MA-004-47-1

F.S.D .... N°D'ordre : .....

Université Saâd DAHLAB de Blida

Faculté des Sciences
Département d'informatique

Mémoire présenté par :

M^elle ADDAD Noura.

et

M^elle SAIDANI Mounira.

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et Informatique.

Filière : Informatique.

Spécialité : Informatique.

Option : ingénierie de logiciel.

Sujet : Approche méta-heuristique basée sur la méthode de recherche d'harmonique pour la résolution des problèmes d'ordonnancement industriels.

Organisme d'accueil : Centre de Développement des Technologies Avancées (CDTA)

Soutenue le : /07/2011 devant le jury composé de:

M.	Président.
Mr.GAHAM Mehdi	Promoteur.
M.	Examinateurs.

دعاء

ربي...اشرح لي صدري و يسرلي أمري و احلل العقدة من لساني يفقه قولي

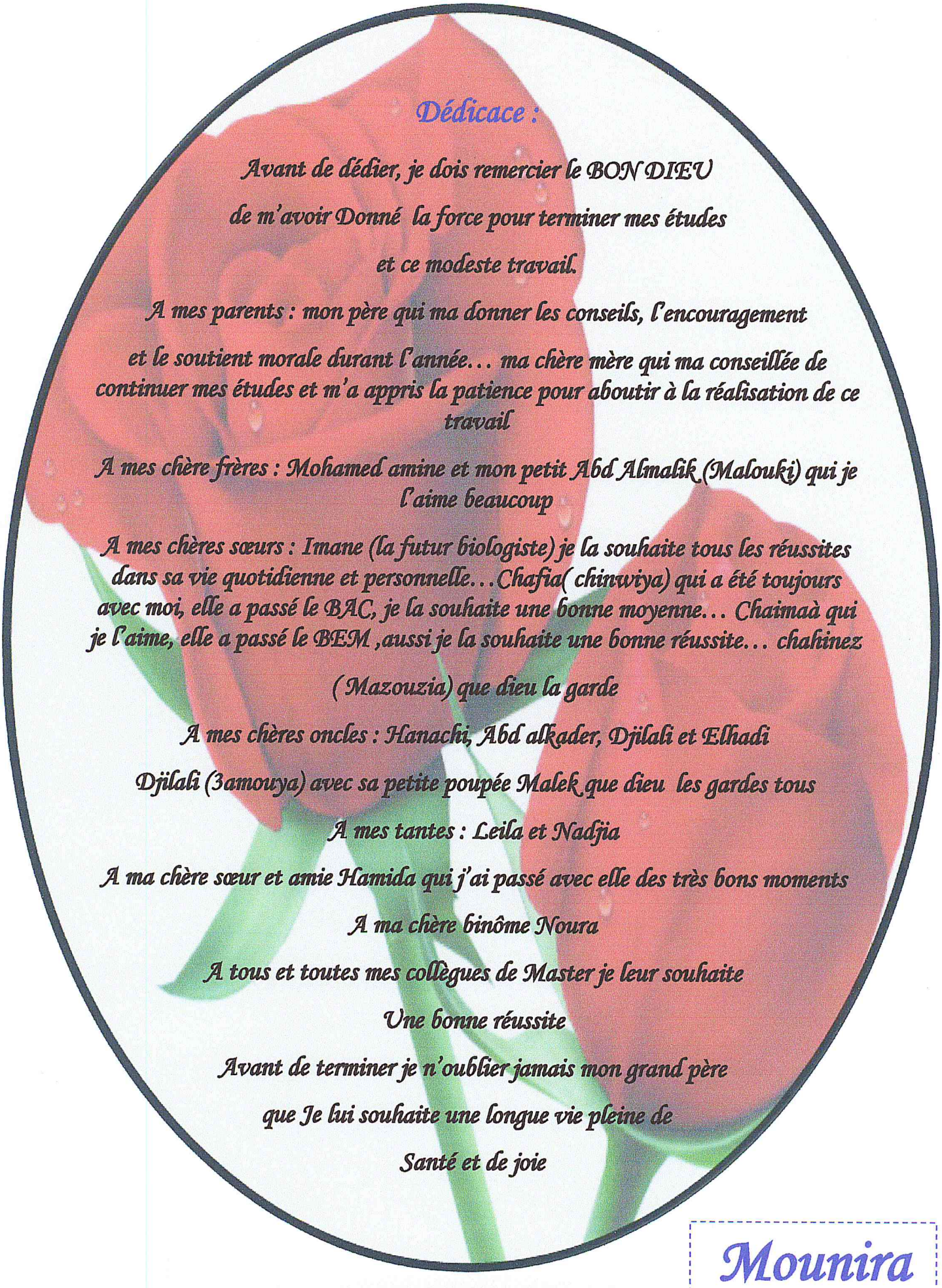
ربي ...لا تدعني أصاب بالغرور إذا نجحت ولا أصاب باليأس إذا فشلت...بل ذكرني دوما بان الفشل هو التجارب التي تسبق النجاح

ربي...علمني أن التسامح هو اكبر مراتب القوة...و أن حب الانتقام هو أول مراتب الضعف

ربي...إذا جردتني من المال...اترك لي نعمة الأمل...وإذا جردتني من الأمل...اترك لي قوة الصبر كي أتغلب على الفشل وإذا جردتني من الصبر...اترك لي نعمة الإيمان

ربي...إذا اسأت إلى الناس...امنحني شجاعة الاعتذار...و إذا أساء الناس إلي أعطني شجاعة العفو

ربي...إذا نسيتك فلا تنساني



Dédicace :

Avant de dédier, je dois remercier le BON DIEU

de m'avoir Donné la force pour terminer mes études

et ce modeste travail.

A mes parents : mon père qui ma donner les conseils, l'encouragement

et le soutient morale durant l'année… ma chère mère qui ma conseillée de continuer mes études et m'a appris la patience pour aboutir à la réalisation de ce travail

A mes chère frères : Mohamed amine et mon petit Abd Almalik (Malouki) qui je l'aime beaucoup

A mes chères sœurs : Imane (la futur biologiste) je la souhaite tous les réussites dans sa vie quotidienne et personnelle…Chafia(chinwiya) qui a été toujours avec moi, elle a passé le BAC, je la souhaite une bonne moyenne… Chaimaà qui je l'aime, elle a passé le BEM ,aussi je la souhaite une bonne réussite… chahinez

(Mazouzia) que dieu la garde

A mes chères oncles : Hanachi, Abd alkader, Djilali et Elhadi

Djilali (3amouya) avec sa petite poupée Malek que dieu les gardes tous

A mes tantes : Leila et Nadjia

A ma chère sœur et amie Hamida qui j'ai passé avec elle des très bons moments

A ma chère binôme Noura

A tous et toutes mes collègues de Master je leur souhaite

Une bonne réussite

Avant de terminer je n'oublier jamais mon grand père

que Je lui souhaite une longue vie pleine de

Santé et de joie

Mounira

Dédicace :

Je dédie ce modeste travail à :

* *Mes parents qui m'encouragent toujours.*

* *Mes frères et sœurs sincèrement à ma petite sœur Hanan Ahlam.*

* *Tous mes amies :A.Habiba R.Siham, S.Fouzia, M.Sanaa, K.Hala. Et sans oublié ma binôme Saidani Mounira .*

* *Tous mes enseignants surtout à Melle Farhi . F*

Ainsi qu'à tout les personnes qui m'ont encouragé.

NOURA

Remerciement

Nous remercie le BON DIEU qui nous a offert la vie, et Seul capable de nous offrir la joie, le bonheur, la prospérité et la Santé pour arriver au maximum de notre travail

Nos premiers remerciements iront à notre promoteur Monsieur « Gaham Mehdi » qui a toujours été présent avec nous dans notre travail et leur suivi, Et qui nous a fait bénéficier de son expérience et de sa connaissance.

Nous adressons également un remerciement a :

Toutes l'équipes robotique et architecture système au CDTA

Nous tenons à remercier ici « Faiza » et tous les personnes qui, par leurs conseils et leurs encouragements ont contribué à l'aboutissement de ce travail.

Enfin nous tenons à remercier les membres de jury.

Listes des figures

Liste des tableaux

Liste des tableaux

Sommaire.

Sommaire.

Sommaire.

Introduction Générale

Introduction Générale

Parmi les problèmes rencontrés par les chercheurs et les ingénieurs, les problèmes d'optimisation occupent à notre époque une place de choix. Les problèmes d'ordonnancement dans le secteur industriel, sont parmi les problèmes d'optimisation les plus étudiés. Améliorer le rendement des ressources et minimiser les coûts de production sont des impératifs. Chercher le meilleur moyen de maximiser son profit est aujourd'hui l'un des objectifs principaux de toute entreprise.

Nous pouvons distinguer parmi les problèmes d'ordonnancement d'atelier les plus rencontrés ceux à machine unique, à machines parallèles, le flow-shop, le job-shop, l'open-shop, etc. Dans le cadre notre travail nous nous focalisons sur l'ordonnancement d'atelier de type job-shop flexible dans les cas monocritère et multicritère. En effet, ce type de structure d'atelier est intéressant pour la modélisation de plusieurs problèmes issus du monde industriel.

La résolution optimale des problèmes d'ordonnancement de type Job Shop Flexible, s'avère dans certains cas impossible à cause de leur caractère fortement combinatoire. Ainsi, la taille et la complexité du problème entrent en compte pour le choix de la méthode d'optimisation. Si le problème est de petite taille et de complexité réduite, la mise en oeuvre d'une méthode exacte peut suffire pour déterminer à une solution optimale. Dans le cas de problèmes de tailles importantes, les méthodes approchées constituent le moyen le plus efficace de se rapprocher le plus possible de la solution optimale.

Le paradigme des Méta-heuristiques que nous adoptons ici, s'impose comme une approche très prometteuse pour la résolution des problèmes d'ordonnancement. En effet, en plus de leur adaptabilité aux différents problèmes combinatoires, les méta-heuristiques ont l'avantage de ne parcourir qu'une faible fraction de l'espace de solution pour parvenir à une solution acceptable. Ce qui réduit nettement les temps de calcul.

Plus particulièrement nous proposons dans ce mémoire l'adaptation d'une nouvelle méta-heuristique dite de recherche d'harmonique pour la résolution du job shop flexible. En effet, cette méthode qui a été récemment développée par *Zong Woo Geem* pour résoudre les problèmes d'optimisation trouve ces dernières années un champ applicatif très large et s'avère tres performante pour la résolution de ces derniers.

Le contenu de notre étude a été organisé selon 2 parties majeures :

Partie I : étude prémilitaire

Cette partie contienne deux chapitres.

Chapitre I : L'ordonnancement industriel.

Chapitre II : Les algorithmes de résolution.

Partie II : Implémentation

Chapitre III: Application De l'algorithme harmony search au job shop flexible.

Chapitre VI : Mise en œuvre et résultats.

Partie I

Etude préliminaire

Chapitre 1:

L'ordonnancement Industriel

I.1. Introduction :

La théorie de l'ordonnancement est une branche de la recherche opérationnelle. Elle occupe un rôle important dans nombreux secteurs de l'économie, notamment en gestion de Production des entreprises de biens ou de services. L'enjeu économique considérable de ces Problèmes ainsi que leurs champs d'applications qui n'ont pas cessé d'augmenter et d'évoluer, ont suscité depuis des décennies une recherche intensive. La fonction ordonnancement vise à définir les dates d'exécution d'un ensemble de travaux en tenant Compte de la disponibilité des ressources, de manière à optimiser un ou plusieurs critère(s) Donné(s), tout en respectant les contraintes de fabrication et d'organisation.

Nous proposons dans ce chapitre une brève introduction, dans laquelle nous rappelons Simplement les notions de base utiles à la compréhension de ce mémoire, en particulier à la Présentation des problèmes d'ordonnancement d'atelier, et à la fin les différentes approches de résolution.

I.2. Le problème d'ordonnancement : [1]

Les problèmes d'ordonnancement d'atelier sont des problèmes avec contraintes de ressources. Les ressources sont des machines ne pouvant réaliser qu'une opération à la fois et les machine sont renouvelables. D'autre part chaque travail (tâche) est composer de plusieurs entités appelée opération. Un travail ne pouvant se trouver simultanément en deux lieux distincts, un même travail ne peut être exécuté qu'à raison d'une opération la fois, sur une seule des machines.

Établir un ordonnancement revient donc à coordonner l'exécution de toutes les tâches, en utilisant au mieux les ressources disponibles.

I.3. Les concepts de base de l'ordonnancement : [2,3]

Les différentes données d'un problème d'ordonnancement sont les tâches, les ressources, les contraintes et les critères. Ainsi, étant donnés un ensemble de tâches et un ensemble de ressources, il s'agit de programmer les tâches et affecter les ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), en respectant un ensemble de contraintes.

I.3.1. Les tâches :

Une tâche est une entité élémentaire localisée dans le temps, par une date de début et/ou une date de fin, et dont la réalisation nécessite une durée préalablement définie.

Elle est constituée d'un ensemble d'opérations qui requiert pour son exécution certaines ressources, et qu'il est nécessaire de programmer de façon à optimiser un certain objectif.
On distingue deux types de tâches :

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

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

I.3.2. Les ressource :

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

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

- ***Les ressources consommables :*** qui après avoir été allouées à une tâche ne sont plus disponibles (argent, matières premières, …etc).

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Par ailleurs dans le cas des ressources renouvelables, on distingue principalement :

- **Les ressources disjonctives :** qui ne peuvent exécuter qu'une tâche à la fois.
- **Les ressources cumulatives :** qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité.

I.3.3. Les contraintes :

Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distingués :

- ***Les contraintes de ressources*** **:** plusieurs types de contraintes peuvent être induites parla nature des ressources. A titre d'exemple la capacité limitée d'une ressource implique un certain nombre à ne pas dépasser de tâches à exécuter sur cette ressource.

Les contraintes relatives aux ressources peuvent être disjonctives, induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource ou cumulatives impliquant la limitation du nombre de tâches à réaliser en parallèle.

- ***Les contraintes temporelles*** **:** elles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement.

Ces contraintes peuvent être :

- **Des contraintes de dates butoirs :** certaines tâches doivent être achevées avant une date préalablement fixée.
- **Des contraintes de précédence :** une tâche *i* précède la tâche *j* c.à.d. la tache j ne peut être exécuté soft si la tache i terminer sa exécution,
- **Des contraintes de dates au plus tôt** liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

I.3.4. Les critères d'optimisation :[3]

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi. Les critères dépendant d'une application donnée sont très nombreux; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la plus satisfaisante dépend du ou des critères préalablement définis pouvant être classés suivant deux types les critères réguliers et les critères irréguliers.

Les différents critères ne sont pas indépendants; certains même sont équivalents.
Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement.

- ***Les critères réguliers :*** constituent des fonctions décroissantes des dates d'achèvement des opérations. Quelques exemples sont cités ci-dessous:
  - la minimisation des dates d'achèvement des actions.
  - la minimisation du maximum des dates d'achèvement des actions.
  - la minimisation de la moyenne des dates d'achèvement des actions.
  - la minimisation des retards sur les dates d'achèvement des actions.
  - la minimisation du maximum des retards sur les dates d'achèvement des actions.
- ***Les critères irréguliers :*** sont des critères non réguliers c'est-à-dire qui ne sont pas des fonctions monotones des dates de fin d'exécution des opérations, tels que:
  - la minimisation des encours.
  - la minimisation du coût de stockage des matières premières.
  - l'équilibrage des charges des machines.
  - l'optimisation des changements d'outils.

La satisfaction de tous les critères à la fois est souvent délicate, car elle conduit souvent à des situations contradictoires et à la recherche de solutions à des problèmes complexes d'optimisation.

I.4. Les ateliers : [3]

Une classification des problèmes d'ordonnancement dans un atelier peut s'opérer selon le Nombre de machines et leur ordre d'utilisation pour fabriquer un produit, qui dépend de la nature de l'atelier considéré. Un atelier est caractérisé par le nombre de machines qu'il contient et par son type.

On distingue les trois types d'ateliers suivants : flow-shop, job-shop et open-shop, avec des extensions possibles pour chacun d'eux.

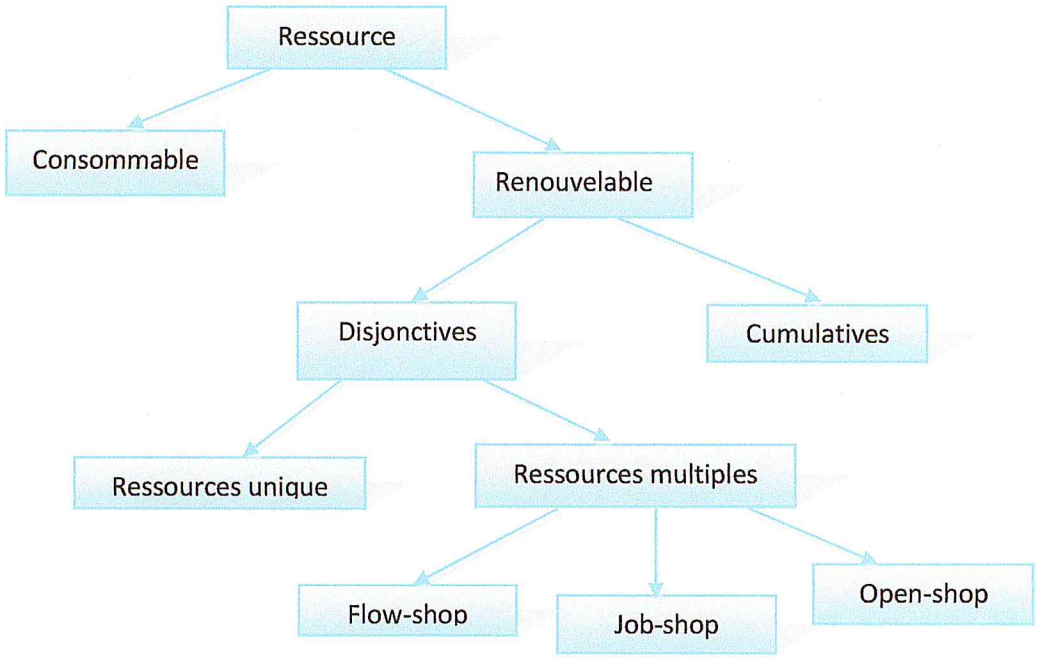


Figure I. 1 : Classification des types d'ateliers.

I.4.1. Les ateliers de type flow-shop :

Appelés également ateliers à cheminement unique, ce sont des ateliers où une ligne de fabrication est constituée de plusieurs machines en série; toutes les opérations de toutes les tâches passent par les machines dans le même ordre. Dans les ateliers de type *flow-shop hybride*, une machine peut exister en plusieurs exemplaires identiques fonctionnant en parallèle.

I.4.2. Les ateliers de type job-shop :

Appelés également ateliers à cheminement multiple, ce sont des ateliers où les opérations sont réalisées selon un ordre bien déterminé, variant selon la tâche à exécuter; le *job-shop flexible* est une extension du modèle job-shop classique; sa particularité réside dans le fait que Plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations.

I.4.3. Les ateliers de type open-shop :

Ce type d'atelier est moins contraint que celui de type flow-shop ou de type job-shop. Ainsi, l'ordre des opérations n'est pas fixé a priori; le problème d'ordonnancement consiste d'une part, à déterminer le cheminement de chaque produit et d'autre part, à ordonnancer les Produits en tenant compte des gammes trouvées, ces deux problèmes pouvant être résolus simultanément.

I.5. Optimisation mono-objectif / Optimisation multi-objectifs : [3]

La résolution d'un problème d'optimisation mono-objectif consiste à trouver le minimum ou le maximum d'une fonction f donnée. Ce besoin d'optimisation vient de la nécessité de fournir à l'utilisateur un système qui réponde au mieux à ses attentes. Mais parfois, il s'avère nécessaire d'optimiser plusieurs fonctions, d'où la nécessité de recourir à une optimisation Multi-objectifs.

I.5.1. Optimisation mono-objectif :

Dans le cas mono-objectif, la définition de f ne pose généralement pas de problème, lorsqu'une seule valeur est associée à une seule fonction objectif.

D'un point de vue mathématique, un problème d'optimisation mono-objectif se présente de la façon suivante :

$$\begin{cases} \text{minimiser } f(x) & x \in D,\ f(x) \in R \\ \text{avec } g_i(x) \leq 0 & g_i(x) \in R \\ \text{et } h_j(x) = 0 & h_j(x) \in R \end{cases}$$

où :

- D l'ensemble de solution du problème à optimiser.
- f la fonction objectif.
- g_i la i^{eme} contrainte inégalité $\forall i \in \{1,..., m\}$.
- h_j la j^{eme} contrainte égalité $\forall j \in \{1,..., p\}$.

Les objectifs les plus régulièrement considérés sont la minimisation de la durée totale de l'ordonnancement , le respect des dates au plus tard, la minimisation des retards, la minimisation d'un coût,...etc.

I.5.2. Optimisation multi-objectifs :

Pour mieux résoudre ces problèmes d'optimisation multi-objectifs, il est nécessaire de simplifier les différentes fonctions objectives pour faciliter leur traitement.

Parmi les différentes méthodes utilisées, la méthode de pondération des fonctions objectif qui se présente comme suit : A chacune des fonctions objectif est appliqué un coefficient de pondération, et la somme pondérée des fonctions objectifs est effectuée Une nouvelle fonction objectif est ainsi obtenue.

D'un point de vue mathématique, la formulation du problème avec la méthode de pondération des fonctions objectif se présente de la façon suivante :

$$\begin{cases} \text{Minimiser } f(x) = \sum_{i=1}^{k} \alpha_i f_i(x) \quad x \in D, k \geq 2 \\ \alpha_i \geq 0 \quad \forall i \in \{1,\ldots, k\} \text{ et } \sum_{i=1}^{k} \alpha_i = 1 \\ \text{avec } g_l(x) \leq 0 \qquad g_l(x) \in R \\ \text{et } h_j(x) = 0 \qquad h_j(x) \in R \end{cases}$$

où :

- D l'ensemble de solution du problème à optimiser.
- f_i la i^{eme} pondération de la fonction objectif et α_i le i^{eme} coefficient de pondération.
- g_l la l^{eme} contrainte inégalité $\forall l \in \{1,\ldots, m\}$.
- h_j la j^{eme} contrainte égalité $\forall j \in \{1,\ldots, p\}$.

I.6. Le job shop flexible:[4]

Un job shop flexible est une extension du problème à cheminements multiples classique. En effet, la résolution de ce problème présente une difficulté supplémentaire dans la mesure où chaque opération nécessite une machine pour être réalisée et cette machine doit être choisie dans un ensemble défini a priori.

Les problèmes d'ordonnancement d'ateliers de type job-shop flexible sont connus dans la littérature comme étant les problèmes les plus difficiles à résoudre. La difficulté réside dans le choix de la meilleure méta-heuristique pour leur résolution ainsi que pour la détermination des meilleurs ordonnancements en des temps raisonnables, les plus proches possibles de la solution optimale.

Dans un problème d'ordonnancement de type job-shop flexible, une opération donnée pouvant être réalisée par une ou plusieurs ressources, possède une durée de traitement dépendant de la ressource utilisée. Les contraintes relatives à ce type de problème sont de type précédence, temporel ou disjonctif.

