République Algérienne Démocratique et Populaire
Ministère De l'Enseignement Supérieur Et De la Recherche Scientifique

Université Saad Dahlab De Blida

N° D'ordre :.....

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

Salhi Walid Hamdi Hamadi

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Sujet :

Conception et Expérimentation d'un algorithme à Estimation de Distribution avec Décomposition pour le pilotage multi-objectif des systèmes de production

Soutenu le :

M.	Tenbalin	Président
M.	Guessoum	Examinateur
M.	Sidemoun	Examinateur
M.	Chikhi Nacim	Promoteur
M.	Gaham Mehdi	Encadreur

Promotion 2014 / 2015

MA-004-279-1

Résumé

Nous proposons dans ce travail, la conception et la mise en œuvre d'une approche évolutionnaire multi-objectifs pour la résolution de problèmes complexes de pilotage industriel.

L'approche proposée est basée sur l'hybridation de deux algorithmes : EDA (Estimation of Distribution Algorithm) et MOEA/D (Multi-Objective Evolutionary Algorithm with Decomposition). Cette hybridation aura pour objectif le développement d'un algorithme de calcul, qui comme EDA, ne nécessitera pas d'importantes ressources de calcul puisqu'il utilisera une représentation probabiliste de la population, et qui peut comme pour MOEA/D être implémenté en parallèle dans le cadre de l'optimisation multi-objectifs.

Abstract

In this work, we design and implement a new multi-objective evolutionary approach to complex problems solving in the context of industrial management.

The proposed approach is based on a hybridization of two algorithms: EDA (Estimation of Distribution Algorithm) and MOEA/D (Multi-Objective Evolutionary Algorithm with Decomposition). The aim of this hybridization is to design an algorithm which is efficient like EDA, and which can be implemented in a parallel way like MOEA/D as part of the multi-objective optimization.

ملخص

نقترح في هذا العمل تصميم وتنفيذ نهج تطوري متعدد الأهداف في حل المشاكل المعقدة للإدارة الصناعية

وسوف يستند هذا النهج على التهجين اثنين من الخوارزمية (EDA & MOEA\D) وسيهدف هذا التهجين لتطوير خوارزمية تتطلب موارد الحوسبة لاستعمالها EDA بسبب سهولة تطبيقه والتمثيل الاحتمالي للمترشحين MOEA/D بسبب التنفيذ بالتوازن من اجل تحسين متعدد الأهداف.

Remerciements

Pour commencer, on remercie Dieu, le Tout Puissant, le Miséricordieux, qui nous a donné l'opportunité de mener à bien ce travail.

Nous adressons toute notre gratitude à Monsieur le directeur ***Brahim Bouzouina*** *pour nous avoir accueilli au sein du Centre de Développement des Technologies Avancées (CDTA) basé à Baba Hassen Alger.*

Nos remerciements les plus sincères vont à notre encadreur Monsieur ***Gaham Mehdi****, pour les appuis scientifiques qu'il nous a procurés, sa disponibilité et les moyens techniques qu'il nous a fournis ainsi que l'énorme soutien dans nous avions besoin tout au long de notre stage.*

Nous remercions également très chaleureusement notre promoteur, Monsieur ***Chikhi Nacim Fateh****, qui nous a soutenus tout au long de notre stage.*

On aimerait exprimer notre gratitude et nos sincères remerciements à tous les membres du jury et à toute l'équipe pédagogique et administrative du master IL du département d'informatique et de la faculté des sciences.

On réserve une pensée spéciale à tous les enseignants de l'université de Saad Dahlab et spécialement ceux du département d'informatique, qui ont su nous donner une formation didactique et appréciable durant tout notre cursus. Ainsi qu'à tous ceux qui nous ont accordé de leur temps précieux.

Dédicace

Je dédie ce travail :

- *A mes très chers parents qui m'ont toujours soutenue et encouragé à tout moment, que dieu les protèges.*
- *A mes grands-parents, mon oncle, mes tentes et l'ensemble de mes cousins et cousines.*
- *A mes très chers frères.*
- *A mon très cher frère et ami Mohamed pour tous l'aide et le soutien dans il m'a apporté.*
- *A mon binôme « Hamdi ».*
- *A tous mes amis d'enfances, mes amies d'études, mes camarades de classe ainsi qu'à toute la promotion 2010 informatique pour tous les bons moments passé ensemble.*
- *A tous mes enseignants depuis l'école primaire jusqu'à l'université pour le savoir qu'ils m'ont transmis, je vous dédier ce modeste travail en reconnaissance des efforts fournis.*
- *A toutes les personnes proches de moi, avec qui j'ai partagé des moments de bonheur je vous dédie ce travail.*

Salhi Walid

Dédicace

Merci Allah (mon dieu) de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel et de dire " Ya Kayoum "

Je dédie ce modeste travail à celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère, A mon père, qui a été mon ombre durant toutes les années des études, et qui a veillé tout au long de ma vie à m'encourager, à me donner l'aide et à me protéger. Que dieu les gardes et les protège.

