride monocritére. Thése de doctorat . Université de Reims 2010

[38] H. Boukef, F. Tangour et M. Benrejeb. Sur la formulation d'un problème d'ordonnancement de type flow-shop d'ateliers de production en industries pharmaceutique . Journées Tunisiennes d'Electrotechnique et d'Automatique, Hammamet, 2006.

[39] S.A. Brah, J.L. Hunsuchker .Branch and bound algorithm for the flow shop with multiple processors .Eur. J. Oper. Res. 51 (1991) 88–99.

[40] M. Clerc ,J.Kennedy . The Particle Swarm. Explosion, stability, and convergence in a multidimensional complex space . IEEE Transactions on Evolutionary Computation, 6 (2002) 58-73.

[41] Goldberg, D.E., Genetic algorithms in search, Optimization andmachine learning, Addison-Wesley publishing company.(1989).

[42] Melle Ferhat Halima, Optimisation de la QoS dans un réseau de radio cognitive en utilisant l'algorithme de la recherche par harmonie, **2015 .**

[43] Teich, T., Algorithms for the Job Shop Problem : a comparison of different methods. In : Proceedings of the European Symposium on Intelligent Techniques, 1999.

[44] Bierwirth, C., A Generalized Permutation Approach to Job-Shop Scheduling with Genetic Algorithms, OR Spektrum, vol. 17(2-3), pp. 87-92.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

Résumé

Dans ce travail nous nous intéressons à l'adaptation de deux métaheuristiques à base de population, l'algorithme génétique et la recherche d'harmonies pour la résolution d'un problème d'ordonnancement Flow Shop.

Le choix de ces deux techniques est dû au fait qu'elles ont une certaine ressemblance dans leurs principes de recherche.

Les résultats de simulation ont montré que la recherche d'harmonie dépasse l'algorithme génétique en termes de qualité des solutions obtenues pour différentes classes de problèmes.

Mots clés

Ordonnancement, Algorithme génétique, Recherche d'harmonies, Flow Shop.

Abstract

In this work we are interested to the adaptation of two population based metaheuristics, the genetic algorithm and the harmony search to solve a Flow shop Scheduling problem.

The choice of the two techniques is due the fact that they have some resemblance in their search technique principles.

Computational results have shown that the harmony search technique outperform the genetic algorithm in terms of the solutions quality obtained for different classes of problems.

Key words

Scheduling, genetic algorithm, Harmony search, Flow Shop.

ملخص

نتطرق في هذا العمل لتطبيق فوقيتي استدلال متعددة الحلول و هما خوارزمية الجينات و تقنية البحث عن الإيقاع لحل إشكالية الترتيبات وحيدة المسار. اختيار هاتين التقنيتين يعود لكونهما متشابهتين في مبادئ تقنية البحث.

نتائج المحاكاة أظهرت أن تقنية البحث عن الإيقاع تخطت خوارزمية الوراثة من حيث نوعية الحلول المحصل عليها لعدة طبقات من الإشكاليات.

الكلمات المفتاحية

الترتيبات، خوارزمية الوراثة ، خوارزمية البحث عن الإيقاع ، أحادي المسار.