I.6.1. Formulation des problèmes FJSP :

Les problèmes d'ateliers job-shop flexibles peuvent être formulés comme suit :

- soit un ensemble de n produits indépendants à réaliser sur m machine M_k, $k = 1, 2..m$.
- chaque produit P_i est constitué d'une séquence de l_i opérations O_{ij}, $j = 1, 2, \ldots, l_i$, à exécuter selon un ordre bien défini.
- l'exécution de chaque opération O_{ij} d'un job P_i nécessite une ressource sélectionnée à partir d'un ensemble de machines disponibles.
- chaque machine ne peut réaliser qu'une seule opération à la fois.
- L'assignation d'une opération O_{ij} à une machine M_k entraîne l'occupation de cette machine durant tout le temps d'exécution de l'opération, noté D_{ijk}.
- la préemption n'est pas autorisée.

Le FJSP présente deux difficultés principaux :

- la première est relative à l'assignation de chaque opération O_{ij} à une machine M_k.
- la seconde correspond au calcul des temps de début TD_{ij} et des temps de fin TF_{ij} de chaque opération O_{ij}.

I.7. Les approches de résolution des problèmes d'ordonnancement :[1]

Bien que les problèmes d'ordonnancement soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données.

Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielle (IA).

Ces méthodes peuvent être classées sommairement en deux grandes catégories :

- les méthodes exactes.
- les méthodes approchées.

I.7.1. Les méthodes exactes :[3]

les approches exactes ont pour but de rechercher un ordonnancement optimal. Elles se caractérisent par l'augmentation du temps de calcul de façon exponentielle avec la taille du problème, ce qui explique leurs utilisations le plus souvent pour la résolution des problèmes de petite taille.

I.7.2. Les méthodes approchées : [3]

Les méthodes approchées, dites aussi d'approximation, constituent une alternative intéressante pour traiter les problèmes d'ordonnancement de grande taille. Elles sont définies comme étant des procédures exploitant au mieux la structure du problème considéré afin de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible. Ainsi, les performances de ce type d'approches dépendent de la qualité de la solution obtenue et du temps de calcul nécessaire pour sa réalisation. Plusieurs méthodes approchées ont été développées pour la résolution des problèmes d'ordonnancement.

Lorsque ces méthodes sont conçues de manière simple, rapide et ciblée sur un problème particulier, on les appelle ***heuristique***. Leur principe de base repose généralement sur l'utilisation de règles empiriques simples qui exploitent les propriétés structurelles d'une solution et tente de retrouver une solution admissible par des critères de décision déduits de la connaissance du problème. Cependant, lorsque celles-ci sont générales, adaptables et applicables à plusieurs catégories de problèmes d'optimisation combinatoire, elles portent le nom de ***méta heuristique***.

Les méta heuristiques sont généralement des algorithmes stochastiques, qui progressent vers un optimum par échantillonnage d'une fonction objectif. Ainsi, elles sont le plus souvent, guidées par l'exploration aléatoire de l'espace de recherche.

I.7.2.1. Quelques propriétés associées aux méthodes approchées :

La plupart des méthodes approchées ne sont pas déterministes, c'est-à-dire que deux exécutions successives utilisant les mêmes données peuvent retourner deux résultats différents. Ils reposent généralement sur un ensemble de concepts à savoir :

- ***la mémoire*** : qui est un support d'apprentissage permettant de sauvegarder les informations recueillies par l'algorithme, à mesure que la recherche avance afin d'en faire usage ultérieurement.

- ***l'intensification*** : qui permet de rechercher des solutions a priori de plus grande qualité en exploitant les informations rassemblées par le système durant la recherche.
- ***la diversification*** : qui permet d'explorer de nouvelles régions de l'espace de recherche non encore visitées. Les associations multiples de ces concepts peuvent conduire à une grande variété d'autres méthodes.

I.7.3. Comparaison entre méthodes exactes et méthodes approchées : [1]

On représente ci-dessous quelques différences entre les méthodes exactes et les méthodes approchées.

Méthodes approchées	Méthodes exactes
Efficacité pour les problèmes de grande taille	Efficacité pour des cas particuliers des problèmes de taille raisonnable
Espace mémoire raisonnable	Espace mémoire considérable
Durée de résolution très petite par rapport aux méthodes exactes	Durée de temps de résolution est généralement considérable pour les problèmes de grande taille
Aucune garantie d'optimalité	Garantie d'optimalité
Pratiquement simple à implémenter et à comprendre	Pratiquement difficile à implémenter et à comprendre.

Tableau I.1 : Comparaison entre les méthodes exacte et méthodes approchées.

I.8. Représentation des problèmes d'ordonnancement : [3]

Il existe trois sortes de représentations possibles d'un problème d'ordonnancement:

- Le diagramme de Gantt.
- Le graphe Potentiel-Tâches.

I.8.1. Le diagramme de Gantt :

Le diagramme de Gantt est un outil permettant de modéliser la planification des tâches nécessaires à la réalisation d'un projet. Il s'agit d'un outil élaboré en 1917 par *Henry, Gantt*.

Il permet de représenter graphiquement l'avancement du projet et constitue également un bon moyen de communication entre les différents acteurs d'un projet.

Le diagramme de Gantt présente en ordonnée la liste des tâches, notées T_i à exécuter par les machines notées M_j et en abscisse l'échelle du temps, comme le montre la figure I.2 dans le cas où i = 1,2,…,5 et j = 1,2.

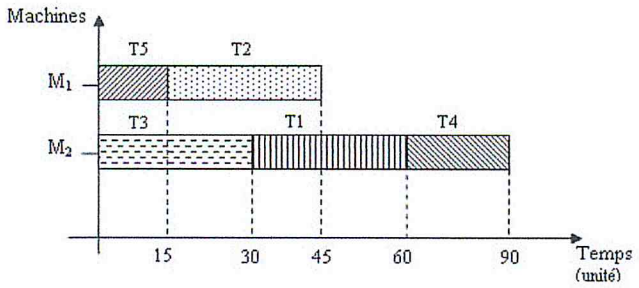


Figure I.2 : Exemple sur le Diagramme de Gantt.

I.8.2. Graphe Potentiel-Tâches :

Cette outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à évènements discrets. Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les contraintes par des arcs, comme le montre la figure I.3. Ainsi, les arcs peuvent être de deux types :

- ***les arcs conjonctifs :*** illustrant les contraintes de précédence et indiquant les durées des tâches.
- ***les arcs disjonctifs :*** indiquant les contraintes de ressources. Pour qu'une tâche puisse commencer, il est nécessaire que toutes les tâches qui la relient à la tâche du début S du projet soient réalisées.

On définit donc :

- **la date au plus tôt de la tâche :** qui correspond à la date de début au plus tôt de l'exécution de la tâche.

- **la date au plus tard de la tâche :** qui correspond à la date de début au plus tard de l'exécution de la tâche.

- **la durée du projet :** qui correspond au plus long chemin entre S (tâche de début du projet) et S' (tâche de fin du projet).

Exemple de graphe potentiel-tâches :

Tâches	Durées	Contraintes
a	6 mois	
b	3 mois	
c	6 mois	
d	2 mois	b achevée
e	4 mois	b achevée
f	3 mois	d et a achevées

Tableau I. 2 : Données utilisées pour la réalisation d'un graphe potentiel-tâches.

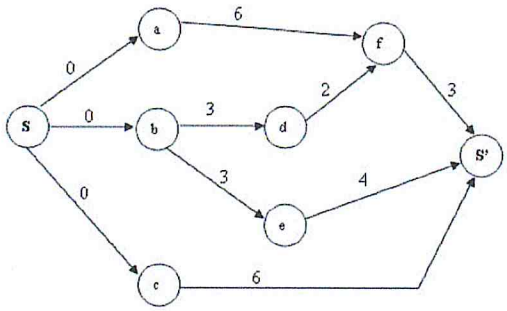


Figure I. 3 - Graphe Potentiel-Tâches d'un ordonnancement .

On remarque que la tâche **f** ne peut s'exécuter que si **a** et **d** ont été réalisées. Donc, pour exécuter **a** il faut 6 mois et pour exécuter **d** il faut 2+3 mois. La tâche **f** ne pourra commencer au plus tôt que 6 mois après le début du projet: c'est donc le plus long chemin entre **a** et **f**.

I.9. Conclusion :

Dans ce chapitre introductif, nous avons rappelé les principales définitions et notations utilisées dans la résolution des problèmes d'ordonnancement en précisant les différents éléments qui les déterminent. Ces derniers étant basés sur les caractéristiques des problèmes, comme l'organisation de l'atelier. Parmi ces ateliers, nous nous intéressons particulièrement à l'étude du job-shop flexible.

Dans le prochain chapitre nous nous intéresserons aux approches méta-heuristiques d'optimisation, particulièrement celles généralement utilisées pour la résolution des problèmes d'ordonnancement.

Chapitre 2:

Les approches de résolution «Les méta-heuristiques »

II.1. Introduction :

À l'opposé des algorithmes d'optimisation exacte il existe d'autres méthodes, les Méta-heuristiques, qui sans garantir l'optimum absolu, peuvent fournir d'excellentes solutions pour la résolution des problèmes d'optimisation complexes. Ces méthodes possèdent l'immense avantage d'être beaucoup moins contraignantes de sorte qu'elles sont pratiquement toujours applicables. Ainsi, dans de nombreux cas, les méta-heuristiques deviennent la seule option performante envisageable.

Ce sont en quelque sorte des méthodes de dernier recours, contrairement aux méthodes mathématiques nécessitant de modéliser le problème à optimiser.

Dans ce chapitre nous présenterons les méthodes méta-heuristiques les plus connues ainsi que celle qui nous concerne qui est la méthode de recherche d'harmonique.

II.2. Les méta-heuristiques :[5]

Les méta-heuristiques sont des méthodes de recherche générales, dédiées aux problèmes d'optimisation difficile. Elles sont, en général, présentées sous forme de concepts. Les principales méta-heuristiques sont celles basées sur la recherche locale, telles que le recuit simulé et la recherche Tabou, et celles basées sur les algorithmes évolutionnistes telles que les algorithmes génétiques ainsi que les algorithmes basés sur la recherche globale tels que les algorithmes de colonies de fourmis et les algorithmes reposant sur la méthode d'optimisation par essaim particulaire.

D'autre part, on peut partager les méta-heuristiques en deux grandes classes :

- ***Les méta- heuristique à solution unique*** : fondés sur la recherche locale de solution, ces algorithmes itératifs procèdent par la génération d'une solution initiale de manière aléatoire. Ils essayent ensuite l'améliorer l'avantage en se basant seulement sur les meilleures solutions se trouvant autour de son voisinage.
- ***celles à solution multiple ou population de solution*** : améliorent au fur et à mesure des itérations, une population des solutions. Ces algorithmes appelés aussi méthodes évolutionnaires, présentent l'avantage d'être dotés d'une certaine intelligence.

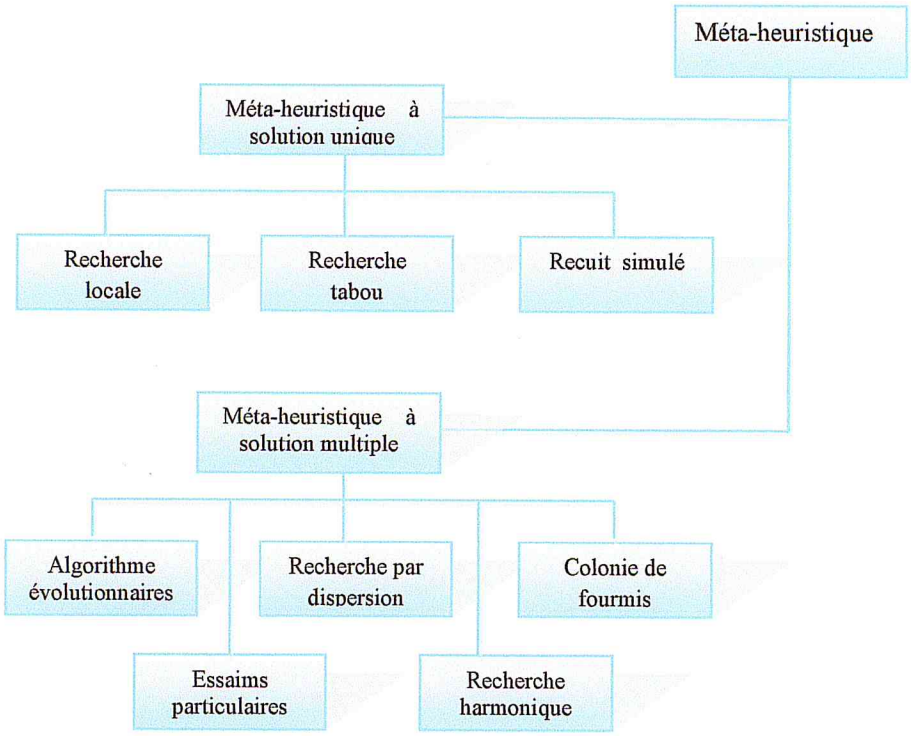


Figure II.1 : Classification des méta-heuristiques.

II.2.1. Méta-heuristique évolutionnaire/génétiques « AG »: [1]

Ici, le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique. Nous parlerons donc d'*individus*, pour parler de solutions. L'ensemble des individus formera une *population*, que nous ferons évoluer pendant une certaine succession d'itérations appelées *générations*, jusqu'à ce qu'un critère d'arrêt soit vérifié. Pour passer d'une génération à une autre, nous soumettrons la population à des *opérateurs de sélection*, de *croisement* et de *mutation* qui permettront de transformer la population, de façon à favoriser l'émergence de meilleurs individus.

- ***La sélection*:** permet de favoriser les individus qui ont un meilleur fitness. Pour nous le fitness sera le plus sauvent la valeur de la fonction objectif de la solution associée à l'individu.
- ***Le croissement:*** combine deux solutions parents pour former un ou deux enfants en essayant de conserver les bonnes caractéristiques des solutions parents.
- ***La mutation:*** permet d'ajouter de la diversité à la population en mutant certaines caractéristiques (gènes) d'une solution.

La représentation des solutions (le codage) est un point critique de la réussite d'un algorithme génétique. Il faut bien sûr qu'il s'adapte le mieux possible au problème et à l'évaluation d'une solution.

II.2.1.1. Procédure de l'Algorithme génétique :

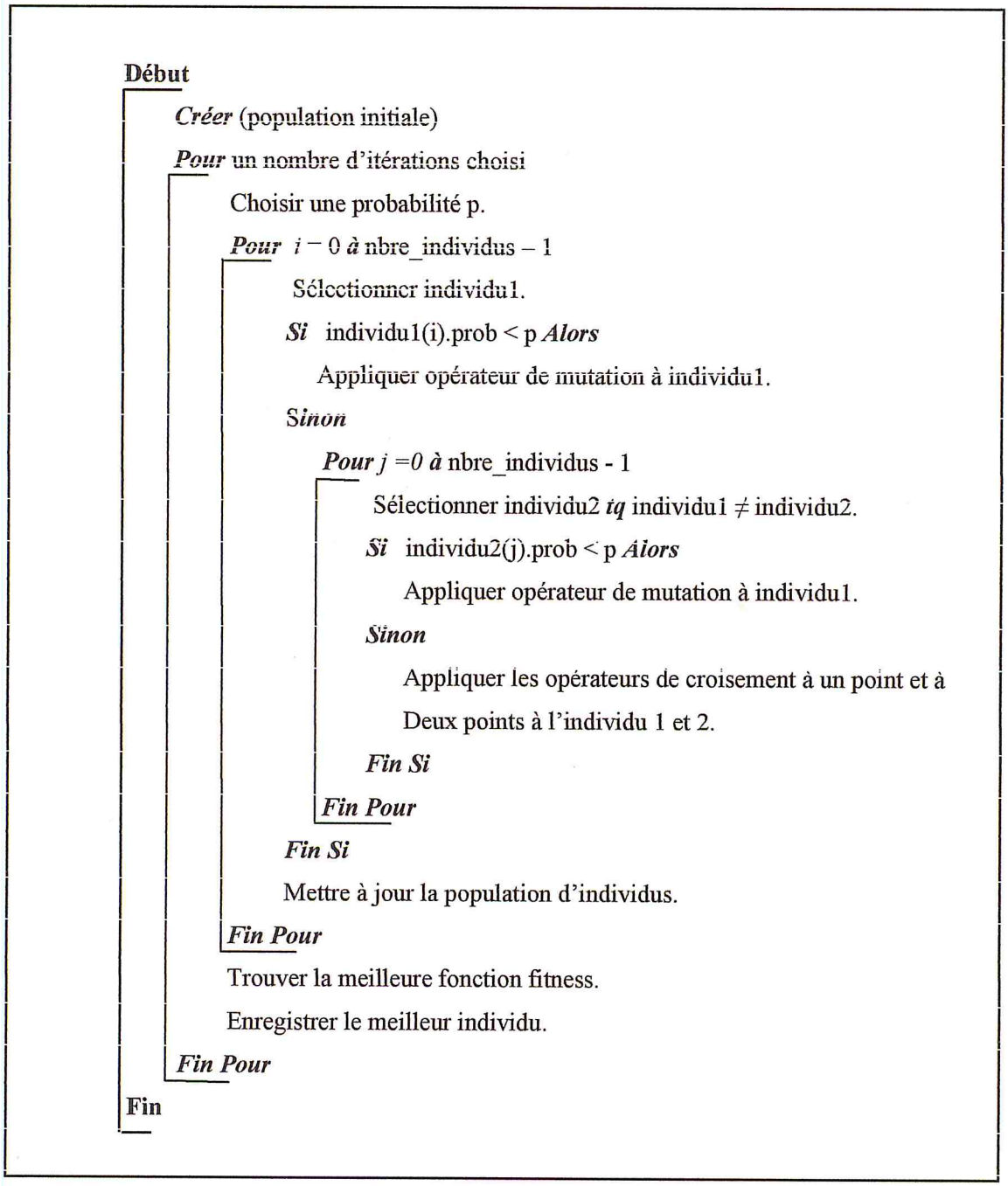


Figure II.2 : Procédure de l'Algorithme génétique.

La démarche de cet algorithme est proposée dans la figure II.2 ci-dessous.

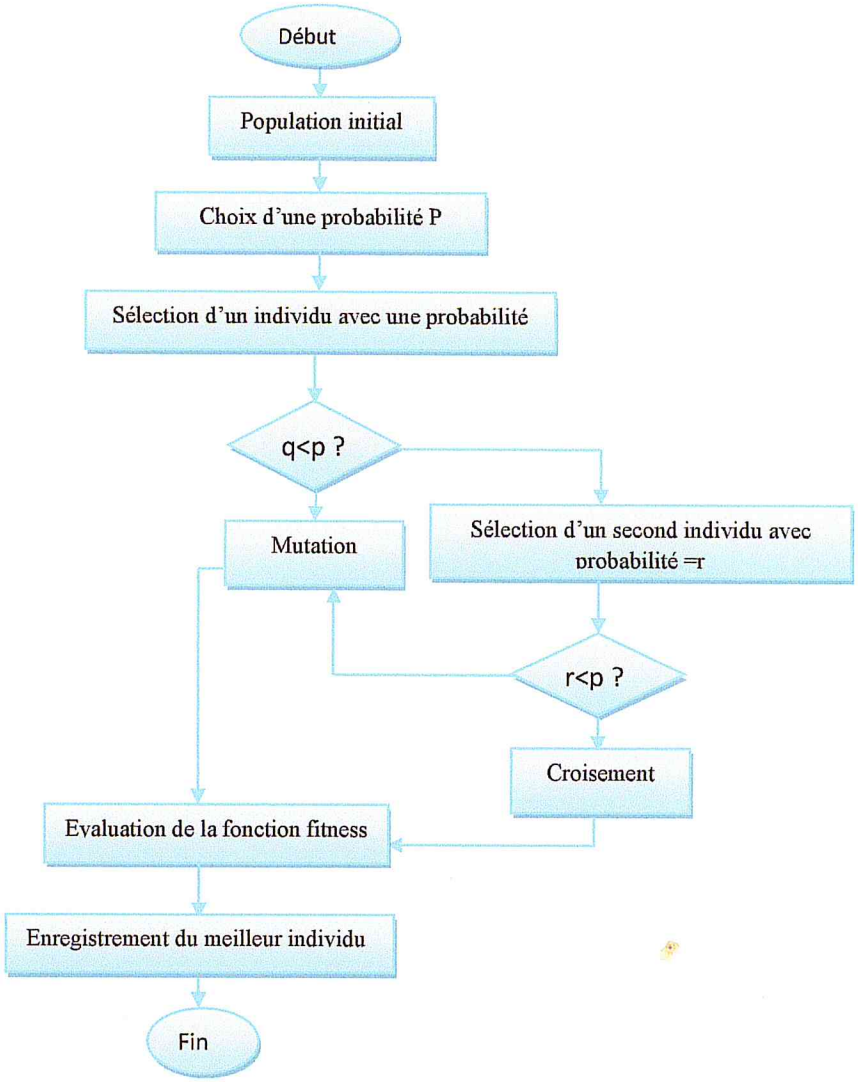


Figure II.3 : Organigramme relatif au fonctionnement général de l'algorithme génétique.

II.2.1.2. Avantages des algorithmes génétiques :

- Le grand avantage des algorithmes génétiques est qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes, et trop éloignés des problèmes combinatoires classiques pour qu'on puisse tirer profit de certaines propriétés connues. Ils doivent simplement déterminer entre deux solutions quelle est la meilleure, afin d'opérer leurs sélections.
- On les emploie dans les domaines où un grand nombre de paramètres entrent en jeu, et où l'on a besoin d'obtenir de bonnes solutions en quelques itérations seulement, par exemple dans les systèmes de régulation de transport en temps réel par exemple.

II.2.1.3. Inconvénients des algorithmes génétiques :

- Les algorithmes génétiques sont coûteux en temps de calcul, puisqu'ils manipulent plusieurs solutions simultanément. C'est l'évaluation qui est le plus pénalisant, et on optimise généralement l'algorithme de façon à éviter d'évaluer trop souvent cette fonction.
- l'ajustement d'un algorithme génétique est délicat. L'un des problèmes les plus caractéristiques est celui de la *dérive génétique*, qui fait qu'un bon individu se met, en l'espace de quelques générations, à envahir toute la population. On parle dans ce cas de convergence prématurée, qui revient à lancer à une recherche locale autour d'un minimum qui n'est pas forcément l'optimum attendu.
- Un autre problème surgit lorsque les différents individus se mettent à avoir des performances similaires : les bons éléments ne sont alors plus sélectionnés, et l'algorithme ne progresse plus.

II.2.2. Méta-heuristique par recherche tabou : [7]

La méthode tabou est une procédure itérative qui partant d'une solution initiale tente de converger vers la solution optimale en exécutant à chaque pas un mouvement dans l'espace de recherche. Chaque pas consiste d'abord à engendrer un ensemble de solutions voisines de la solution courante pour ensuite en choisir la meilleure même si ce choix entraîne une augmentation de la fonction objectif à minimiser. En acceptant de détériorer la valeur de la solution courante le minimum local peut être évité mais en contre partie des parcours répétitifs sont déplorés. Aussi, pour palier à l'inconvénient majeur des méthodes de recherche locale, la recherche tabou a pour but d'améliorer à chaque étape la valeur de la fonction objectif en utilisant une mémoire afin de conserver les informations sur les solutions déjà visitées.

Cette mémoire constitue la *liste Tabou* qui va servir à interdire l'accès aux dernières solutions visitées. Lorsqu'un optimum local est atteint il y a interdiction de revenir sur le même chemin. Un *critère d'aspiration* est également utilisé pour lever l'interdiction d'utilisation d'un mouvement si ce dernier conduit à une meilleure solution. Plusieurs stratégies ont été proposées récemment afin d'améliorer l'efficacité de la méthode tabou « L'intensification » et « la diversification » :

- **L'intensification :** consiste à explorer en détails une région de l'espace de recherche jugée prometteuse. Sa mise en œuvre consiste le plus souvent en un élargissement temporaire du voisinage de la solution courante dans le but de visiter un ensemble de solutions partageant certaines propriétés.