Je dédie ce mémoire à toute ma famille, mon père hamdi Abderrahmane, ma mère tahraoui naima, mon frères amine, mes sœurs romaisa et bouchra, qu'ils ont consentis pour me permettre de suivre mes études dans les meilleures conditions possibles et n'avoir jamais cessez de m'encourager tout au long de mes années d'étude.

Je le dédie particulièrement à ma grand-mère, Je ne saurai terminer sans citer mes amis.

On dit souvent que le trajet est aussi important que la destination. Les Cinque années de maîtrise effet, ne s'est pas réalisé sans défis et sans soulever de nombreuses questions pour lesquelles les réponses nécessitent de longues heures de travail.

*Je remercie infiniment le professeur **Ms Nassim Chikhi**, mon promoteur **Ms Gaham**, pour l'effort fourni, les conseils prodigués, sa patience et sa persévérance dans le suivi.*

Je remercie aussi tous ceux qui m'ont aidé à réaliser ce travail, particulièrement Imane Boussaid.

Enfin je remercie tous les professeurs de département informatique de l'université de Blida ainsi que, tous les étudiants

Enfin je le dédie à tous mes amis que je n'ai pas cités et à tous ceux qui me connaissent,

Qu'ils trouvent à travers ce travail ma sincère reconnaissance.

H.HAMADI

SOMMAIRE

Liste des figures

Liste des tableaux

Introduction générale

De nos jours, les entreprises doivent être compétitives et ont besoin pour cela d'outils performants pour gérer la complexité de leur organisation. La gestion de production appartient à ces outils permettant le contrôle et la planification du processus de production. Parmi les fonctions de la gestion de production, la résolution du problème d'ordonnancement. Les problèmes d'ordonnancement sont très variés. On peut les rencontrer dans de très nombreux domaines : les systèmes industriels de production (activités des ateliers en gestion de production et problèmes de logistique), les systèmes informatiques (les tâches sont les programmes et les ressources sont les processeurs, la mémoire...), les systèmes administratifs (gestion du personnel, emplois du temps,…), les systèmes de transport, la construction, … etc…

C'est pour cette raison qu'ils ont fait et continuent de faire l'objet de nombreux travaux de recherche.

Dans un système de production, le problème d'ordonnancement consiste à organiser dans le temps l'exécution d'opérations interdépendantes à l'aide de ressources disponibles en quantités limitées pour réaliser un plan de production.

Les problèmes d'ordonnancement d'ateliers constituent pour les entreprises une des difficultés importantes de leur système de gestion et de pilotage de la production. Dans ce type d'ordonnancement, les ressources sont généralement des machines et chaque travail à ordonnancer concerne un produit à fabriquer en respectant les gammes (les contraintes) de fabrication.

Dans le domaine industriel, parmi les problèmes d'ordonnancement les plus difficiles et les plus étudiés nous avons les problèmes d'ordonnancement de type job shop. L'objectif consiste à programmer la réalisation des produits de à manière à optimiser la production en respectant les contraintes.

La résolution de ce problème de manière optimale est généralement impossible avec les méthodes exactes ; ces derniers requièrent un effort calculatoire qui croît exponentiellement avec la taille du problème, alors que les méthodes approchées ont été proposées pour résoudre ce problème en temps raisonnable et en réduisant l'espace de mémoire pour le calcul. On trouve parmi ces méthodes les métaheuristiques.

L'avantage des métaheuristiques se résume dans leur capacité à résoudre tous les problèmes avec un bon compromis entre le temps de recherche et la qualité de solution. Grâce à ces méthodes on peut proposer des solutions approchées pour des problèmes difficiles et de plus grande taille qu'il était impossible de résoudre avec des méthodes exactes. Parmi ces méthodes il y a les algorithmes évolutionnaires représentés notamment par l'approche EDA (Estimation Distribution Algorithme).

Dans ce travail, nous voulons montrer qu'il est possible de résoudre des problèmes complexes avec une approche évolutionnaire multi objectifs en mettant en œuvre une hybridation des deux algorithmes EDA et MOEA/D pour le développement d'un algorithme de calcul qui ne nécessitant pas d'importantes ressources pour le calcul et l'implémentation en parallèle.

Ce mémoire est structuré en quatre chapitres qui permettent un cadrage progressif du sujet.

Dans le premier chapitre, un contexte général de l'ordonnancement et des problèmes d'ordonnancement est décrit par la présentation des points essentiels : définition, classification, et organisation de l'atelier concerné (job shop flexible).