- **La diversification :** a pour objectif de diriger la procédure de recherche vers des régions inexplorées de l'espace de recherche. La stratégie de diversification la plus simple consiste à redémarrer périodiquement le processus de recherche à partir d'une solution générée aléatoirement ou choisie judicieusement dans une région non encore visitée de l'ensemble des solutions admissibles.

II.2.2.2. Procédure de recherche Tabou :

Les étapes d'évolution de l'algorithme sont présentées dans la figure II.4.

- s_0 : la solution initiale.
- s* : la meilleure solution actuelle.
- f(x) : la fonction à minimiser.
- T : la liste tabou.
- f(s*) : la valeur de la fonction à minimiser pour obtenir la meilleure solution.

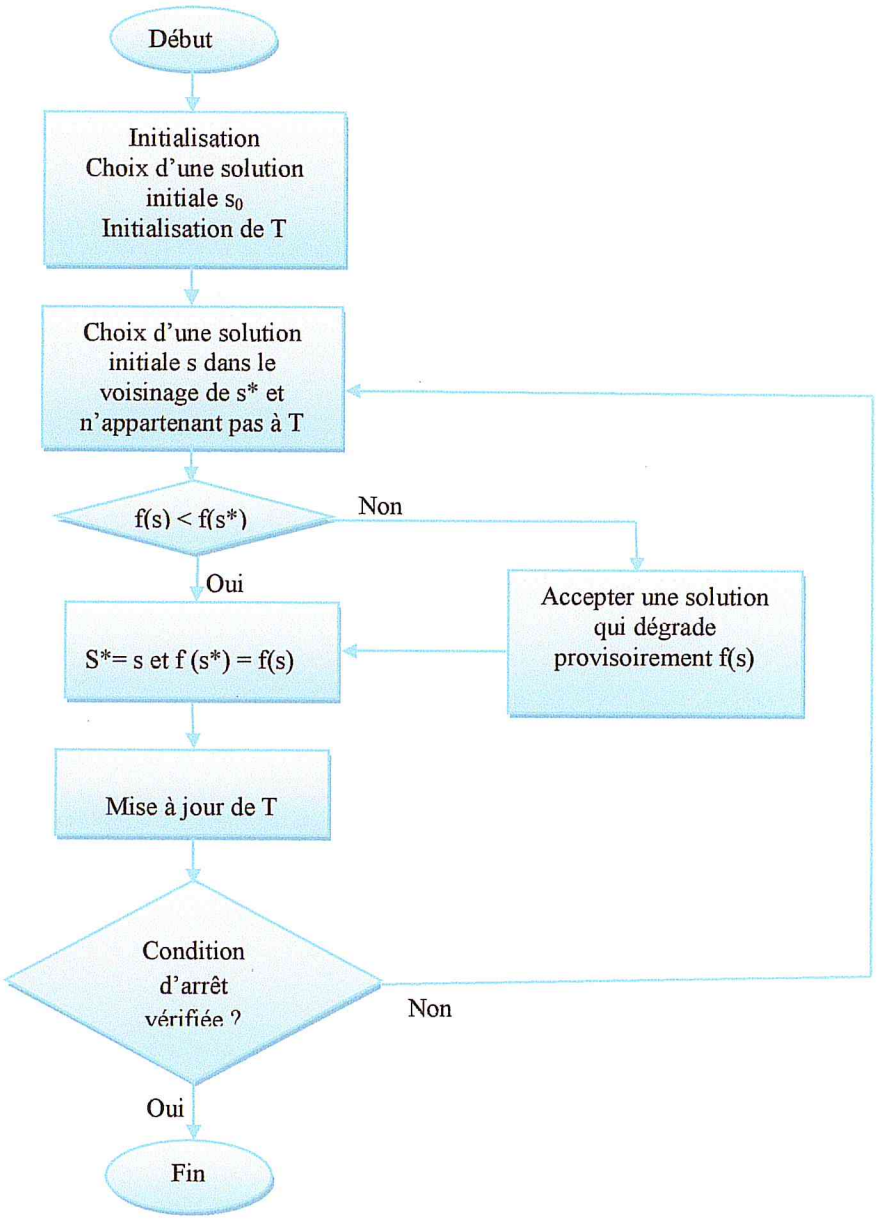


Figure II. 4 : Organigramme relatif au fonctionnement général de la méthode de recherche tabou.

II.2.2.2. Les avantage de l'algorithme de recherche Tabou :

- Grande efficacité : tabou bat tout.
- Fonctionnement simple à comprendre.

II.2.3 .3. Les inconvénients de l'algorithme de recherche Tabou :

- Paramètres peu intuitifs.
- Demande en ressource importantes si la liste des tabous est trop imposante.
- Aucune démonstration de la convergence.

II.2.5. Méta-heuristique de recherche harmonique « HSA » : [8,9]

L'algorithme harmony search (HSA) est un récent algorithme méta-heuristique à été développé par *Lee et Geem* inspiré par la recherche d'une meilleure harmonie musicale lors d'une session de musique improvisée.

Une harmonie musicale est une notion purement esthétique lors d'une combinaison de plusieurs sons simultanés, et au même titre qu'un algorithme d'optimisation tend à obtenir une configuration idéale parmi les états de plusieurs variables selon une fonction d'optimisation donnée, un groupe de musiciens recherche une harmonie parfaite en modulant l'état de chacun

Ici, le vocabulaire employé est directement calqué sur celui de la théorie de la music. Nous parlerons donc d'*harmonie*, pour parler de solutions. L'ensemble des harmonies formera une population on l'appelle *mémoire harmony,* que nous ferons évoluer pendant une certaine succession d'itérations jusqu'à ce qu'un critère d'arrêt soit vérifié.

En effet lorsqu'un musicien improvise un lancement (un ton), habituellement, il suit n'importe laquelle des trois règles :

- Règle 1 : jouant un lancement de sa mémoire.
- Règle 2 : jouant un lancement adjacent d'un lancement de sa mémoire.
- Règle 3 : jouant le lancement totalement aléatoire de la gamme saine et possible.

Par analogie quand l'algorithme HSA affecte une valeur à une variable de décision il suit l'une
des trois règles :

- **Considération de mémoire** *(HMCR)*: le choix d'une valeur quelconque de la mémoire des Harmonies.
- **Ajustement de lancement***(PAR)* : le choix d'une valeur adjacente à la valeur d'HM.

- **Randomisation** : le choix d'une valeur totalement aléatoire dans l'intervalle des valeurs possibles.

II.2.5.1. Algorithme de recherche harmonique :

Début

- Initialisation: générer une mémoire harmonie initiale HM de solutions de taille|HM|=n.
- fixer un Taux d'acceptation *HMCR* ∈ (0, 1).
- fixer un Taux d'ajustement *PAR* ∈ (0, 1).

Tant que (critère d'arrêt non attend) ***faire***

Pour (j=1 à TH) ***faire (tq TH la taille de l'harmonie)***

Sélectionner une harmonie X dans HM ;

Déterminer *hmcr* ∈ (0,1) ;

Si (*hmcr* < *HMCR*) ***alors***

Déterminer *par* ∈ (0,1) ;

Si (*par* < PAR) ***alors***

Ajuster la j^{eme} valeur de X et la insérer dans X' (tq X' le NV harmonie)

Sinon

Insérer directement la j^{eme} valeur de X dans X'

Fin si.

Sinon

Générer un nombre aléatoire valide et insérer le dans X'

Fin si.

Fait.

Si X' vérifier les critères d'optimisations ***alors***

Mètre à jour la mémoire harmonie

Fin si.

Fin tant que.

Fin.

Figure II.5 : Procédure d'Algorithme de recherche harmonique.

II.3. Conclusion

Ce chapitre nous a permis de passer en revue les principales méta-heuristiques généralement utilisées pour la résolution des problèmes d'ordonnancement. Nous avons aussi présenté la méthode de recherche d'harmonique qui est une méta-heuristique prometteuse pour la résolution des problèmes d'optimisation combinatoire difficiles.

Dans le chapitre suivent nous présenterons notre adaptation de cette méta-heuristique au problème du job shop flexible.

Partie II

Implémentation

Chapitre 3:

Application d'algorithme harmony search au job shop flexible

I.1. Introduction :

Dans le cadre de résolution des problèmes d'ordonnancement de type job shop flexible, nous avons présentés dans le chapitre précédent les méta-heuristique les plus connus , considérées par plusieurs chercheurs comme des méthodes bien adaptées au ce type de problème. Notre application porte sur la méthode de recherche harmonique qu'est utilisée pour la première fois sur le problème job shop flexible.

Dans ce chapitre, nous donnerons un aperçu de la démarche suivie pour concevoir le travail réalisé.

I.2. Implémentation de la méthode de recherche harmonique :

Dans notre Implémentation nous suivons les cinq étapes de l'algorithme qui sont les suivantes :

- initialisation des paramètres de l'algorithme HSA.
- initialisation de la mémoire harmonique.
- Improviser une nouvelle harmonie.
- Mise à jour de la mémoire harmonie.
- Réitérer les étapes 3 et 4 jusqu'à satisfaction du critère d'arrêt .

Dans notre travail de conception de la méthode de recherche a particulièrement consisté en l'adaptation de ses différentes étape et operateurs au problème traité. Nous résumons dans ce qui suit les points clé relatifs à cette adaptation :

- En premier lieu il fallait déterminer un codage des solutions du problème d'ordonnancement. Nous avons adopté un codage en deux partie, une pour l'allocation des opérations (MS : machine sélection) et une autre pour leur séquençage (OS : Order Sequensing). Le codage choisi et très utilisé dans la littérature notamment dans le cadre de l'application des algorithmes génétiques.
- En deuxième lieu il fallait adapté les operateurs de l'approche à ce codage, particulièrement l'operateur d'ajustement qui appartient à l'étape d'improvisation de l'harmonie. Pour se faire nous avons développé deux operateurs différents d'ajustement : l'ajustement par valeur adapté à la partie MS, et un operateur d'justement par indice, adapté à la partie OS. Ce dernier à été particulièrement conçu en vu de généré des harmonies réalisable. Effectivement, les manipulation de cette partie sont très sensible puisque on risque de généré des harmonies irréalisables.

La démarche de cet algorithme est proposée dans la figure ci-dessous.

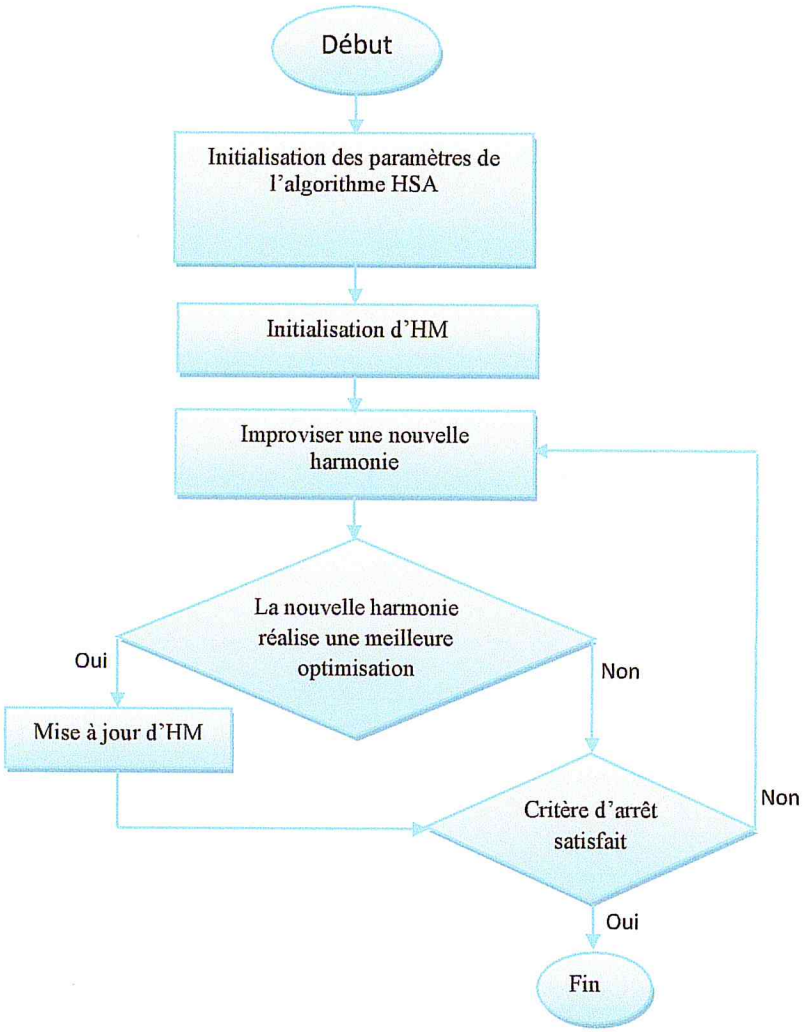


Figure III.1 : Procédure d'optimisation de HSA.

I.2.1. Initialisation des paramètres :

L'initialisation des paramètres aura une influence très importante sur les résultats obtenus par l'algorithme de la recherche harmonique, les paramètres nécessaires sont les suivants :

- **taille de la mémoire harmonique** *(Taille-HM)* **:**

  Ce paramètre représente le nombre d'éléments de l'espace des solutions qui sont présents à chaque itération. Une mémoire de très petite taille ne pourra pas évoluer de manière satisfaisante, car une mauvaise solution aura une influence très importante sur le fitness moyenne de la mémoire d'harmonique. Dans les cas où la mémoire d'harmonique aura une grande taille le temps de calcul seront augmentés.

- **nombre des itérations** *(NB-Itération)* :
  Ce nombre doit être suffisant pour permettre une exploration correcte de l'espace de recherche, mais pas trop grand pour ne pas devenir pénalisant sur le plan du temps de calcul.

- **le taux d'acceptation** *(HMCR : Considération de mémoire d'harmonie)* :
  Si ce taux est trop fort, la probabilité d'acceptation sera augmentée, D'un autre côté, un taux d'acceptation trop faible risque de refusé tous les harmonies de la mémoire.

- **le taux d'ajustement** *(PAR* : Ajustement de lancement) :
  Le taux d'ajustement doit demeurer assez faible afin de ne pas perturber les informations des harmonies de la mémoire.

I.2.2. Initialisation de la mémoire harmonique :

L'initialisation de la mémoire harmonique constitue le point de départ de l'algorithme de la recherche d'harmonique; il est généralement admis que l'efficacité ultérieure de l'algorithme est étroitement liée à la qualité de la mémoire harmonique. En effet, il est intuitivement préférable d'avoir une mémoire harmonie qui constitue un échantillon représentatif du domaine de l'ensemble des solutions du problème.

Dans notre implémentation la mémoire harmonique est remplie par des harmonies aléatoires.

Le remplissage aléatoire présente l'avantage de proposer une mémoire variée, assurant un bon recouvrement de l'espace de recherche. Elle permette de générer une mémoire acceptable, quand aucune information n'est a priori disponible sur la localisation de l'optimum.

I.2.2.1. Codage d'une Harmonie :

Pour le traitement d'un problème d'optimisation par la méthode de la recherche d'harmonique, il est primordial de définir un codage de l'harmonie.

Le codage est une représentation conceptuelle, manipulable par la méthode de la recherche d'harmonique. Dans notre travail nous utilisons un codage qui est représenté sous forme de deux vecteurs numériques contenant toute l'information nécessaire à la description d'une harmonie :

- un vecteur pour le codage des choix de machines pour chaque opérations on l'appel MS « *machine sélection* »

- un vecteur pour codé le séquençage globale des opérations,, on l'appel OS « *opération sélection* ».

Exemple :

	opérations	M1	M2	M3	M4
Job_1	O11		5	3	
	O12	8	2		
Job_2	O21	7			10
Job_3	O31		2	10	3
	O32	3	7	5	4
	O33	12		4	

Tableau III.1 : Exemple d'un problème d'ordonnancement.

Le tableau ci-dessus présente un problème de 4 machines et 3 taches (jobs) tel que :

- La tache_1 : contient deux opérations O11 qui à deux choix machine (la machine 2 avec le temps 5,et la machine 3 avec le temps 3) et O12 qui à deux choix machine (la machine 1 avec le temps 8,et la machine 2 avec le temps 2).
- La tache 2 : contient une seul opération O21 qui à deux choix machine (la machine 1 avec le temps 7,et la machine 4 avec le temps 10).
- La tache_3 : contient trois opérations O31 qui à trois choix machine (la machine 2 avec le temps 2, la machine 3 avec le temps 10, et la machine 4 avec le temps 3), O32 qui à quatre choix machine (la machine 1 avec le temps 3, la machine 2 avec le temps 7, la machine 3 avec le temps 5, et la machine 4 avec le temps 4), et O33 qui à deux choix machine (la machine 1 avec le temps 12, et la machine 3 avec le temps 4).

- ***Partie machine sélection « MS »*** **:** Le MS est un vecteur qui contient des valeurs entiers, son taille est égales à la sommes des opérations pour toutes les taches (jobs). On lire ce vecteur de gauche a droit tel que la 1^{cr} case correspondre à la 1^{cr} opération de 1^{cr} tache et la d^{er} case à la d^{er} opération de d^{er} tache ,donc chaque case correspondre à une opération telle que sa valeur doit être entre un et le nombre maximal de choix machine de cette opération.

La figure suivante représente la partie machine sélection « *MS* » pour une harmonie de l'une des solutions du problème de l'exemple dans le tableau III.1.

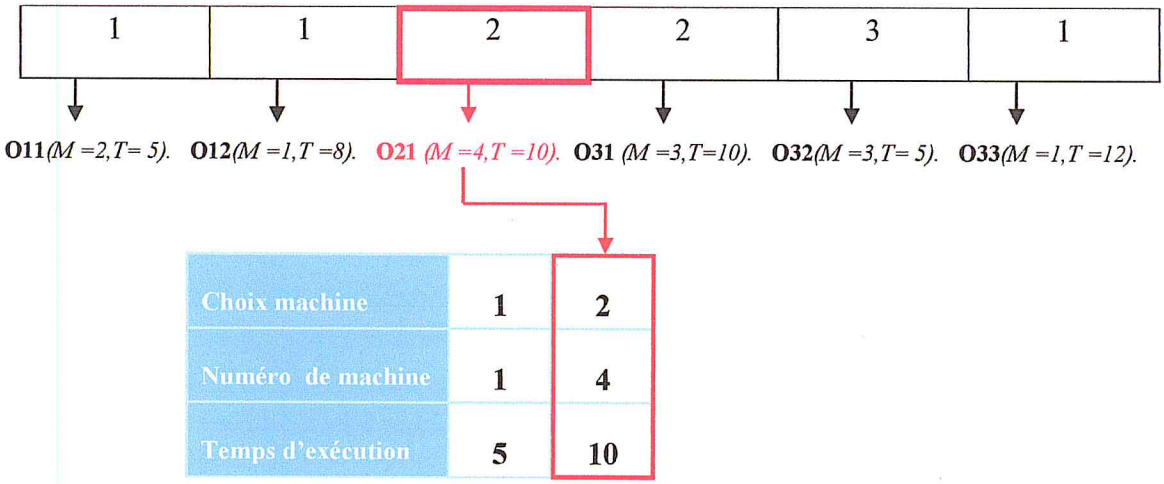


Figure III.2 : Codage d'une partie MS d'une harmonie.

Donc la 1^{cr} opération de la 2^{cmc} tache « O21 » prend le 2^{cmc} choix (2) qui correspondre à la machine 4 avec le temps d'exécution 10 que nous avons présenté dans le petit tableau ci-dessus. De cette manière on lit toutes les valeurs du vecteur MS jusqu'à la dernière opération de la dernière tache.

- ***Partie opération sélection « OS »*** **:** L'OS est un vecteur qu'il est constitué d'une série de nombres entiers qui représentent l'ordre d'exécution des opérations pour toutes les taches, tel que ce vecteur pris le même taille que le vecteur MS. Chaque opération est représentée par le numéro de la tache qu'elle appartient.

La figure suivante représente un vecteur « OS » de l'exemple précédent.

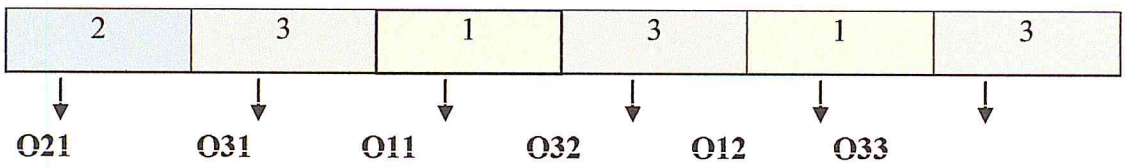


Figure III.3 : Codage d'une partie OS d'une harmonie.

Donc la suivie d'exécution des opérations de cet exemple prend l'ordre suivant : O21→O31→O11→O32→O12→O33.

II.2.2.2. Décodage d'une harmonie *(évaluation d'harmonie)* **:**

On doit évaluer chaque harmonie pour présenter toutes les informations concernons cette dernière tels que : la machine qui va exécuter chaque opération, dates de début et de fin pour chaque opération la totalité de temps d'exécution,... etc.

L'évaluation des harmonies suivre les étapes suivants :

Pour chaque lancement de la partie OS :

(1)- Identification de l'opération et de la tache.

(2)- Retrouve le choix de machine correspondent dans la partie MS.

(3)- Détermine à partir de choix machine et des données de problème, la machine et le temps d'exécution.

(4)- Placer l'opération en respectent les contraints de précédence et de disjonction.

Fin pour.