Dans le deuxième chapitre, nous présentons les méthodes de résolution des problèmes d'ordonnancement en accordant une attention particulière aux métaheuristiques et les algorithmes évolutionnaires. Nous faisons ensuite un état de l'art sur les approches de résolution pour le problème du job shop flexible.

Le troisième chapitre est consacré à l'adaptation des algorithmes EDA et MOEA/D pour résoudre le problème de job shop flexible.

Dans le quatrième chapitre, des expérimentations sont effectuées sur des benchmarks comme échantillon pour tester et valider l'approche proposée.

Ce mémoire se termine par une conclusion et un ensemble de perspectives pour de futurs travaux.

Chapitre 1

En raison de la croissance de la concurrence, les entreprises poussent fortement à vouloir mettre en place les meilleurs processus organisationnels visant à optimiser leur production, tout en respectant les délais de livraison et à fabriquer des produits de qualité à moindres coûts.

Les performances d'une entreprise dépendent donc de l'organisation adoptée et de l'exploitation optimale de ses ressources.

Ainsi, l'amélioration des performances de l'entreprise est tributaire des méthodes d'organisation et d'exploitation des ressources dont elle dispose. C'est pourquoi, l'utilisation d'outils adéquats de gestion de la production est aujourd'hui une nécessité de plus en plus préoccupante pour les entreprises.

Dans ce chapitre introductif nous commençons par expliquer les notions relatives aux systèmes de production et à la gestion de production, ensuite, nous détaillons la notion d'ordonnancement en mettant l'accent sur le problème d'ordonnancement.

La résolution des problèmes d'ordonnancement est une branche de la recherche opérationnelle qui consiste à trouver une séquence optimale pour l'exécution de n tâches sur m ressources afin de minimiser une fonction objectif. Il s'agit également de calculer les dates de début et de fin d'exécution des tâches.

Dans ce chapitre, un rappel de quelques notions de base relatives aux problèmes d'ordonnancement est d'abord introduit, ensuite nous présentons les modèles classiques d'ateliers les plus étudiés dans la littérature. La dernière partie de ce chapitre est consacrée à la représentation du problème d'ordonnancement de type « Job Shop Flexible » qui fait l'objet de cette étude.

1 Système de production :

1.1 Définition :

Un système de production est l'association d'un ensemble de ressources en interaction pour réaliser une activité de production.

En effet, la production s'effectue par une succession d'opérations dites de transformation, de transfert, d'assemblage et de désassemblage en exploitant les ressources disponibles (machines) afin de transformer les matières premières qui entrent dans le système en produits finis sortant de ce système (Letouzey, 2001).

Ainsi un système peut être classé selon ce qu'il produit :

- Système produisant des « objets » (par exemple, un système de fabrication de moteurs).
- Système produisant un « comportement » (par exemple, un système de climatisation) (SENECHAL O. , 2004).

1.2 Pilotage d'un système de production :

Il existe plusieurs définitions du terme pilotage dans la littérature. Parmi ces définitions on trouve :

Le pilotage consiste à décider dynamiquement des consignes pertinentes à donner à un système soumis à une perturbation pour atteindre un objectif donné décrit en termes de maitrise de performance (Trentesaux, 2002).

Le concept de pilotage concerne la définition et l'organisation des interrelations entre un système physique (système opérant) et son système de décision (pilotage) conduisant à la mise en œuvre d'une boucle de rétroactions.

2 Généralité sur l'ordonnancement :

2.1 Définition :

Nombreuses sont les définitions de l'ordonnancement ; dans ces définitions on trouve l'aspect commun de l'affectation de ressources aux tâches.

Etant donné un ensemble de tâches à accomplir, l'ordonnancement consiste à déterminer quelles opérations doivent être exécutées, et à assigner des dates et des ressources à ces opérations de façon à ce que les tâches soient, dans la mesure du possible, accomplies en un temps minimal et à moindre coût (vachar, 2000).

Donc, l'ordonnancement consiste à programmer et à planifier l'exécution des tâches en leur attribuant les ressources nécessaires matérielles ou humaines.

2.2 Les objectifs d'ordonnancement :

Dans un ordonnancement, il existe différents critères qui doivent être satisfaits, d'une manière générale, on distingue plusieurs classes d'objectifs pour un ordonnancement :

- ✓ **Les objectifs liés au temps :** par exemple, la minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales, etc…
- ✓ **Les objectifs liés aux ressources :** parmi les objectifs de ce type on trouve la maximisation de la charge d'une ressource ou la minimisation du nombre de ressources nécessaires pour réaliser un ensemble de tâches
- ✓ **Les objectifs liés au coût :** ces objectifs sont généralement de minimiser les coûts de lancement, de production, de stockage, de transport, etc…

2.3 Problème d'ordonnancement :

Dans un problème d'ordonnancement, il s'agit d'ordonnancer des activités sur des ressources disponibles en quantité limitée. Les activités sont liées entre elles par des relations de précédence, qui expriment qu'une activité i ne peut pas commencer tant que l'activité jqui la précède n'est pas achevée. L'objectif est de déterminer une solution qui minimise la date de fin d'ordonnancement, en respectant les contraintes de précédence et les contraintes de ressources (KONE, 2009).