Exemple :

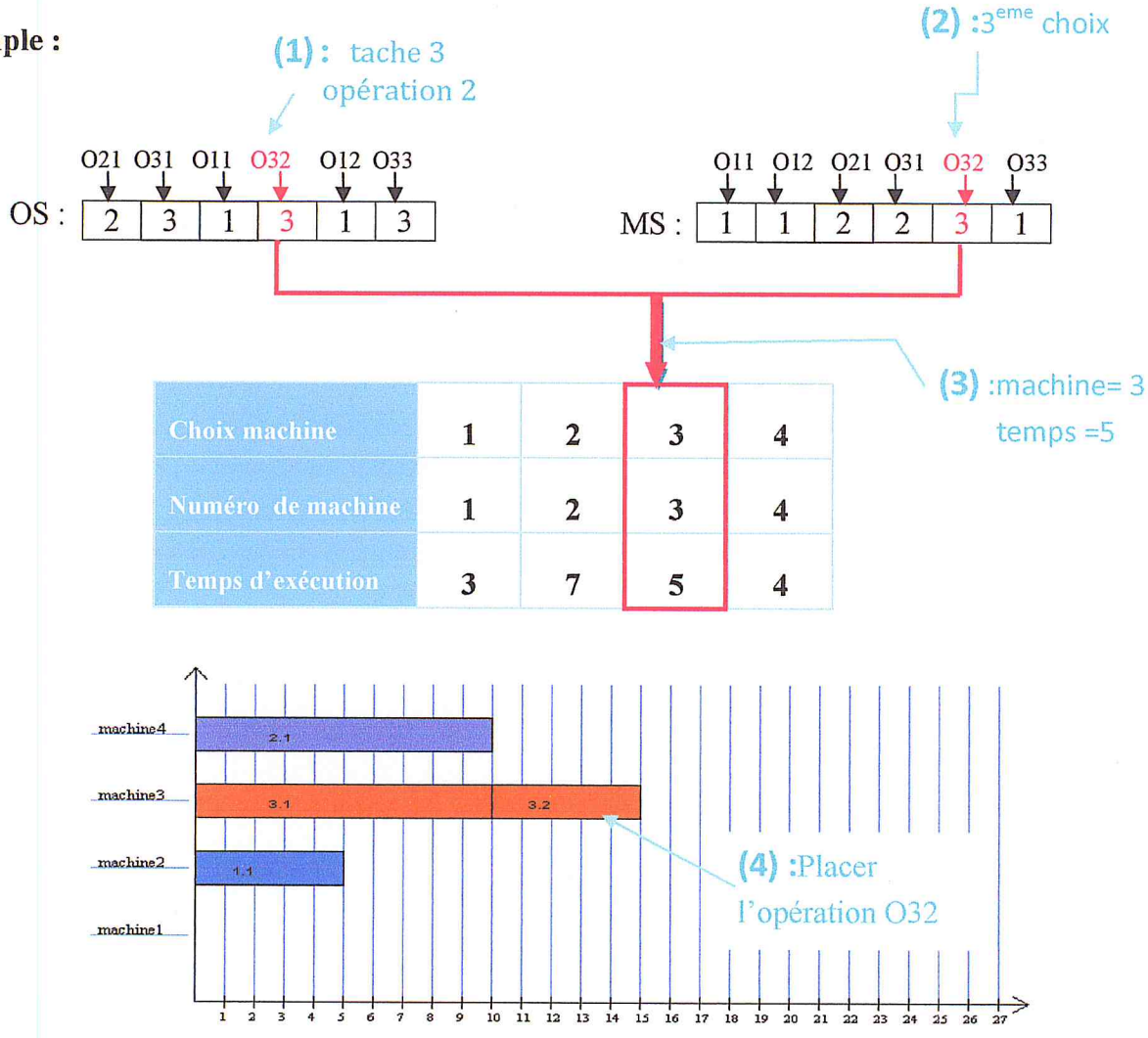


Figure III.4 : Décodage d'une harmonie.

L' évaluation complet de ce harmonie donne la suivie d'exécution suivant :

O21*(M 4,T 10)* →O31*(M 3,T 10)*→O11*(M 2,T 5)*→O32*(M 3,T 5)*→O12*(M 1,T 8)*→O33*(M 1,T 12)*.

II.2.3. Improvisation d'une nouvelle harmonie :

L'improvisation de notre nouvelle harmonie se fait à l'aide des opérateurs *sélection* et *l'ajustement*. . Apres la présentation l'operateur de sélection, on va détailler dans ce qui suit l'operateur d'ajustement selon les deux parties MS et OS:

II.2.3.1. La Sélection :

La sélection est utilisée afin de favoriser au cours du temps les harmonies les mieux adaptés. (qui ont un meilleur fitness, pour nous le fitness sera le plus sauvent la valeur de la fonction objective associée à l'harmonie).

C'est un operateur de l'algorithme génétique qui n'appartient pas aux operateurs de la méthode de recherche d'harmonique de base, mais nous le testerons afin d'améliorer la méthode.

L'organigramme suivant montre la stratégie de la sélection d'harmonie utilisé dans notre algorithme de recherche harmonique.

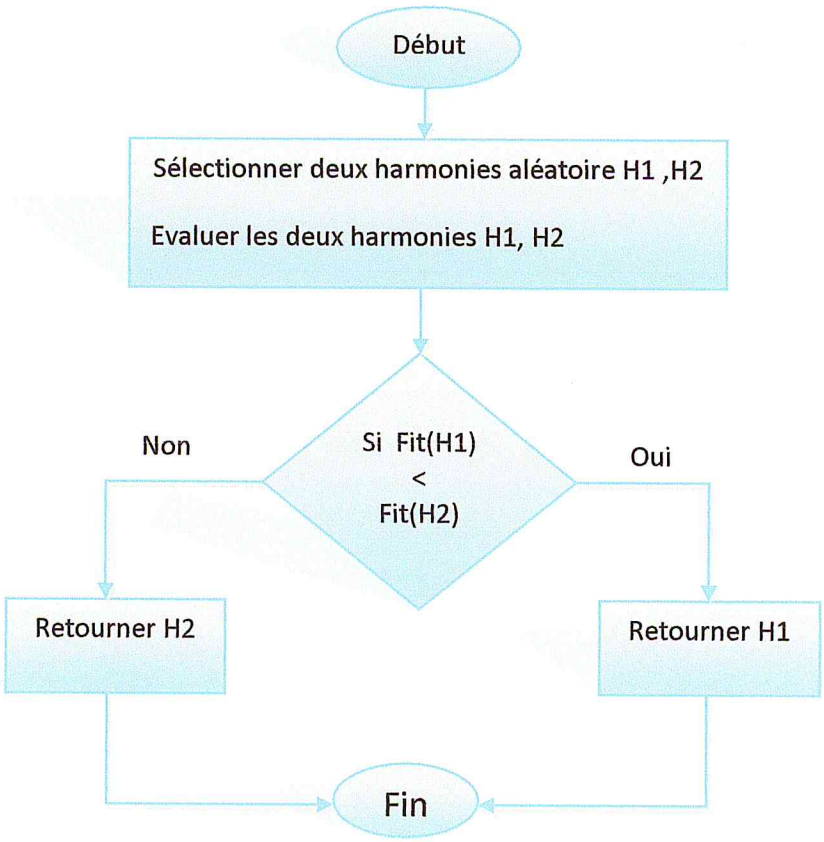


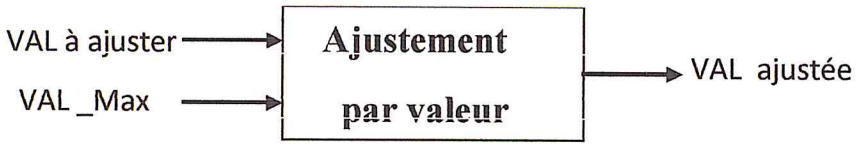
Figure III.5 : Organigramme détaillé de l'opérateur de sélection.

II.2.3.2. L'ajustement :

L'ajustement est utilisé afin de conserver une certaine variété de l'harmonie Improviser, et de garantir d'une bonne exploration de l'espace des solutions.

Nous avons développés deux types d'ajustement : **par valeur** et **par indice**.

- ***L'opérateur d'ajustement par valeur :*** consiste a modifié une valeur donnée soit on l'augmente ou on la diminue d'une unité, mais la nouvelle valeur doit être entre un et une valeur maximum donnée.



L'organigramme suivant montre la stratégie d'ajustement par valeur utilisé dans notre algorithme de recherche harmonique.

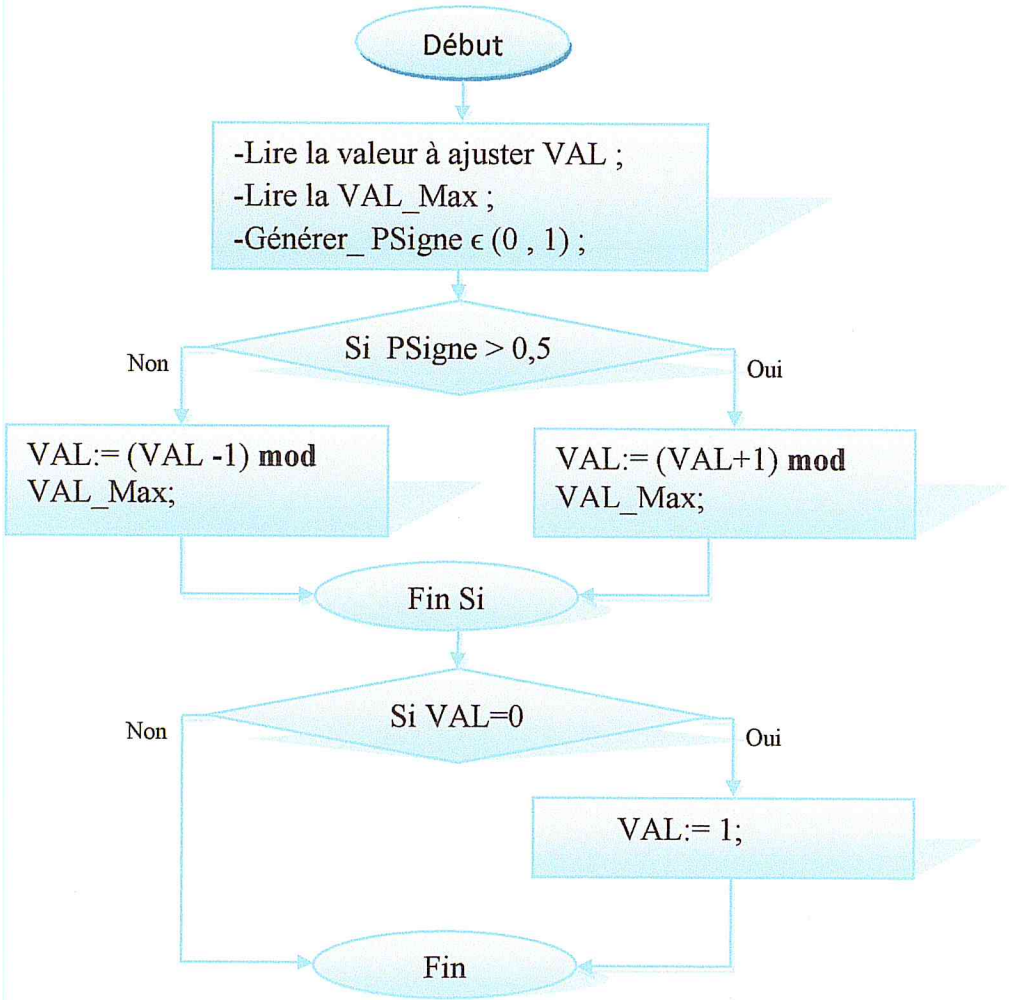


Figure III.6 : Organigramme détaillé de l'opérateur ajustement par valeur.

Exemple :

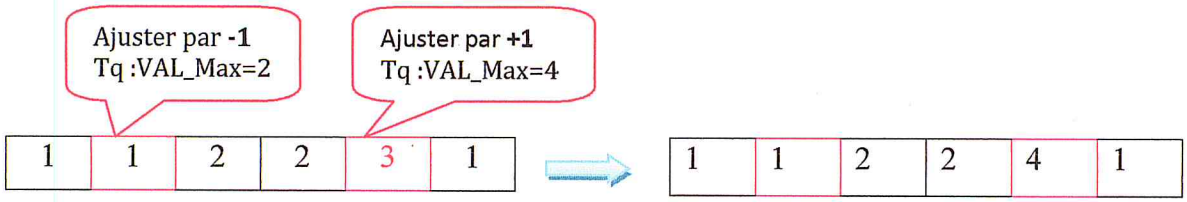


Figure III.7: Ajustement par valeur .

- ***L'opérateur d'ajustement par indice :*** consiste à ajusté l'indice d'une valeur dans un vecteur soit on la décale vers la droite ou vers la gauche par une unité, tel que le nouveau indice doit être compris entre zéro et la taille de ce vecteur moins un.

. L'organigramme suivant montre la stratégie d'ajustement par indice utilisé dans notre algorithme de recherche harmonique

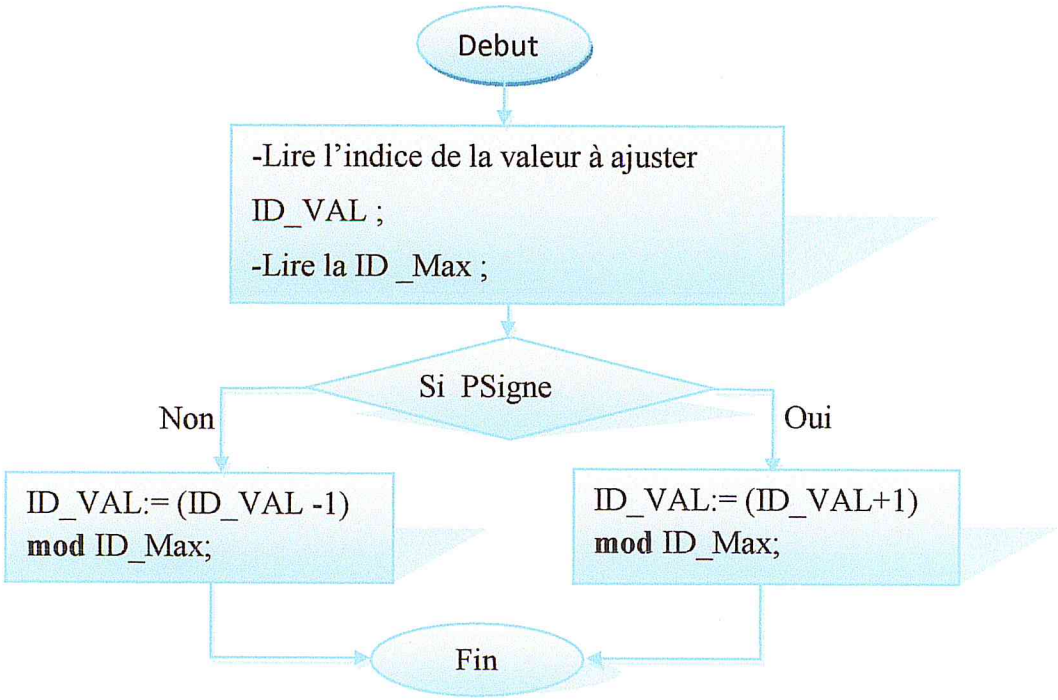


Figure III.8 : Organigramme détaillé de l'opérateur ajustement par indice.

Exemple :

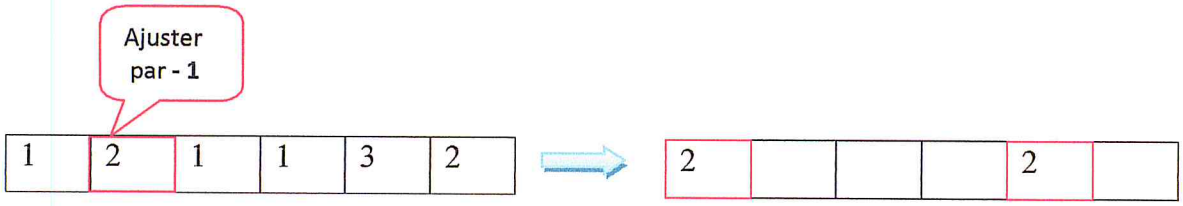


Figure III.9: Ajustement par indice.

III.2.3.3. Improvisation de la partie MS :

Pour la Improvisation de la partie MS on suivre quelque étapes qui sont les suivants :

Pour i de un à nombre des lancements de la partie MS :

(1): Sélectionner un Harmonie H dans HM Selon le principe de sélection.

(2): déterminer *hmcr* ∈ (0. 1).

(3): déterminer *par* ∈ (0. 1).

(4): **si** *hmcr* < HMAR et *par* < PAR **alors**

(5): ajuster par valeur le lancement H_MS [i] et la insérer dans NH_MS [i] (NH nouvelle harmonie); **Fin si**.

(6): **si** *hmcr* < HMCR et *par* > PAR **alors**

(7): insérer directement le lancement H_MS [i] dans NH_MS [i]; **Fin si**.

(8): **si** *hmcr* >HMCR **alors**

(9): insérer un lancement aléatoire dans NH_MS [i]; **Fin si**.

Fin pour.

L'organigramme suivant détaillés les étapes de l' improvisation de la partie MS .

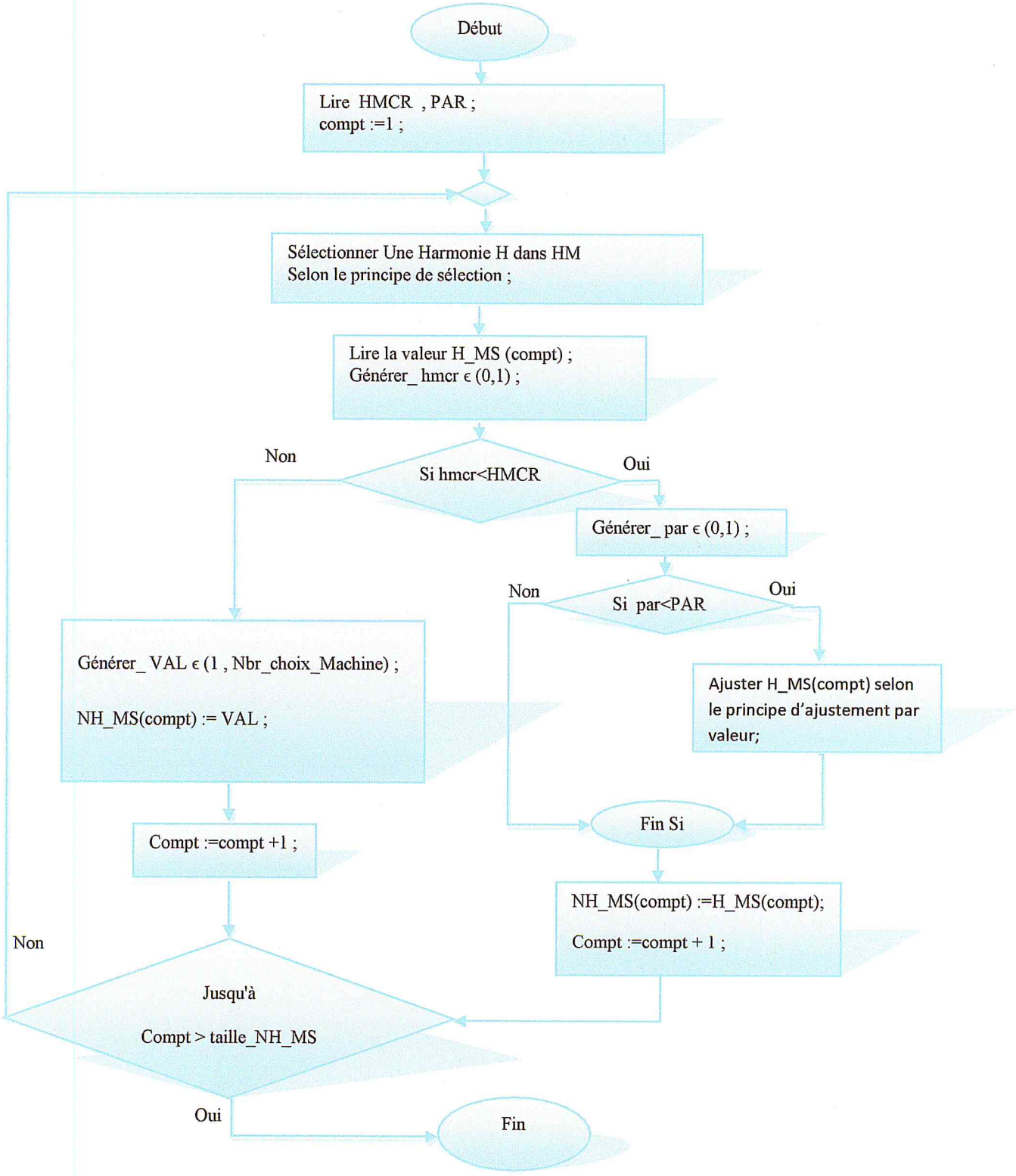


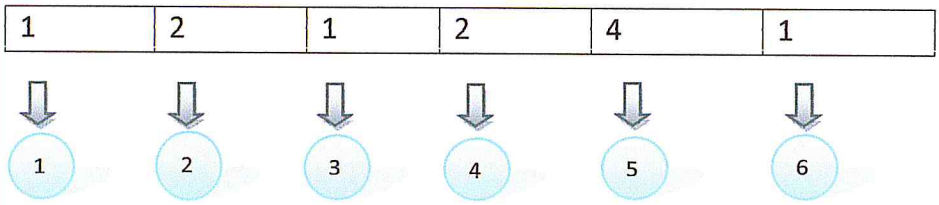
Figure III.10 : Organigramme détaillé l'Improvisation de la partie MS.

On va présenter maintenant un exemple sur l'improvisation de la partie MS d'une nouvelle harmonie pour expliquer ce principe de manière plus claire.

Exemple :

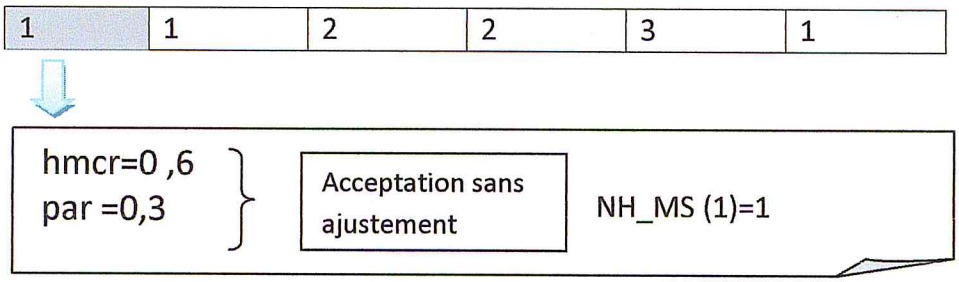
On utilise le même exemple (tableau III.1) avec un taux d'acceptation HMCR=0,9 et un taux d'ajustement PAR =0,1.

NH_MS : Supposant que la partie MS de notre nouvelle harmonie NH contient les valeurs ci-dessous.

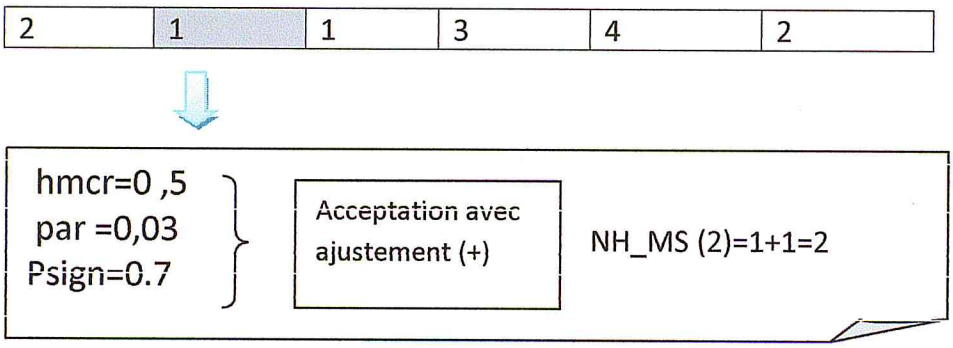


1 : On Sélectionne une Harmonie H dans HM Selon le principe de la sélection puis on génère un taux d'acceptation « *hmcr* », un taux d'ajustement « *par* » et une probabilité « *PSign* » pour vérifie le signe d'ajustement. On va répéter ce procédure pour le reste des lancements de NH_MS.

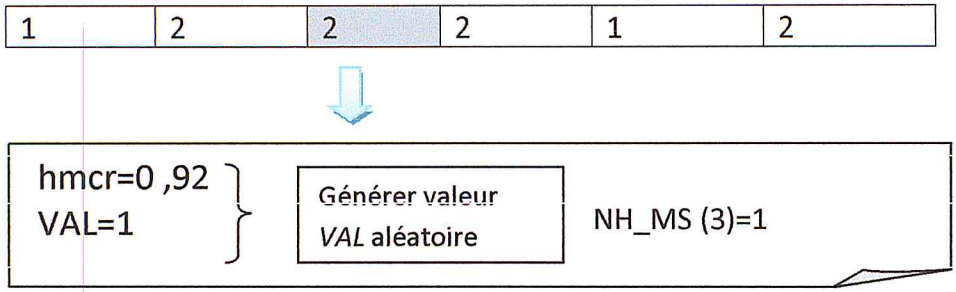
Voila la partie MS de l'harmonie H que nous avons sélectionnée pour le 1er lancement.



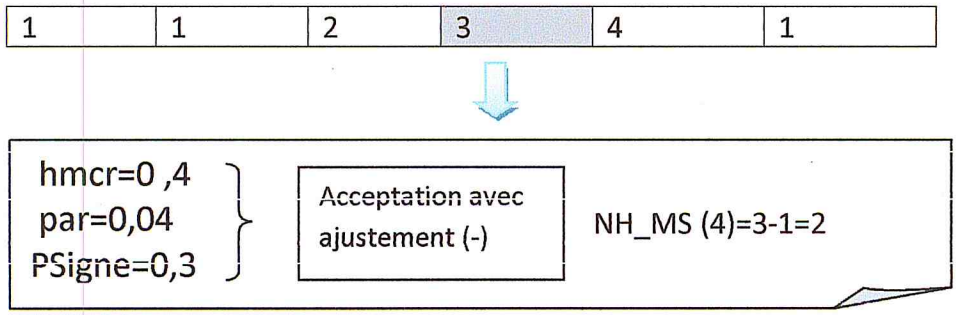
2 :Voila la partie MS de l'harmonie H que nous avons sélectionnés pour le 2eme lancement .



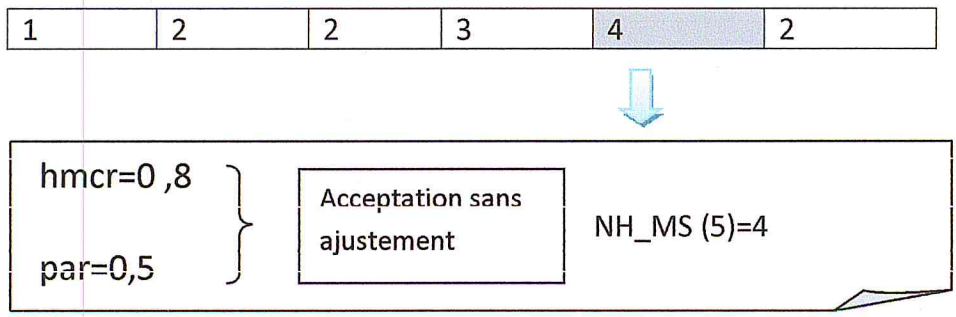
3 : Voila la partie MS de l'harmonie H que nous avons sélectionné pour le 3eme lancement.



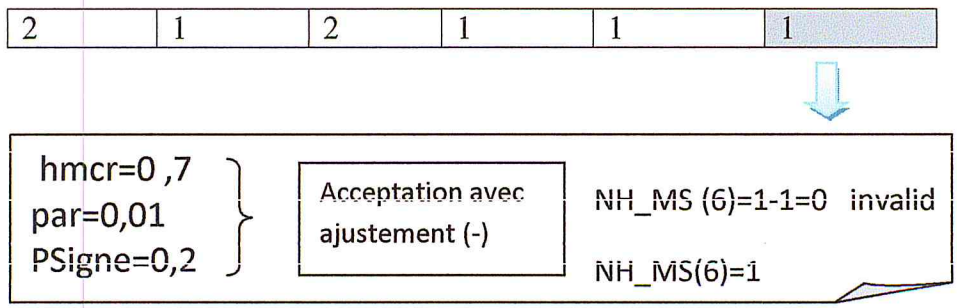
4 :Voila la partie MS de l'harmonie H qui nous avons sélectionnés pour le 4eme lancement.



5 : Voila la partie MS de l'harmonie H qui nous avons sélectionné pour le 5eme lancement.



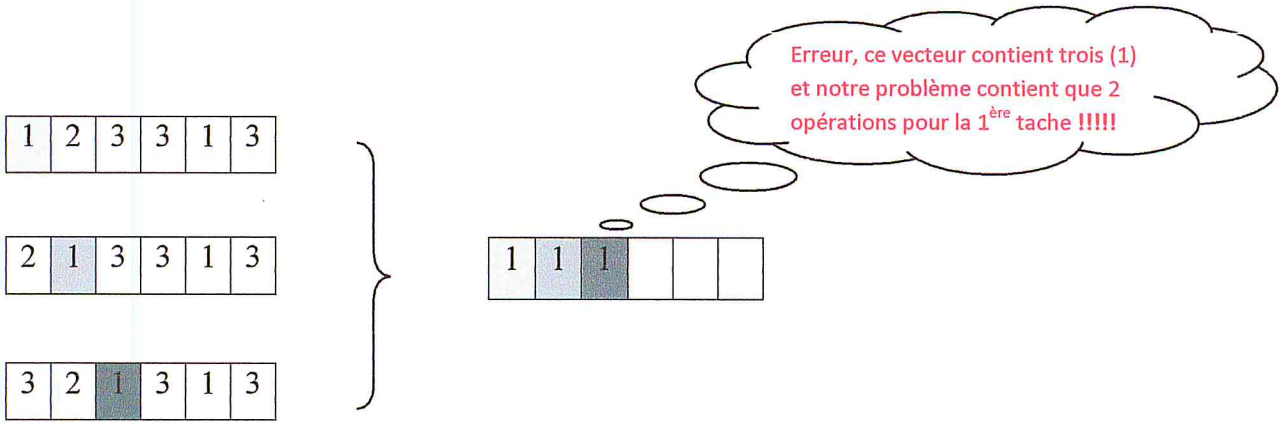
6 : Voila la partie MS de l'harmonie H qui nous avons sélectionné pour le 6eme lancement.



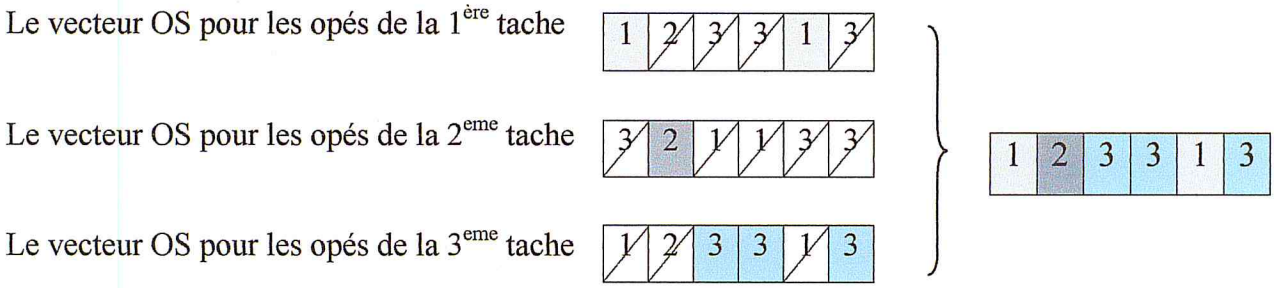
III.2.3.4. Improvisation de la partie OS :

L'improvisation de la partie OS est complètement différente que celle avec la partie MS, pour les raisons suivants :

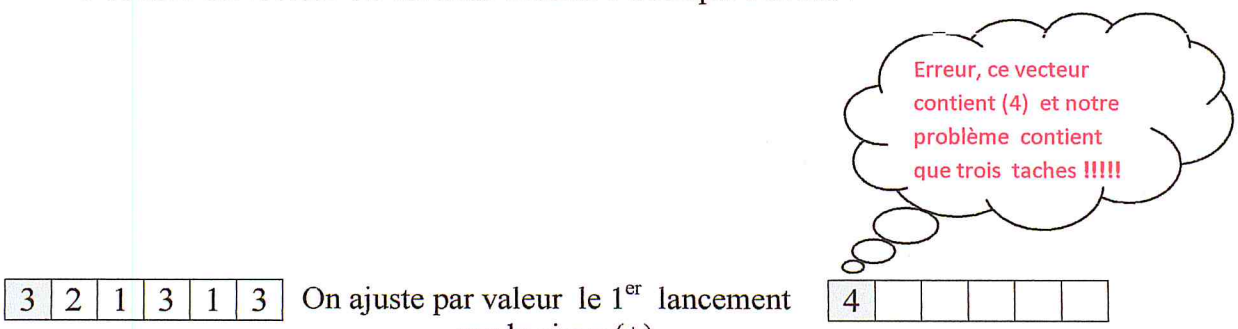
- ***Dans la sélection :*** Si On sélectionne une harmonie pour chaque opération comme la partie MS on va risquer d'obtenir un vecteur OS invalide comme l'exemple suivant.



Pour résoudre ce problème on doit sélectionner une seul harmonie pour toutes les opérations de la même tache, comme suivant :



- ***Dans l'ajustement :*** Si On ajuste par valeurs comme dans la partie MS on va risquer d'obtenir un vecteur OS invalide comme l'exemple suivant :



Pour résoudre ce problème on doit utiliser l'ajustement par indice, comme suivant :

3	2	1	3	1	3

On ajuste par indice le 1er lancement par le signe (+)

	3				

Pour l'improvisation de la partie OS nous avons suivis quelque étapes:

Pour i de un à nombre des taches dans le problème à optimiser :

(1): Sélectionner Une Harmonie H dans HM Selon le principe de sélection.

(2): déterminer *hmcr* ∈ (0, 1).

(3): déterminer *par* ∈ (0, 1).

(4): **si** *hmcr* < HMCR et *par* < PAR **alors**

(5): ajuster par indice les opérations de la tache i de H_OS et la insérer les dans NH_OS selon ses indices ; **Fin si**.

(6): **si** *hmcr* < HMCR et *par* > PAR **alors**

(7): insérer directement les opérations de la tache i de H_MS dans NH_OS ; **Fin si**.

(8): **si** *hmcr* >HMCR **alors**

(9): insérer les opérations de la tache i d'un H_OS aléatoire dans NH_MS ; **Fin si**.

Fin pour.

L'organigramme suivant présente l'improvisation de la partie OS :

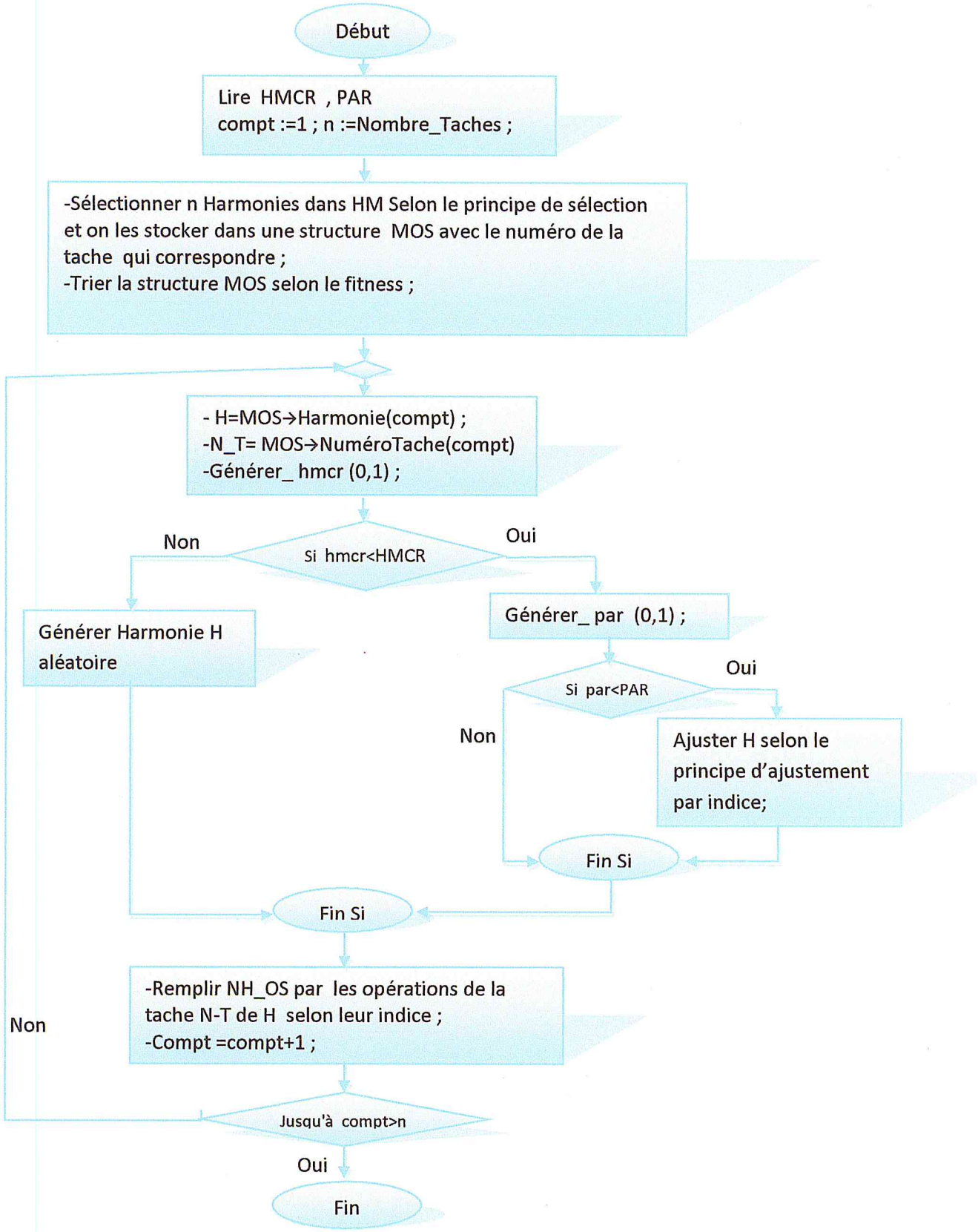


Figure III.11 : Organigramme détaillé de l'improvisation de la partie OS.

III.2.4. Mise à jour de la mémoire harmonique :

Lorsque on termine l'improvisation de notre nouvelle harmonie avec ses deux parties MS et OS on doit faire une mise à jour de la mémoire harmonie selon le principe suivant : Si le nouveau harmonie est meilleur que le plus mauvais harmonie dans le HM, en terme de valeur de fonction objective (fitness), la nouvelle harmonie est incluse dans HM et la plus mauvaise harmonie existante est exclue de HM.

III.2.5. Réitérer les étapes 3 et 4 jusqu'à satisfaction du critère d'arrêt :

Dans l'implémentation de notre algorithme de recherche harmonique, le critère d'arrêt c'est atteindre le nombre maximum des itérations cité.

III.3. Conclusion :

A l'aide d'exemples simples, nous avons présenté dans ce chapitre la démarche suivie pour la réalisation de notre algorithme de recherche.

Le chapitre suivant sera consacrée à sa mise en œuvre et à la présentation des différents résultats de test et de validation.

Chapitre 4:

Mise en œuvre et résultats

IV.1. Introduction :

Dans le chapitre précédent nous avons présenté notre adaptation de la méthode de recherche harmonique pour la résolution des problèmes d'ordonnancement de type job shop flexible. Nous présenterons dans ce dernier chapitre, d'une part l'implémentation de l'approche développée en utilisant le langage JAVA sous l'environnement NetBeans, et d'autre part un ensemble de résultats validant l'implémentation.

IV.2. Les outils et les langages :

On a utilisé l'éditeur de NetBeans avec la version 6.8 (www.netbeans.org) pour la réalisation de notre implémentation et JAVA comme un langage de programmation.

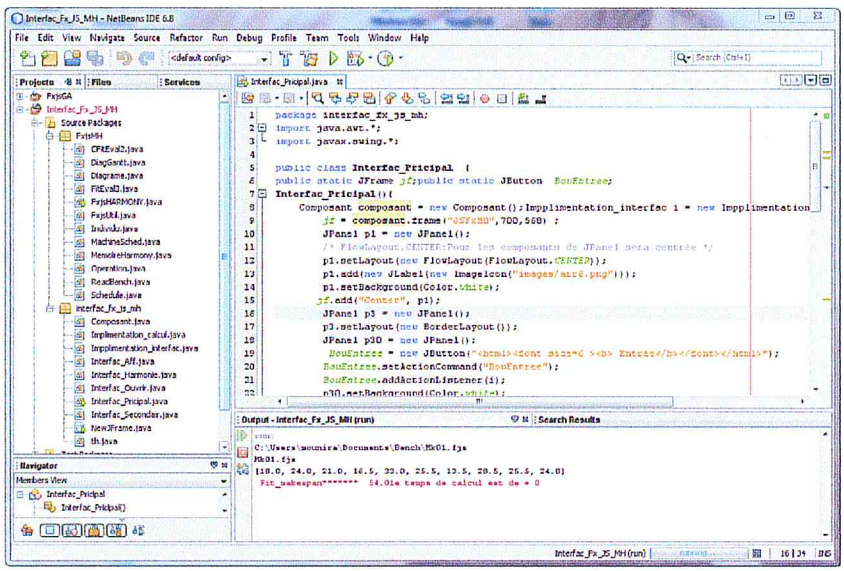


Figure IV.1 : Interface de l'éditeur NetBeanS.

Pour quoi Java:

Nous avons choisi le langage JAVA pour les raisons suivantes :

- C'est un langage orienté objet qui met à la disposition du développeur plusieurs Paquetages prêt à l'utilisation (javax.swing , java.io....etc).
- Offre une encapsulation, la généricité et la réutilisation de notre code métier plus facilement car il offre beaucoup de souplesse.
- La disponibilité de la documentation et de l'assistance (forums).
- La disponibilité d'un ensemble de classes JAVA au niveau du laboratoire SRP du CDTA, développées dans le cadre de travaux relatifs à l'utilisation de méta-heuristiques pour la résolution de problèmes d'ordonnancement type job shop flexible.

IV.3. Application :

Dans cette partie nous allons présenter les principales interfaces et fenêtres d'affichage de l'application réalisée.

IV.3.1. Les fenêtres principales :

L'interface principale de notre application contient : (voir la figure IV.2).

- Un Menu Bar de trois Menus qui sont les suivants : file, méthode et aide.
- Un bouton principal « *Entré* » qui nous permet de passer à la 2éme interface dans la figure IV.3
- Le Menu méthode contient les deux Menu Item qui représente deux versions de notre approche : recherche harmonique avec et sans sélection. Le chois de la 1er version nous permet de passer à l'interface dans la figure IV.4.

Figure IV.2 : Interface principale de notre impiémentation.

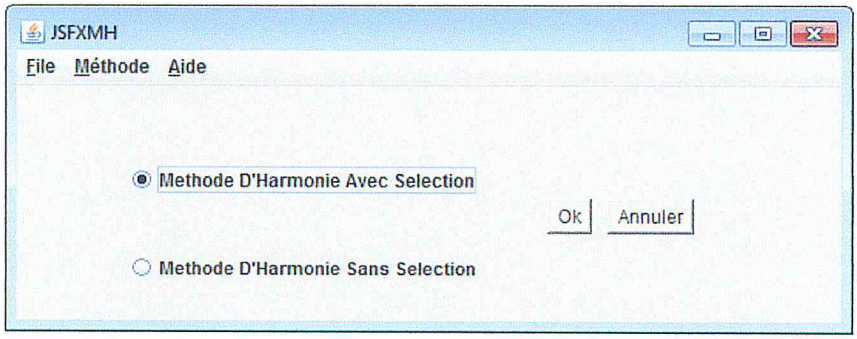


Figure IV.3 : Interface de choix de version.

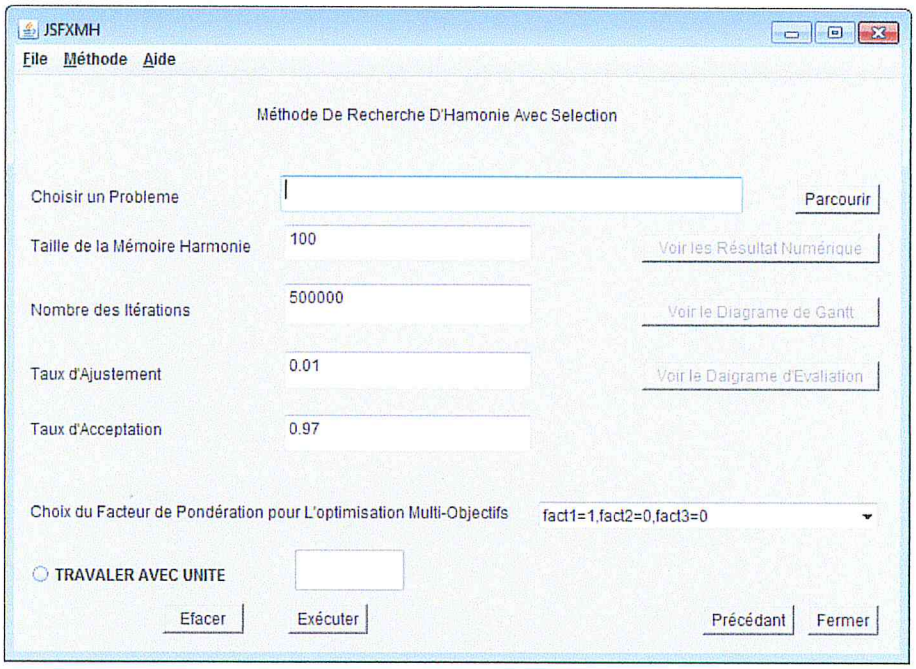


Figure IV.4 : Interface de la méthode de recherche d'harmonique.

L' interface ci-dessus permet la saisie des paramètres d'algorithme, elle contient plusieurs boutons et zones de text plus une liste de choix:

- Le bouton « *parcourir* » permet la selection d'un probléme test à partir d'une fenetre d'éxploration, (Voir la figure IV.5).
- la liste de choix qui nous permet de choisir un facteur de pondération pour l'optimisation multi-objectif, (Voir la figure IV.6).
- Le bouton « *les résultats numérique* » permet la visualisation des resultats numeriques aprés exécution, (voir la figure IV.7).
- Le bouton « *diagramme de Gantt* » permet l'affichage du diagramme de Gantt correspendant à la meuilleure solution, (Voir figure IV.8).
- Bouton « *diagramme d'évolution* » représente l'évolution de la fonction objective, (Voir la figure IV.9).

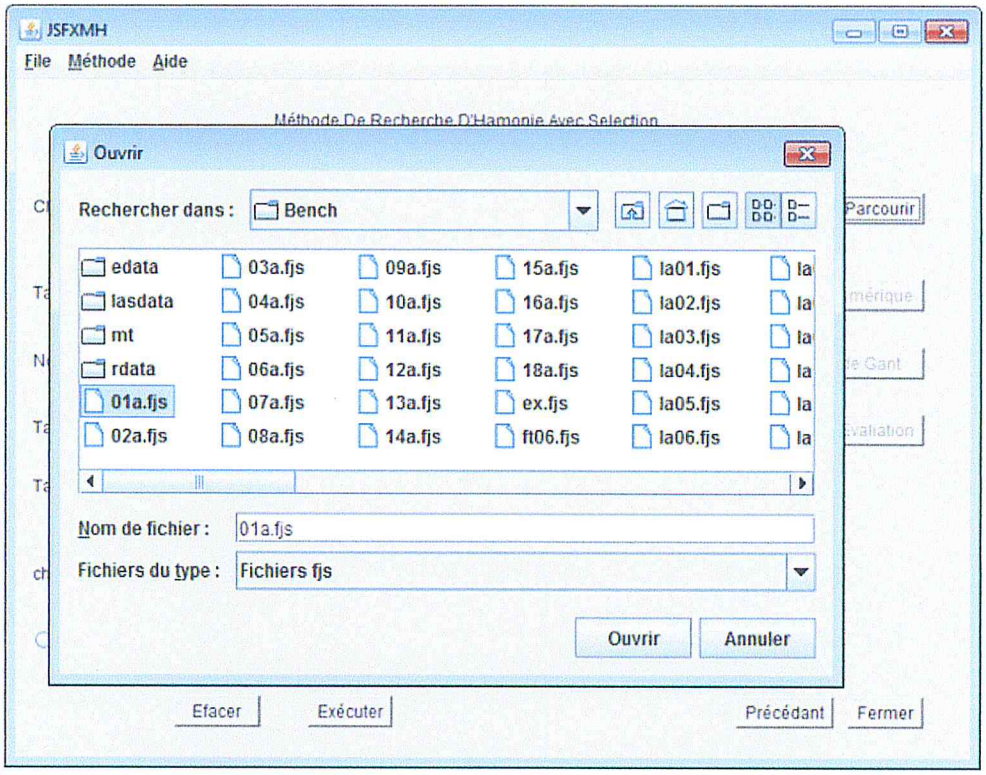


Figure IV.5 : Fenêtre d'exploration pour le choix du Benchmark.

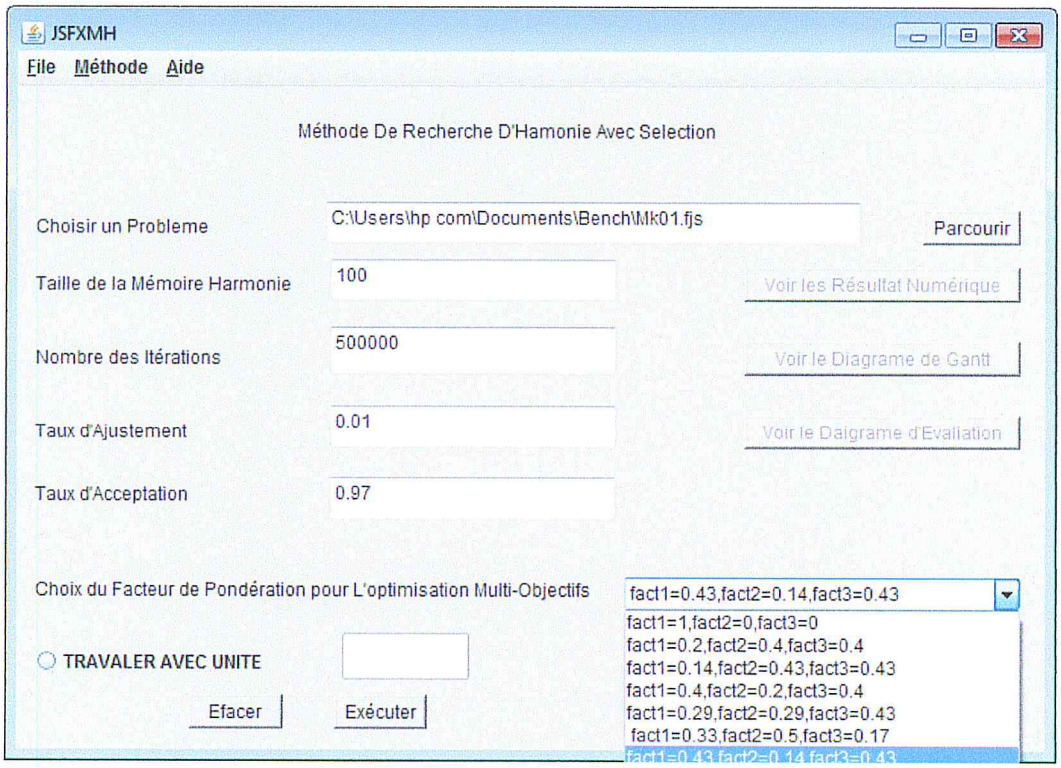


Figure IV.6 : Choisir les facteurs de pondération pour L'optimisation multi-objective.

IV.3.2. Les fenêtres d'affichages :

Figure IV.7 : Interface d'affichage des résultats numériques.

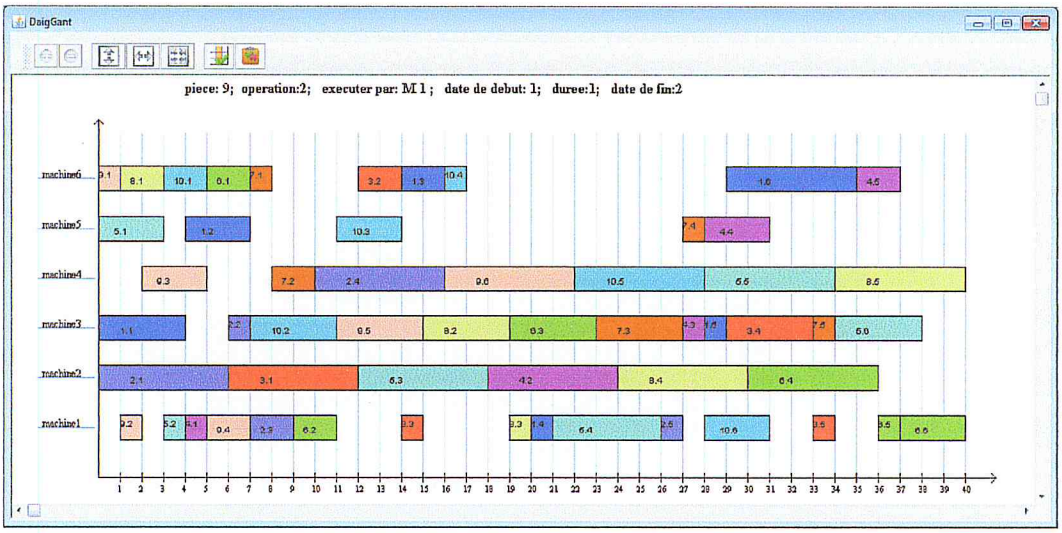


Figure IV.8 : Interface d'affichage du diagramme de Gantt.

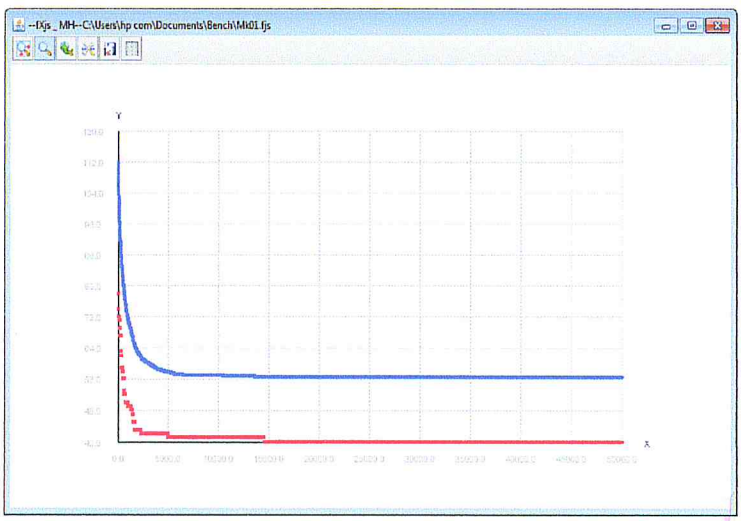


Figure IV.9 : Interface d'affichage du diagramme d'évolution.

IV.4. Résultats de l'application de la recherche d'harmonique :

L'objectif de cette partie est de donner une synthèse des résultats relatifs à l'utilisation de l'approche développée pour la résolution des problèmes d'ordonnancement type job shop flexible. Nous validerons ainsi notre approche dans les cadres mono-objectifs et multi-objectifs.

Dans un premier temps, nous présenterons la phase de détermination empirique des paramètres de fonctionnement de la méthode. Nous avons aussi choisi d'évaluer l'opérateur de sélection et de comparer ses performances au sein de notre approche. Les différents testes ont été réalisés sur des benchmark connus en utilisant un PC avec un processeur Intel (R) Pentium (R) de 2.16 GHz et 2.00 Go de RAM.

IV.4.1. Les benchmarks [1] :

On ordonnancement d'ateliers de type Job Shop Flexible il est souvent utile d'avoir des *Benchmarks* pour testé les modèles d'ordonnancement et surtout les méthodes de résolution appropriées.

Dans notre étude, nous utilisons un échantillon de ces benchmarks comme des problèmes tests, afin d'évaluer la performance de la stratégie méta-heuristiques employée.

Un grand nombre de benchmarks de Job Shop Flexible sont proposés par plusieurs auteurs. Certains sont largement employés dans les recherches. Parmi ces benchmarks, on peut citer les classes suivantes :

IV.4.1.1. BRdata : Cette classe est constituée de 10 problèmes décrits par Brandimarte (1993). Le nombre de jobs n varie entre 10 et 20, le nombre de machines M varie entre 4 et 15, le nombre d'opérations pour chaque job varie entre 5 et 15.
La moyenne du nombre de machines par opération (nommée aussi flexibilité et notée *flex.*), varie entre 1.43 et 4.10. Les machines considérées dans cette classe sont non-reliées.

IV.4.1.2.BCdata : Cette classe est constituée de 21 problèmes définis par Barnes et Chambers (1996) et construits à partir des instances de type job shop classique (mt10, la24, la40). Le nombre de jobs n varie entre 10 et 15, le nombre de machines varie entre 11 et 18, le nombre d'opérations constituant chaque job varie entre 10 et 15 et la moyenne du nombre de machines par opération varie entre 1.07 et 1.30. Les machines considérées dans cette classe sont identiques.

IV.4.1.3. DPdata : Cette classe est constituée de 21 problèmes fournis par Dauzère-Pérès et Paulli (1997). Le nombre de jobs *n* varie entre 10 et 20, le nombre de machines varie entre 5 et 10 et la moyenne du nombre de machines par opération varie entre 1.13 et 5.02. Les machines considérées dans cette classe sont de deux types : identiques et non-reliées.

IV.4.2. Détermination des paramètres de l'algorithme :

Dans le but d'évaluer les performances de notre méta-heuristique et pour pouvoir montrer l'influence des paramètres de dimensionnement sur l'Algorithme de recherche d'harmonique, une série d'essais a été effectuée sur un échantillon de benchmarks de la classe BRdata (MK01, MK04 et MK09). La série d'essais s'est faite en utilisant différentes valeurs de paramètres combinés et deux versions de la recherche d'harmonique. La première sans l'utilisation de l'opérateur de sélection et la deuxième en utilisant ce dernier.

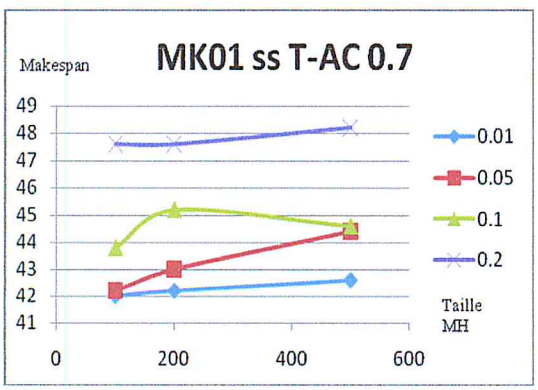
Les jeux de paramètres utilisés sont les suivantes :

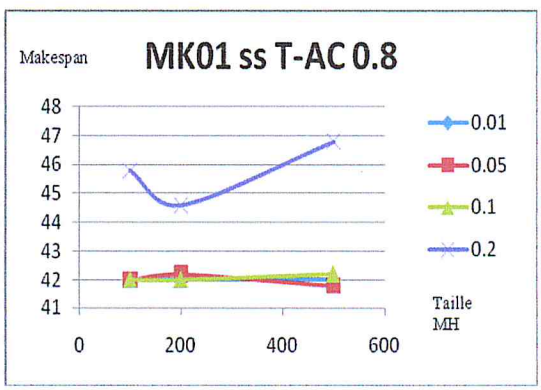
- La taille de la mémoire harmonique : variée entre 100,200 et 500.
- Le nombre d'itérations : fixé à 500000.
- Le taux d'acceptation : varié entre 0.7, 0.8, 0.9, 0.95.
- Le taux d'ajustement : varié entre 0.01, 0.05, 0.1, 0.2.

Pour chaque problème cinq expériences ont été réalisées et nous prenons leur moyenne arithmétique. Les résultats obtenus sont représentés dans des graphiques associant la valeur de la fonction objective aux tailles de la mémoire harmonique testées. Les séries de tests pour chaque benchmark sont représentées sur quatre graphiques. Chacun correspondant à une valeur donnée du taux d'acceptation et aux quatre valeurs du taux d'ajustement.

IV.4.2.1. Sans l'opérateur de sélection :

- ***Le problème MK01 :***





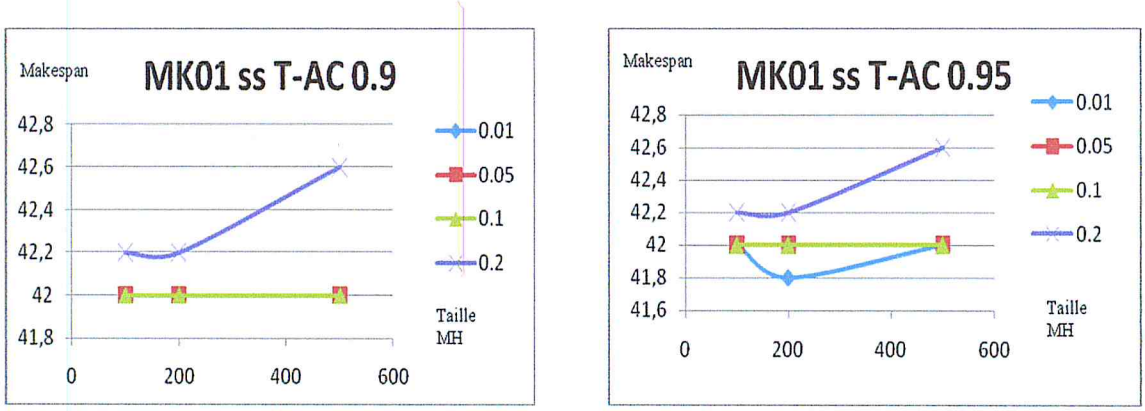


Figure IV.10 : Graphique des résultats de résolution du problème MK01 sans l'opérateur de sélection.

- ***Le problème MK04 :***

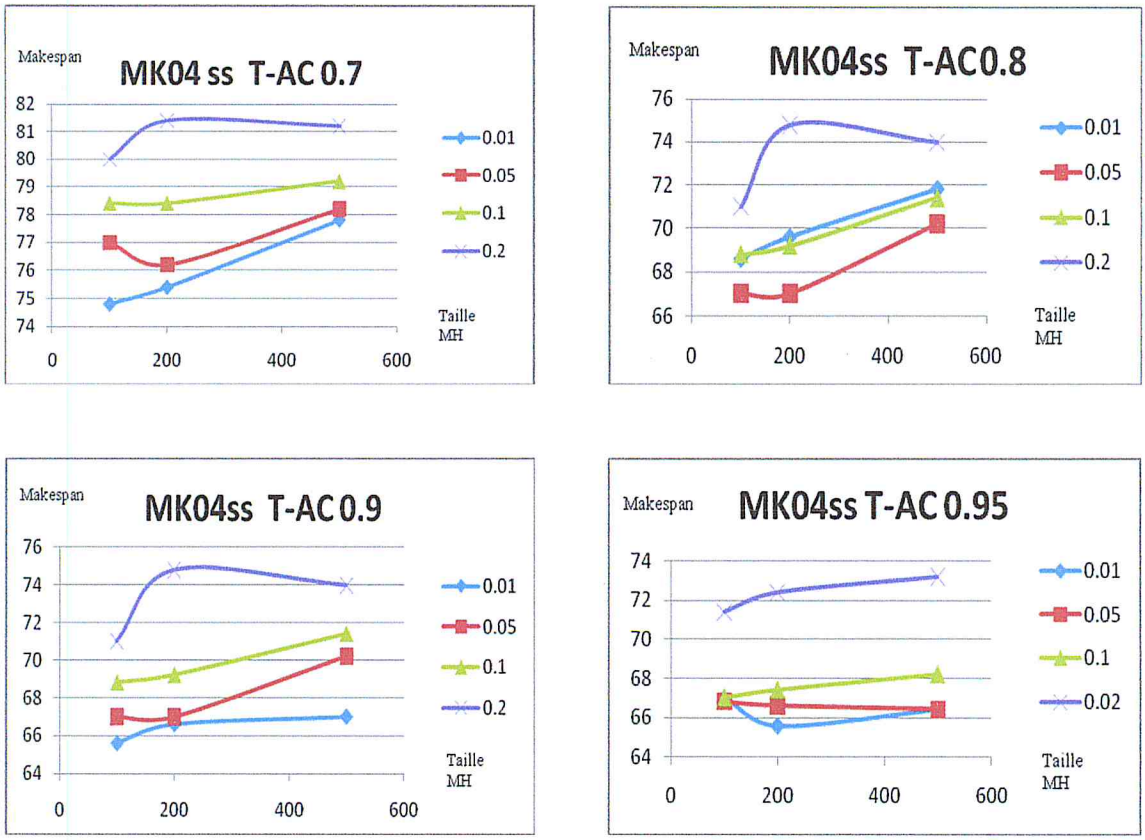
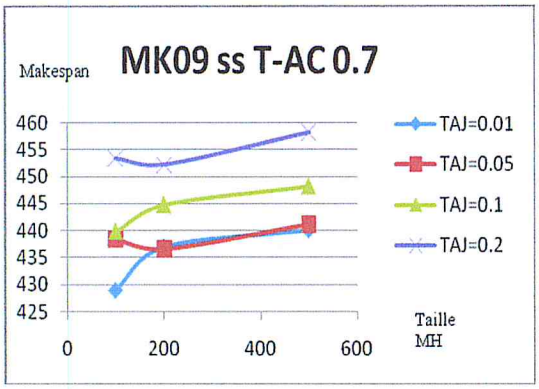
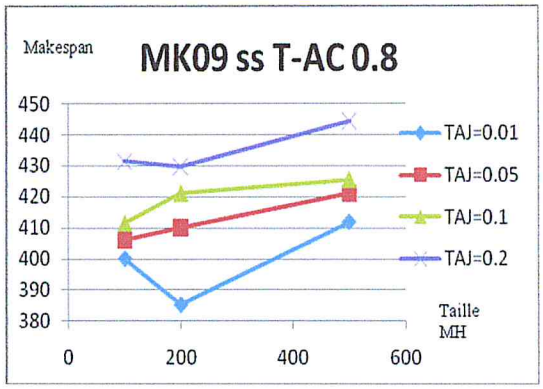
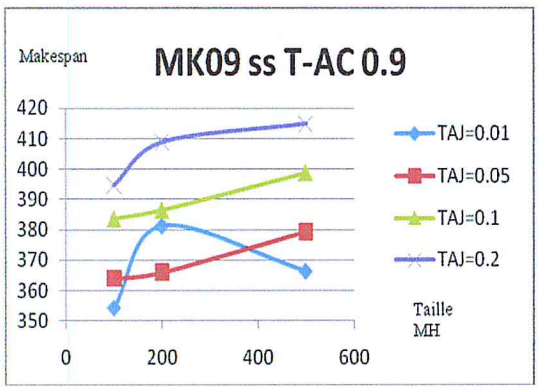


Figure IV.11 : Graphique des résultats de résolution de problème MK04 sans l'opérateur de sélection.

- *Le problème MK09:*







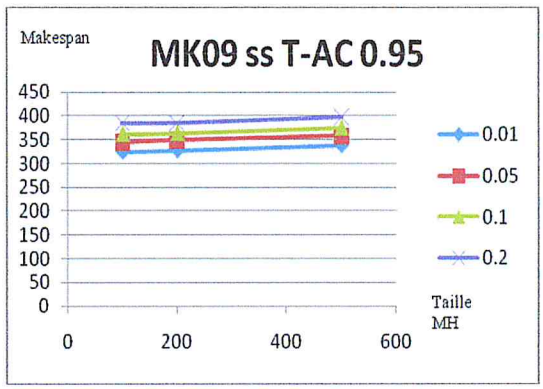
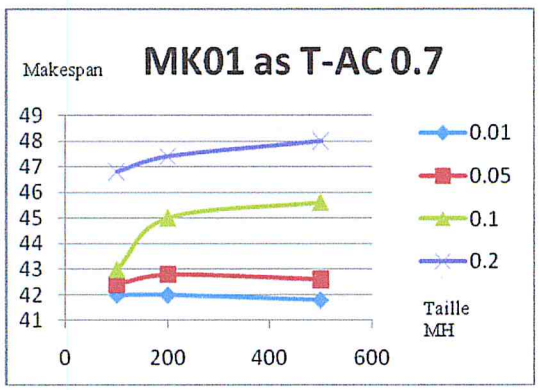
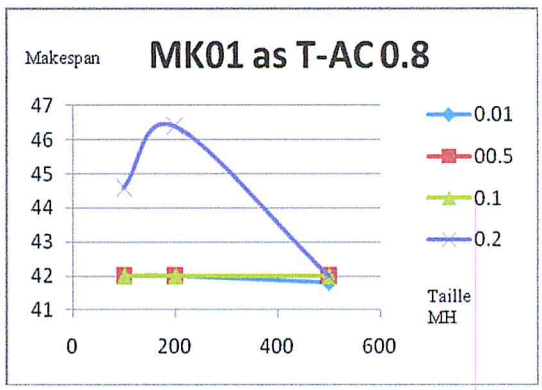


Figure IV.12 : Graphique des résultats de résolution de problème MK09 sans l'opérateur de sélection.

IV.4.2.2. Avec l'opérateur sélection :

- **Le problème MK01 :**





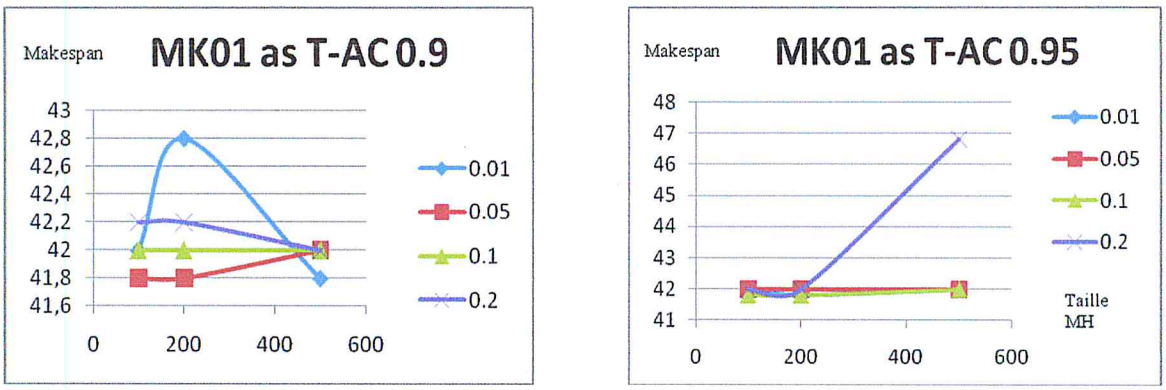


Figure IV.13 : Graphique des résultats de résolution de problème MK01 avec l'opérateur de sélection.

- **Le problème MK04 :**

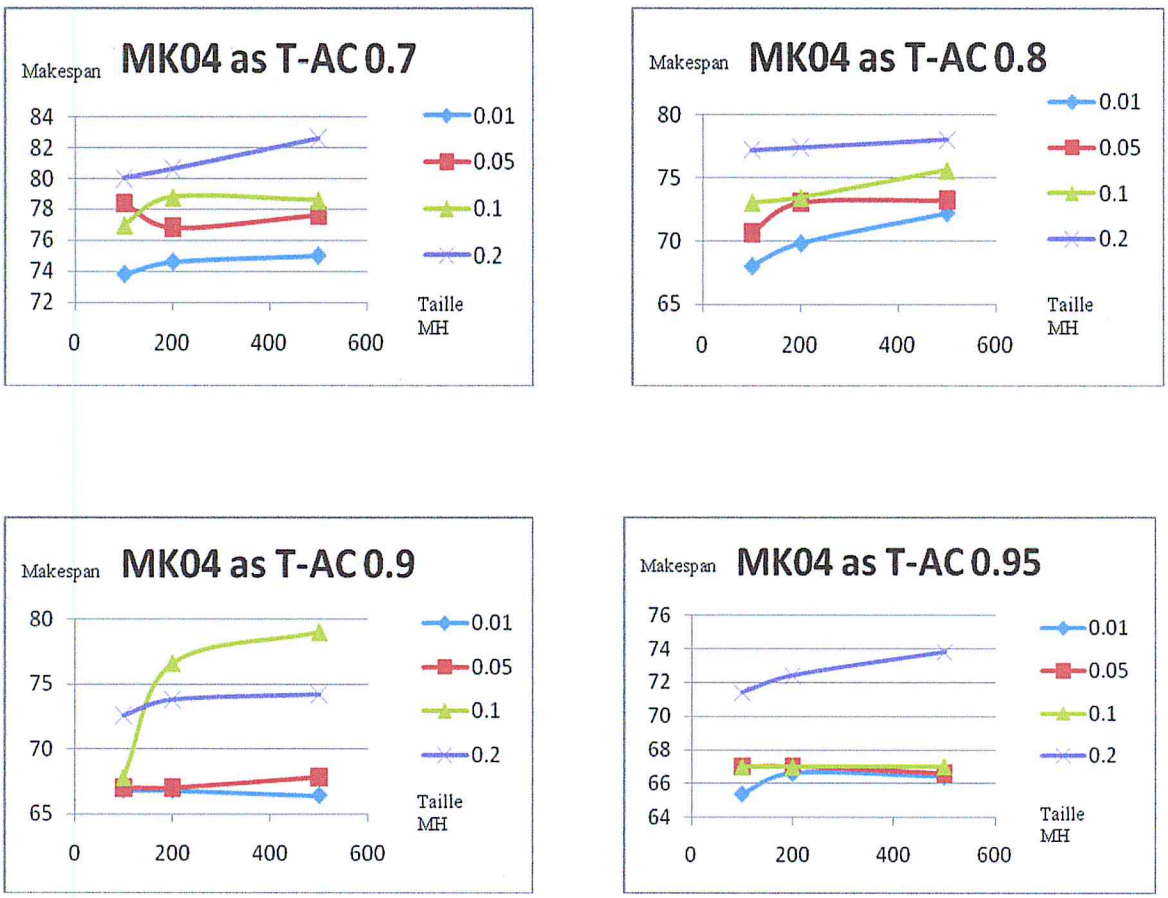


Figure IV.14 : Graphique des résultats de résolution de problème MK04 avec l'opérateur de sélection.

- **Le problème MK09 :**

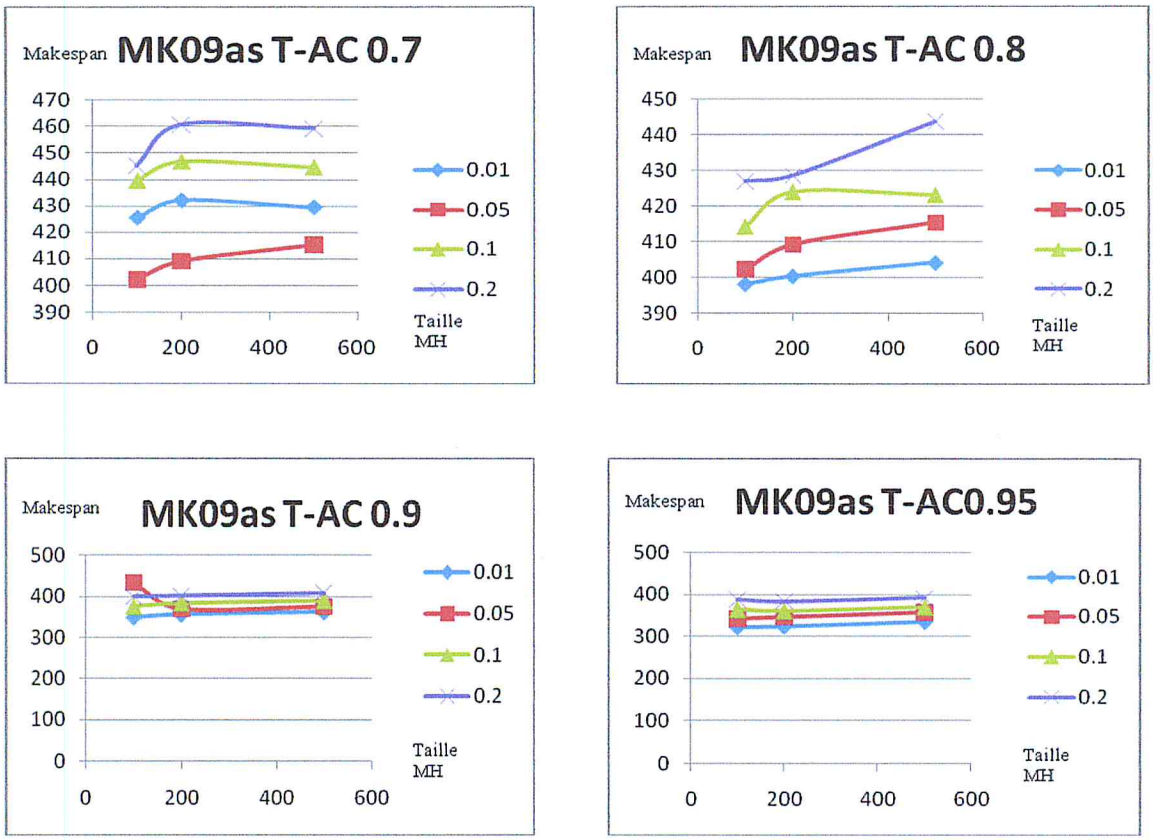


Figure IV.15 : Graphique des résultats de résolution du problème MK09 avec l'opérateur de sélection.

- **Discussion :**

D'après les graphes obtenus on peut apprécier que :

- Une taille de la mémoire harmonique égale à 100 suffit largement comme paramètre de notre algorithme pour les problèmes test. Pour les différents cas l'augmentation de cette valeur dégrade la qualité de la solution.
- L'augmentation du taux d'acceptation permet l'amélioration des résultats.
- Le taux d'ajustement peut être fixé à 0.01. Dans la plupart des cas cette valeur donne de meilleurs résultats.
- Les paramètres de la méthode jouent un rôle important dans le processus de résolution. Ce qui fait qu'un bon choix de ces paramètres est une condition primordiale de la réussite de la recherche.

IV.4.3. Evaluation de l'opérateur de sélection :

Dans notre adaptation de la méthode de recherche d'harmonique à la résolution du job shop flexible nous avons proposé l'intégration de l'opérateur de sélection qui est un opérateur issu des approches évolutionnaires. Afin d'examiner l'effet de l'introduction de cet opérateur nous présentons dans ce qui suit un comparatif des résultats obtenus avec et sans l'utilisation de l'opérateur. Les résultats sont relatifs à la résolution des problèmes : Mk01, Mk04, Mk05, Mk06, Mk07, Mk09. Les paramètres de l'algorithme sont ceux les suivant :

- La taille de la mémoire harmonique : 100.
- Le nombre d'itérations : 500 000.
- Le taux d'acceptation : 0.95.
- Le taux d'ajustement : 0.01.

Les résultats comparatifs sont reproduits dans la figure IV.16.

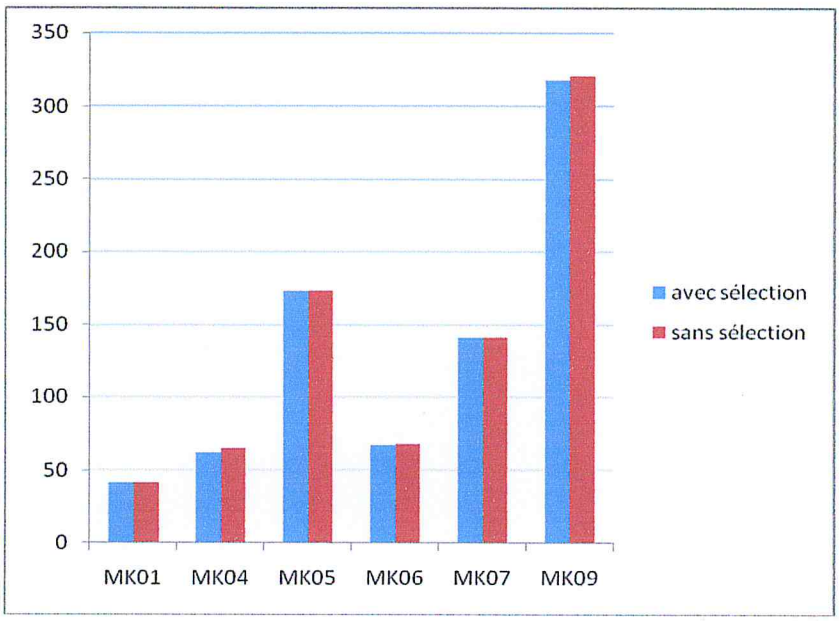


Figure IV.16 : Graphique représente une comparaison entre l'utilisation ou non de l'opérateur sélection selon le fitness.

D'après le graphique on peut déduire que l'opérateur de sélection améliore la solution pour certains problèmes (MK04, MK06 et MK09). Pour les autres problèmes les résultats restent identiques. Cela s'explique en partie par le fait que pour les problèmes MK01, MK05 et MK07 les valeurs obtenues sans l'utilisation de l'opérateur de sélection sont déjà très proches de l'optimum.

Afin aussi d'examiner l'effet de l'introduction de cet opérateur sur le temps de calcul nous présentons dans ce qui suit un comparatif des temps de calcul obtenus avec et sans l'utilisation de l'opérateur. Les résultats sont relatifs à la résolution des problèmes : Mk01, Mk04, et Mk09. pour une taille de la mémoire harmonique égale à 100 et 200 harmonies.

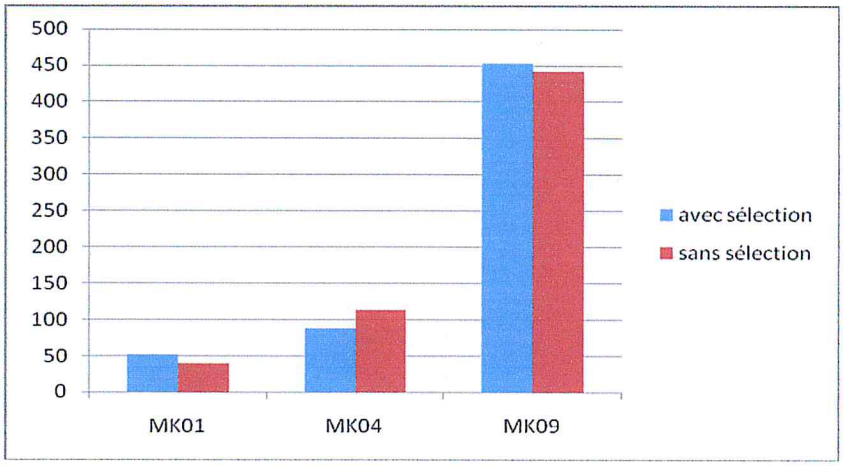


Figure IV.17 : Graphique représentante la différence entre le temps d'exécution avec/sans sélection pour la taille de MH=100.

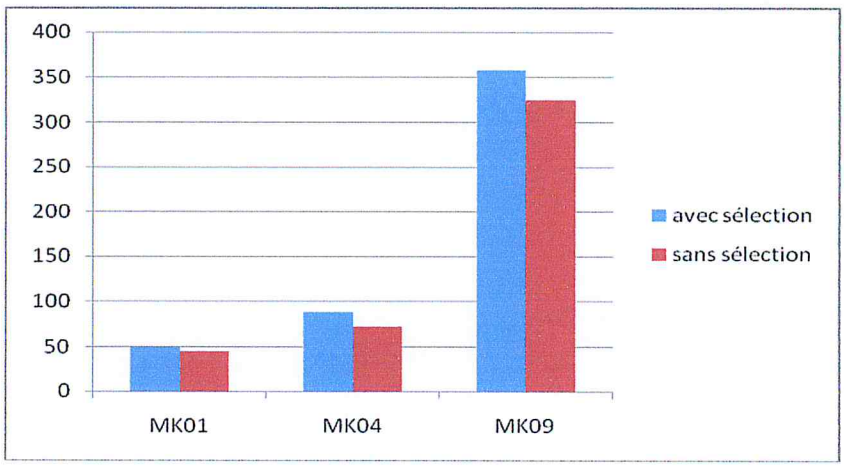


Figure IV.18 : Graphique représente la différence entre le temps d'exécution Avec/sans sélection avec la taille de MH=200.

D'après les graphiques IV.17 et IV.18 l'introduction de l'opérateur de sélection induit une hausse sensible des temps de calcul.

Enfin nous pouvons conclure que l'utilisation de l'opérateur de sélection apporte dans le pire des cas une sensible amélioration de la solution. Même si les temps de calcul sont légèrement augmentés. L'intégration de cet opérateur permet d'améliorer l'algorithme.

IV.4.4. Validation de la méthode de résolution :

IV.4.4.1. Cas Mono-objectif :

Dans ce qui suit nous validons la méthode dans le cadre d'une optimisation mono-objective en considérant le Makespan comme critère d'optimisation.

- *1ere série :*

La première série d'essais est réalisée sur 13 problèmes de la classe **BCdata** (mt10c1, mt10cc, mt10x, mt10xyz, setb4cc, setb4xx, setb4xyz, setb4c9, setb4xy, seti5xyz, seti5xy, seti5xxx, seti5c12). Nous allons comparer les résultats obtenus par la méthode de recherche d'harmoniques avec ceux obtenus en utilisant un algorithme génétique simple (pour le même temps de calcul) et ceux obtenus en utilisant des règles de priorité. Pour chaque problème les tests sont repris trois fois et les résultats reportés sont ceux de la meilleure exécution.

Les résultats relatifs aux règles de priorité ont été obtenus en utilisant le logiciel *LEKIN*, Les règles sont :

LPT: Longest processing time

SPT: Lmallest processing time

FCFS: First comming first served

Les paramètres de notre algorithme sont fixés comme suite :

- Taille de la mémoire harmonie à 100
- Nombre d'itérations 200 000.
- Taux d'acceptation à 0.95.
- Taux d'ajustement à 0.01.

Les résultats sont affichés dans le tableau IV.1 ci-dessous. Ces derniers valident notre implémentation et l'approche. Pour tous les problèmes l'HSA surpasse l'algorithme génétique et les différentes règles de priorité.

Problème	HSA	AG	SPT	LPT	FCFS
Mt10c1	955	974	1247	1168	1234
Mt10cc	924	948	1366	1038	1166
Mt10x	958	964	1413	1173	1220
Mt10xyz	906	915	12091	988	1053
Setb4c9	959	986	1661	1085	1114
Setb4cc	976	965	1504	1088	1137
Setb4xx	960	967	1776	1184	1095
Seti5xyz	1186	1208	2037	1424	1396
Seti5xy	1200	1219	1919	1521	1438
Seti5xxx	1292	1322	1989	1521	1434
Seti5c12	1254	1281	1765	1476	1548
Setb4xyz	936	938	1496	1141	1085
Setb4xy	969	947	1539	1150	1085

Tableau IV.1 : Comparaison ente nos résultats et les résultats obtenus par un algorithme génétique simple et des règles de priorité.

- **2eme série :**

La deuxième série d'essais est réalisée sur les dix problèmes de la classe BRdata (MK01, MK02…MK10) avec les mêmes paramètres sauf le taux d'acceptation qui est égal à 0.97. Nous allons comparer nos résultats avec les approches suivantes :

- *Chen* : algorithme génétique.
- *Jia* : algorithme génétique.
- *Ho-t* : Approche évolutionnaire.
- *AG* : algorithme génétique (PFE 2010).
- *Brandimart* : recherche tabou.
- *Pezz* : algorithme génétique avancé.

Les résultats sont affichés dans le tableau ci-dessous.

PROBs	chen	Jia	Ho-T	AG	Bran.	Pezz	HSA
Mk01	40	40	40	40	42	40	42
Mk02	29	28	29	26	32	26	28
Mk03	204	204	---	204	211	204	204
Mk04	63	61	67	60	81	60	62
Mk05	181	176	176	173	186	173	173
Mk06	60	62	67	62	86	63	67
Mk07	148	145	147	140	157	139	141
Mk08	523	523	523	523	523	523	523
Mk09	308	310	320	307	369	311	318
Mk10	212	216	229	210	296	212	220

Tableau IV.2 : Comparaison ente nos résultats et les résultats obtenus par d'autres chercheurs.

Discussion :

Lorsqu'on compare nos résultats dans le cas Mono-objectif selon les deux séries d'expériences avec les résultats trouvés par d'autres méthodes, on peut déduire qu'on est arrivé à un stade avancé de résultats. La méthode arrive dans la majorité des cas à atteindre des valeurs de la fonction objective proches des meilleures valeurs connues et dans certains cas des valeurs optimales.

IV.4.4.2. Cas multi-objectifs :

Nous abordons dans ce qui suit une validation de la méthode pour le cas multi-objectifs. Pour ce faire nous adoptons une démarche par linéarisation par pondération des objectifs. L'avantage de cette approche qualifiée de naïve de l'optimisation multi-objectif est sa simplicité. Elle est également très efficace algorithmiquement compte tenu du fait qu'elle permet de retrouver la surface de compromis en faisant varier les coefficients de pondération. Ainsi, notre fonction objective (F) regroupe les sous fonctions suivantes : la durée totale *makespan (F1)*, la charge totale des machines (F2) et la machine la plus chargée (F3) :

$$F=\alpha_1*F1+ \alpha_2*F2+ \alpha_3*F3.$$

Nous avons utilisé plusieurs combinaisons des pondérations α_1, α_2, α_3 (Tableau IV.3) :

	α_1	α_2	α_3
Cas 1	0.33	0.33	0.33
Cas 2	0.20	0.40	0.40
Cas 3	0.14	0.43	0.43
Cas 4	0.40	0.20	0.40
Cas 5	0.29	0.29	0.43
Cas 6	0.33	0.50	0.17
Cas 7	0.43	0.14	0.43
Cas 8	0.50	0.33	0.17
Cas 9	0.38	0.38	0.25
Cas 10	0.80	0.05	0.15

Tableau IV.3 : Les différentes valeurs des pondérations utilisé.

- ***1ere série :***

La première série d'essais est réalisée sur 10 problèmes (MK01, MK02, ...MK10) Avec:

- Les pondérations du cas 10.
- La taille de la mémoire harmonique à 100.
- Nombre d'itération à 200000.
- Le taux d'acceptation égale à 0.99.
- Le taux d'ajustement égale à 0.01.

Nous allons comparer nos résultats avec la méthode de recherche SM (Search Method) qui est développé par XING dans le cadre du job shop flexible multi-objectifs **[10]**. Elle a été implémentée sous l'environnement MATLAB avec un PC Pentium 4, processeur de 2.4 G et RAM de 1 Go.

Les résultats sont rapportés dans le tableau ci-dessous.

		F	F 1	F 2	F 3	Temps
MK01	SM	48	42	162	42	4 .78
	HSA	47.35	42	161	36	1.5
MK02	SM	34.35	28	155	28	3.02
	HSA	33.1	27	152	26	0.6
MK03	SM	236.4	204	852	204	26.14
	HSA	236.4	204	852	204	1.37
MK04	SM	82.05	68	352	67	17.74
	HSA	81.1	67	349	67	0.87
MK05	SM	203.25	177	702	177	8.26
	HSA	198.6	173	685	173	1.14
MK06	SM	91.6	75	431	67	18.79
	HSA	85.5	69	414	64	1.35
MK07	SM	178.35	150	717	150	5.68
	HSA	169.45	142	691	142	1
MK08	SM	623.05	523	524	523	67.67
	HSA	623.05	523	2524	523	1.92
MK09	SM	412.35	311	2374	299	77.76
	HSA	410.9	312	2329	299	5.47
MK10	SM	314.2	227	1989	221	122.52
	HSA	303.05	214	2025	204	5.3

Tableau IV.4 : Comparaison des résultats de la méthode HSA avec SM.

- *2eme série :*

Dans cette série nous allons comparer nos résultats avec la méthode de recherche *MOGA* (multi-objective genetic algorithm) qui est développé dans les cadres du job shop flexible multi-objectifs. Elle a été implémentée sous l'environnement C++ avec un PC de 2 GHz pour CPU et RAM 2 Go. **[6]**

La série d'essais est réalisée sur 18 problèmes (01a, 02a, ...18a) Avec:

- La taille de la mémoire harmonique à 100.
- Nombre d'itération à 200000.
- Le taux d'acceptation égale à 0.97.
- Le taux d'ajustement égale à 0.01.
- Les pondérations : nous avons choisir pour chaque problème une pondération qui nous donne un bon résultat ; tel que on a prend (p9 pour 02a, p8 pour 05a, p6 pour 06a, p10 pour 08a, p10 pour 16a, p10 pour 17a).

Les résultats sont affichés dans le tableau ci-dessous.

	MOGA				HSA			
	F 1	F 2	F 3	Temps(S)	F 1	F 2	F 3	Temps(S)
02a	2,289 2,313	11,137 11,137	2,263 2,238	153.4	2276	11137	2241	128
05a	2,292 2,293 2,297 2,315 2,343 2,358 2,376 2,904 2,945 3,056	11,077 11,091 11,054 11,063 11,050 11,038 11,022 10,941 10,941 10,941	2,252 2,242 2,255 2,272 2,298 2,322 2,243 2,620 2,571 2,507	142.4	2250	11071	2244	121
06a	2,250 2,254 2,398 2,437 2,744 2,902 2,967	11,009 10,994 10,973 10,988 10,850 10,847 10,839	2,233 2,223 2,219 2,280 2,448 2,439 2,840	185.6	2233	10909	2194	124
08a	2,187 2,171	16,485 16,485	2,102 2,104	496.0	2283	16485	2213	197
16a	2,447 2,450 2,487 2,492 2,540 2,550 2,568 3,013 3,106	21,602 21,590 21,584 21,576 21,547 21,545 21,540 21,478 21,478	2,354 2,380 2,454 2,417 2,396 2,492 2,428 2,588 2,548	1,291.4	2412.0	21605.0	2320.0	170
17a	2,322 2,322 2,323 2,343 2,480 2,528 2,789 2,808 2,816	21,433 21,362 21,454 21,420 21,344 21,313 21,198 21,200 21,197	2,240 2,280 2,238 2,224 2,285 2,231 2,448 2,303 2,370	1,708.0	2758.0	21493.0	2332.0	175

Tableau IV.5 : Comparaison des résultats de la méthode HSA avec SM.

Discussion :

Selon la première série d'essais, nos résultats sont convainquant en termes de qualité de la solution et de temps de calcul, par rapport à ceux obtenus en utilisant l'algorithme « SM » dans tous les problèmes et pour tous les sous fonctions pas seulement le Makespane. Aussi on remarque que le temps d'exécution est plus réduit.

Dans la deuxième série la méthode arrive à atteindre des valeurs proches dans la majorité des cas, dans les autres cas les résultats obtenus par MOGA sont mieux que ceux obtenus par HSA à cause du nombre d'itérations. Si on augmente ce dernier on peut arriver à de meilleurs résultats. Le tableau ci-dessous représente la liste des résultats obtenus pour le problème 08a par l'approche HSA avec les mêmes paramètres déjà fixés, mais pour un nombre d'itérations égal à 500 000.

	F	F1	F1	F3	Temps
Cas 1	7013.49	2481	16485	2287	395
Cas 2	7941.4	2367	16485	2185	403
Cas 3	8404.08	2575	16485	2221	405
Cas 4	4985.4	2143	16485	2078	423
Cas 5	6309.58	2191	16485	2078	414
Cas 6	9300.15	2134	16485	2079	452
Cas 7	4135.4	2158	16485	2092	421
Cas 8	6862.91	2129	16485	2108	394
Cas 9	7595.36	2137	16485	2076	485
Cas 10	2838.55	2126	16485	2090	439

Tableau IV.6 : Les résultats obtenus pour le problème 08a selon les différentes pondérations.

Discussion des résultats :

D'après les résultats obtenus par les séries d'essais réalisées sur les différents benchmark, nous constatons que la méthode de recherche d'harmonique est très intéressante pour la résolution du problème de Job Shop Flexible en raison de la qualité des solutions retournées.

Dans un premier temps, le choix des paramètres de la méthode joue un rôle important dans le processus de résolution car c'est une condition capitale de la réussite de la recherche.

L'intégration de l'operateur de sélection apporte dans le pire des cas une sensible amélioration de la solution. Même si les temps de calcul sont légèrement augmentés cet opérateur permet d'améliorer l'algorithme.

Aussi la remarque la plus importante à faire ici est que les résultats retournés par la méthode dans le cadre d'une optimisation mono-objective ou d'une optimisation multi-objective sont bonnes dans la majorité des cas mais il reste que la méthode est plus efficace à utiliser dans le cas multi-objectif.

IV.5. Conclusion :

Dans ce dernier chapitre nous avons présenté les résultats de différentes expérimentations réalisées sur un ensemble de problèmes tests. L'objectif est d'un coté d'explorer les performances des stratégies qu'on a utilisées, et de l'autre de valider notre implémentation de la méthode méta-heuristique. Les résultats restent dans leur globalité concluant.

Conclusion Générale

Conclusion Générale

Dans le cadre de notre projet de fin d'études, notre travail a consisté à adapter une nouvelle méta-heuristique dite de recherche d'harmonique aux problèmes d'ordonnancement industriels de type job shop flexible.

Dans un premier temps, nous avons pu définir dans la première partie « étude préliminaire » les concepts de bases de l'ordonnancement avec ses différents types d'ateliers ainsi que les différentes approches méta-heuristiques développées pour la résolution de ce type de problème industriel, notamment l'approche par recherche d'harmonique. D'autre part, dans la deuxième partie « implémentation » nous avons pu exploiter le principe de fonctionnement de cette dernière pour la résolution du job shop flexible. Ainsi après implémentation nous avons comparé ces résultats à ceux obtenus par l'utilisation d'autres méthodes pour pouvoir évaluer d'une part notre implémentation et d'autres parts l'efficacité de la méthode. Les résultats obtenus sont dans leur globalité convainquant et encourageant.

En fin, nous espérons que ce modeste travail servira comme base et sera bénéfique pour d'autres étudiants qui voudraient continuer dans ce domaine. Qu'il soit aussi complété par d'autres études dédiées à la résolution des problèmes NP-difficile en général et la résolution des problèmes d'ordonnancement en particulier.

Annexe

Annexe I :

« La lecture d'un Benchmark »

```
3  4
2  2  2  5  3  3  2  1  8  2  2
1  2  1  7  4  10
3  3  2  2  3  10  4  3  4  1  3  2  7  3  5  4  4  2  1  12  3  4
```

- La première ligne : (3) indique le nombre de job.

  (4) indique le nombre de machine.

- la deuxième ligne : (2) indique le nombre des opérations pour la 1ère tache (job_1)

  Pour la 1er tache (job1) contient deux opérations tel que :

  - Deux possibilités pour la 1cr opération : soit machine 2 avec le temps d'exécution 5

    Soit machine 3 avec le temps d'exécution 3

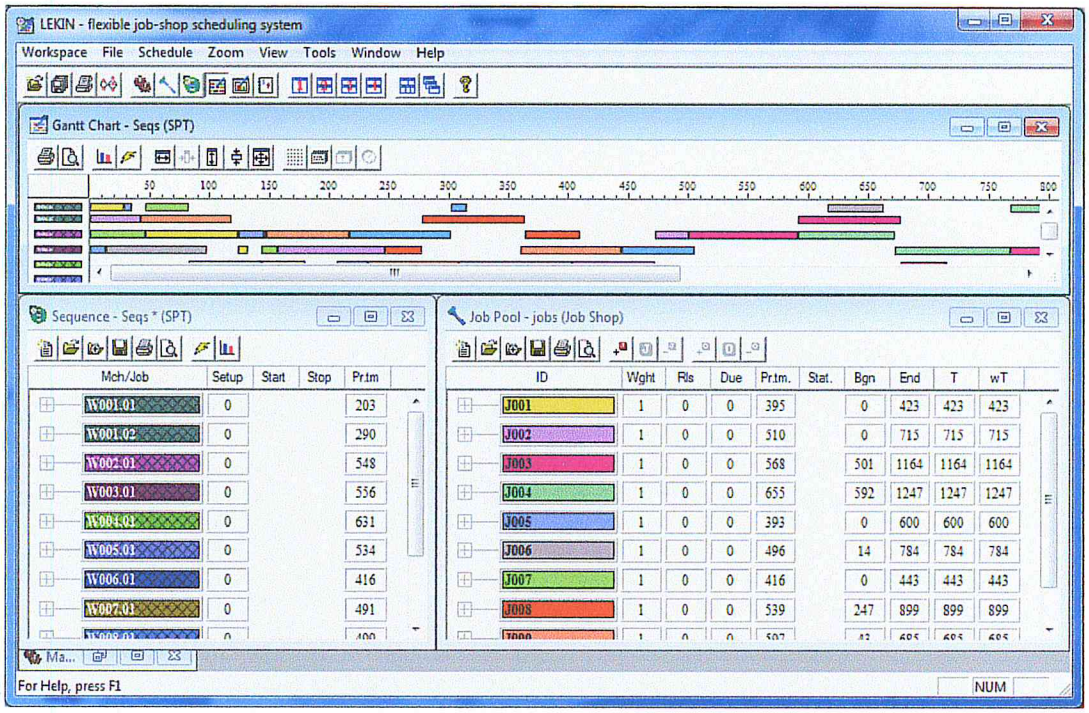
  - Deux possibilités pour la 2eme opération : soit machine 1 avec le temps 8

    Soit machine 2 avec le temps 2

Annexe II :

« LEKIN »

Lekin est un système d'ordonnancement développé à Stern of Business université de New York. De grandes parties du système ont été conçus et codés par des étudiants de l'université Columbia. Lekin a été créé comme un outil pédagogique dont le but principal d'initier les élèves à la théorie d'ordonnancement et ses applications. Outre cela, l'extensibilité des systèmes permet (et encourage) à l'utiliser dans le développement d'algorithmes.



Annexe III :

Les tableaux ci-dessous représentent la liste des résultats obtenus dans le cadre du job shop flexible multi-objectifs avec les problèmes (01a, 02a, 03a, 04a, 05a, 06a, 07a, 09a, 10a, 11a, 12a, 13a, 14a, 15a, 16a, 17a, 18a) par l'approche HSA avec les 10 cas des pondérations (voir Tableau IV.3) et les paramètres :

- La taille de la mémoire harmonique à 100.
- Nombre d'itération à 200000.
- Le taux d'acceptation égale à 0.97.
- Le taux d'ajustement égale à 0.01

Problème 01a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	5436.05	2637	11137	2602	60
Cas 2	5995.4	2693	11137	2505	117
Cas 3	6246.58	2718	11137	2505	123
Cas 4	4305.8	2691	11137	2505	76
Cas 5	5091.91	2707	11137	2505	112
Cas 6	6876.5	**2616**	11137	2616	83
Cas 7	3787 .44	2677	11137	2505	114
Cas 8	5447.6	2655	11137	2617	85
Cas 9	5871.39	2666	11137	2505	108
Cas 10	3083.5	2662	11137	2647	105

Problème 02a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	5186.61	2314	11137	2266	119
Cas 2	5817.6	2332	11137	2241	85
Cas 3	6096.92	2466	11137	2239	105
Cas 4	4047.4	2303	11137	2247	137
Cas 5	4864.99	2313	11137	2243	134
Cas 6	6706.64	2296	11137	2238	125
Cas 7	3523.42	2323	11137	2245	133
Cas 8	5200.54	2286	11137	2249	119
Cas 9	5657.19	**2276**	11137	2241	128
Cas 10	2740.4	2301	11137	2285	106

Problème 03a :

	F	*F1*	*F1*	*F3*	*Temps*
Cas 1	5230.5	2431	11137	2282	119
Cas 2	5902.8	2569	11137	2272	132
Cas 3	6149.8	2767	11137	2264	108
Cas 4	4161	2504	11137	2330	121
Cas 5	4938.87	2550	11137	2255	123
Cas 6	6723.67	2327	11137	2278	115
Cas 7	3622.31	2513	11137	2285	123
Cas 8	5216.09	2312	11137	2264	121
Cas 9	5736.87	2462	11137	2277	125
Cas 10	2740.45	**2305**	11137	2264	112

Problème 04a :

	F	*F1*	*F1*	*F3*	*Temps*
Cas 1	5360.19	2662	11078	2503	114
Cas 2	5961.59	2646	11078	2503	141
Cas 3	6213.87	2684	11074	2503	110
Cas 4	4286.8	2675	11078	2503	124
Cas 5	5063.5	2671	11078	2503	109
Cas 6	6852.73	2645	11071	2614	115
Cas 7	3783.05	2688	11078	2503	106
Cas 8	5409.25	2623	11077	2602	111
Cas 9	5841.55	**2606**	11079	2565	125
Cas 10	2641	2641	11092	2615	119

Problème 05a :

	F	*F1*	*F1*	*F3*	*Temps*
Cas 1	5124.9	**2278**	11021	2231	116
Cas 2	5751.2	2328	10994	2220	80
Cas 3	6002.46	2378	10967	2218	103
Cas 4	4021.2	2300	11038	2234	121
Cas 5	4832.9	2338	11031	2223	110
Cas 6	6639.35	2274	11020	2229	118
Cas 7	3483.01	2280	11048	2223	120
Cas 8	5159.91	**2250**	11071	2244	121
Cas 9	5604.19	2277	11011	2219	125
Cas 10	2281	2281	11075	2268	117

Problème 06a :

	F	*F1*	*F1*	*F3*	*Temps*
Cas 1	5100.81	2310	10938	2209	120
Cas 2	5716.6	2339	10923	2199	105
Cas 3	5966.57	2438	10877	2205	115
Cas 4	4054.2	2375	11017	2252	129
Cas 5	4846.16	2427	10964	2239	117
Cas 6	6564.37	2233	10909	2194	124
Cas 7	3513.09	2340	11011	2245	121
Cas 8	5141.99	2281	10979	2226	122
Cas 9	5559.59	2241	10927	2223	130
Cas 10	2323	2323	11040	2285	119

Problème 07a :

	F	*F1*	*F1*	*F3*	*Temps*
Cas 1	7010.52	2572	16485	2187.	180
Cas 2	7981.4	2563	16485	2187	184
Cas 3	8385.96	2550	16485	2187	196
Cas 4	5197.4	2564	16485	2187	191
Cas 5	6469.83	2582	16485	2187	187
Cas 6	9481.48	2582	16485	2276	195
Cas 7	4377.06	2625	16485	2187	183
Cas 8	7111.47	2569	16485	2276	188
Cas 9	7806.1	2560	16485	2276	201
Cas 10	2560	2560	16485	2403	197

Problème 09a :

	F	*F1*	*F1*	*F3*	*Temps*
Cas 1	7090.05	2804	16485	2196	148
Cas 2	8023.2	2788	16485	2179	200
Cas 3	8428.05	2995	16485	2140	192
Cas 4	5278.2	2709	16485	2244	211
Cas 5	6541.38	2805	16485	2203	199
Cas 6	9503.69	2673	16485	2230	206
Cas 7	4379.21	2618	16485	2199	203
Cas 8	7136.41	2625	16485	2258	203
Cas 9	7849.32	2679	16485	2268	213
Cas 10	2567	2567	16485	2203	219

Problème 10a:

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	7013.49	2549	16513	2191	102
Cas 2	7996	2596	16514	2178	103
Cas 3	8404.36	2620	16514	2178	117
Cas 4	5211.0	2587	16516	2178	116
Cas 5	6461.03	2537	16513	2178	124
Cas 6	9482.94	2596	16512	2178	125
Cas 7	4361.92	2590	16512	2178	120
Cas 8	7105.07	2561	16512	2192	110
Cas 9	7814.25	2596	16519	2263	117
Cas 10	3236.6	2575	16524	2336	118

Problème 11a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	6876.87	2396	16346	2124	428
Cas 2	7862.8	2464	16312	2113	156
Cas 3	8276.8	2698	16283	2087	122
Cas 4	5185	2500	16444	2241	115
Cas 5	6381.83	2452	16359	2155	126
Cas 6	9276	2327	16273	2188	110
Cas 7	4410	2673.0	16370.0	2245	3105
Cas 8	6981.6	2425	16381.0	2138.0	114
Cas 9	7650.16	2409.0	16323.0	2128.0	129
Cas 10	3045.05	2368.0	16470.0	2181.0	124

Problème 12a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	6900.3	2640.0	16119.0	2151.0	234
Cas 2	7845.0	2781.0	16149.0	2073.0	110
Cas 3	10612.6	2585.0	21610.0	2229.0	185
Cas 4	5238.4	2691.0	16296.0	2257.0	124
Cas 5	6457.25	2790.0	16241.0	2182.0	121
Cas 6	9216.28	2492.0	16031.0	2226.0	111
Cas 7	7028.18	2587.0	16194.0	2298.0	116
Cas 8	7711.24	2613.0	16235.0	2196.0	124
Cas 9	3287.7	2645.0	16399.0	2345.0	122
Cas 10	2665.0	2665.0	16403.0	2272.0	120

Problème 13a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	8691.2	2497.0	21610.0	2230.0	1149
Cas 2	10045.1	2542.0	21610.0	2232.0	185
Cas 3	10618.74	2656.0	21610.0	2220.0	167
Cas 4	6221.2	2505.0	21610.0	2243.0	180
Cas 5	7961.5	2509.0	21610.0	2249.0	198
Cas 6	12013.39	2479.0	21610.0	2296.0	165
Cas 7	5104.88	2554.0	21610.0	2282.0	160
Cas 8	8763.83	2480.0	21610.0	2309.0	160
Cas 9	9747.0	2546.0	21610.0	2271.0	178
Cas 10	3409.75	2473.0	21610.0	2339.0	177

Problème 14a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	8819.25	2759.0	21610.0	2356.0	521
Cas 2	10181.8	3037.0	21610.0	2326.0	187
Cas 3	10730.16	3286.0	21610.0	2274.0	198
Cas 4	6448.8	2897.0	21610.0	2420.0	202
Cas 5	8090.9	2949.0	21610.0	2253.0	163
Cas 6	12154.04	2864.0	21610.0	2376.0	207
Cas 7	5290.6	2891.0	21610.0	2377.0	170
Cas 8	8978.75	2867.0	21610.0	2435.0	174
Cas 9	9882.8	2814.0	21610.0	2407.0	210
Cas 10	3627.9	2751.0	21610.0	2311.0	168

Problème 15a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	8926.17	3095.0	21610.0	2344.0	171
Cas 2	10217.59	3274.0	21610.0	2297.0	215
Cas 3	10747.4	3345.0	21610.0	2295.0	173
Cas 4	6447.2	2975.0	21610.0	2338.0	195
Cas 5	8180.4	3157.0	21610.0	2321.0	208
Cas 6	12242.34	2515.0	3060.0	21610.0	201
Cas 7	5389.97	3098.0	21610.0	2401.0	168
Cas 8	9037.44	3015.0	21610.0	2345.0	178
Cas 9	9954.3	3075.0	21610.0	2296.0	305
Cas 10	3884.89	3073.0	21610.0	2307.0	175

Problème 16a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	8684.28	2503.0	21565.0	2248.0	1346
Cas 2	10023.4	2565.0	21546.0	2230.0	185
Cas 3	10595.32	2630.0	21543.0	2241.0	169
Cas 4	6202.6	2464.0	21573.0	2256.0	191
Cas 5	7957.58	2503.0	21577.0	2266.0	199
Cas 6	11988.72	2496.0	21544.0	2312.0	190
Cas 7	5071.89	2499.0	21574.0	2272.0	188
Cas 8	8724.76	2435.0	21550.0	2328.0	183
Cas 9	9718.47	2542.0	21527.0	2289.0	170
Cas 10	3357.85	2412.0	21605.0	2320.0	170

Problème 17a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	8692.2	2770.0	21225.0	2345.0	1224
Cas 2	9956.4	2892.0	21186.0	2259.0	200
Cas 3	10481.19	3028.0	21157.0	2232.0	176
Cas 4	6364.0	2891.0	21388.0	2325.0	182
Cas 5	8024.03	2929.0	21309.0	2314.0	191
Cas 6	11888.5	2757.0	21156.0	2357.0	164
Cas 7	5264.48	2924.0	21460.0	2332.0	187
Cas 8	8781.81	2758.0	21251.0	2294.0	193
Cas 9	9726.86	2760.0	21262.0	2394.0	176
Cas 10	3630.85	2758.0	21493.0	2332.0	175

Problème 18a :

	F	*F1*	*F2*	*F3*	*Temps*
Cas 1	8734.11	2932.0	21187.0	2348.0	603
Cas 2	9985.6	3026.0	21240.0	2211.0	185
Cas 3	10467.51	3176.0	21065.0	2244.0	200
Cas 4	6451.6	3039.0	21464.0	2358.0	216
Cas 5	8072.49	3097.0	21357.0	2281.0	199
Cas 6	11879.88	2787.0	21159.0	2239.0	195
Cas 7	5300.70	3031.0	21430.0	2319.0	192
Cas 8	8892.53	2918.0	21341.0	2300.0	179
Cas 9	9755.66	2859.0	21298.0	2304.0	182
Cas 10	3811.35	2964.0	21474.0	2443.0	184

Annexe IV :

Le tableau ci-dessous représente la série d'essais réaliser sur 10 problèmes (MK01, MK02, ...MK10) dans le cadre du job shop flexible multi-objectifs avec les 10 cas de pondérations et les paramètres suivants :

- La taille de la mémoire harmonique à 100.
- Nombre d'itération à 200000.
- Le taux d'acceptation égale à 0.99.
- Le taux d'ajustement égale à 0.01.

		Cas 1	Cas 2	Cas 3	Cas 4	Cas 5	Cas 6	Cas 7	Cas 8	Cas 9	Cas10
MK01	F	79.2	87.6	89.86	64.2	75.21	99.5	56.06	81.27	85.62	47.35
	F1	42	50	46	41	41	46	40	42	44	42
	F2	160	154	154	167	162	153	167	161	155	167
	F3	38	40	40	36	38	46	36	42	40	36
MK02	F	66.66	74.6	77.88	51.2	61.77	86	43.94	66.62	72.49	33.1
	F1	29	29	28	27	28	31	27	29	28	27
	F2	144	144	145	148	145	141	148	143	145	152
	F3	29	28	27	27	27	31	27	29	27	26
MK03	F	415.46	463.20	482.48	333.6	393.96	528	294.72	417.84	452.28	236.4
	F1	205	204	209	204	204	204	204	204	204	204
	F2	850	852	850	852	852	852	852	852	852	852
	F3	204	204	204	204	204	204	204	204	204	204
MK04	F	158.4	177.6	185.82	121.8	147.73	205.99	103.85	158.4	173.06	81.1
	F1	68	68	68	66	66	69	62	66	66	67
	F2	346	344	344	353	350	344	364	346	346	349
	F3	66	66	66	62	63	66	61	66	66	67
MK05	F	339.9	377.6	392.87	275.4	323.21	428.5	244.68	341.96	369.16	198.6
	F1	173	174	174	173	173	174	173	173	174	173
	F2	684	683	683	685	685	683	685	685	683	685
	F3	173	174	174	173	173	174	173	173	174	173
MK06	F	168.96	189.2	197.07	131.4	158.16	215.32	110.76	169.91	185.18	85.5
	F1	79	82	90	74	79	96	72	79	82	69
	F2	361	362	350	387	370	336	398	355	354	414
	F3	72	70	79	61	65	92	56	78	78	64
MK07	F	317.46	355.6	371.9	250	299.71	409.5	218.56	318.9	346.84	169.45
	F1	144	144	144	144	144	157	141	144	145.	142
	F2	674	673	674	674	676	662	695	674	673	691
	F3	144	144	144	144	144	157	141	144	144	142
MK08	F	1177.1	1322	1381.8	923	1107.7	1521.5	803.14	1182.3	1287.3	623.05
	F1	524	524	524	524	524	524	523	524	524	523
	F2	2519	2519	2519	2519	2519	2519	2524	2519	2519	2524
	F3	524	524	524	524	524	524	523	524	524	523
MK09	F	960.63	1098.2	1154.4	705.8	887.06	1291.9	589.15	964.17	1065.8	410.9
	F1	334	351	362	325	334	358	320	327	336	312
	F2	2267	2258	2244	2281	2280	2232	2307	2264	2261	2329
	F3	310	312	323	299	300	340	299	315	316	299
MK10	F	773.52	887	941.83	561.6	709.28	1065.2	460.56	782.59	861.72	306.8
	F1	238	255	259	232	237	255	225	236	241	214
	F2	1886	1870	1880	1934	1890	1885	1969	1898	1878	2091
	F3	220	220	226	205	215	227	205	225	226	207

Bibliographie

Bibliographie

[1] FERRAH Djahida ,BABA Kahina, «Approche évolutionnaire pour la résolution du problème d'ordonnancement de type Job Shop Flexible Dynamique », 2009.

[2] LATRECHE Faiz , « les problemes d'ordonnancement a cheminement unique - flow shop- et multiple - job shop - 2004

[3] HELA Boukef ben othman « l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques Optimisation par algorithmes génétiques et essaims particulaires », 2011.

[4] Ben Hmida A., Huguet M.-J., Lopez P., Haouari M., "Adaptation of Discrepancy-based methods for solving Hybrid Flow Shop Problems", Proceedings IEEE-ICSSSM'06, 1120-1125, 2006.

[5] Layeb abbdesslem « introduction aux méta-heuristique »,2009.

[6] Xiaojuan Wang « A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem », 2009.

[7] Khoukou faiza « application des algorithmes génétique sur le repliement de protéine », 2008.

[8] L.abdelhakem-koridak « optimisation d'un dispatching économique et environnemental d'energie electrique par l'algorithme harmony search», 2006

[9] BARRETTE Mathieu « méthode de comparaison statistique des performances d'algorithme évolutionnaires », 2008.

[10] Li- xing « an officient search methode for multi-objective flexible job shop scheduling problems», 2009.