Donc, un problème d'ordonnancement consiste à organiser l'exécution d'un ensemble d'activités soumises à des contraintes de temps et de ressources.

2.4 Eléments d'un problème d'ordonnancement :

La formalisation et la description du problème d'ordonnancement se fait par la détermination de quatre éléments fondamentaux : les tâches, les ressources, les contraintes et les objectifs (Hentous, 1999).

2.4.1 Les taches :

Une tâche est un travail mobilisant des ressources et réalisant un progrès significatif dans l'état d'avancement du projet compte tenu du niveau de détail retenu dans l'analyse du problème (giard, 1988).Constituée d'un ensemble d'opérations.

Une tâche qu'on note **i** est une entité élémentaire caractérisée par :

- Une date de disponibilité (date à laquelle elle peut être exécutée au plus tôt r_i)
- Une date d'échéance (appelée date de fin au plus tard d_i)
- Une durée opératoire (durée d'exécution p_i) (Esquirol & LOPEZ, 1999).

On distingue deux types de taches :

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

Le problème de *job shop* appartient à la première catégorie, où chaque tâche est constituée d'un ensemble d'opérations liées entre elles par des contraintes (figure1).

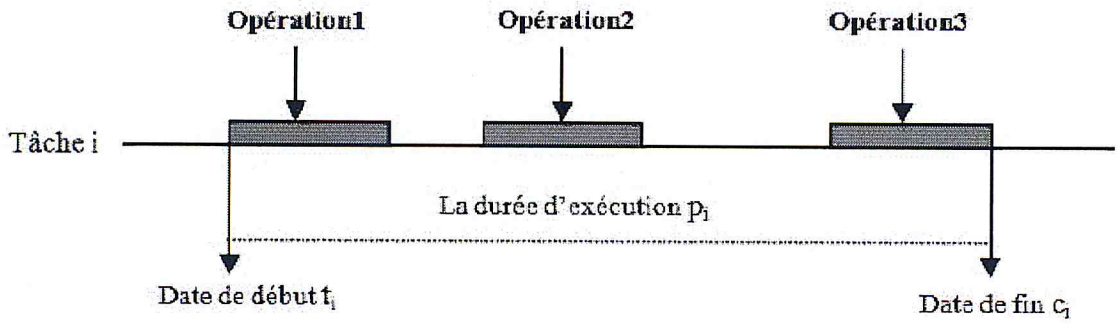

Figure 1 : Exemple de tâche avec trois opérations

2.4.2 Les ressources :

Une ressource est un moyen technique ou humain à utiliser pour réaliser une tâche, et disponible en quantités limitées. On trouve plusieurs types de ressource :

- **Ressource renouvelable** : c'est une ressource qui peut redevenir disponible en même quantité après avoir été allouée à une tâche (machines, personnel,….)
- **Ressource consommable** : c'est une ressource qui devient non disponible après avoir été utilisée pour la réalisation d'une tâche (argent, matières premières,…).

La disponibilité d'une ressource peut varier au cours du temps, dans le cas des ressources renouvelables, on trouve deux types de ressources :

- **Ressources disjonctive** : une ressource est dite disjonctive (ou non partageable) lorsque il s'agit d'une ressource qui ne peut exécuter qu'une tâche à la fois.
- **Ressources cumulative** : une ressource est dite cumulative (ou partageable) lorsque il s'agit d'une ressource qui peut être utilisée par plusieurs tâches simultanément mais en nombre limité (ESQUIROL & LOPEZ, 1999).

2.4.3 Les contraintes :

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décision (Esquirol & LOPEZ, 1999)

Il existe plusieurs types de contrainte :

- **Les contraintes de ressources :** une contrainte de ressources représente le fait que les activités utilisent une certaine quantité de ressources, tout au long de leur exécution. (Baptiste, 1998)

On distingue deux types de contrainte de ressources liées à la nature disjonctive ou cumulative des ressources. Ces contraintes sont soit :

- Des contraintes d'utilisation des ressources qui expriment la nature, la quantité et les caractéristiques d'utilisation de ces ressources,
- Soit des contraintes de disponibilité des ressources qui déterminent les quantités des ressources disponibles au cours du temps. (Esquirol & LOPEZ, 1999)

- **Les contraintes temporelles :** généralement se sont les contraintes de temps alloué, issues d'impératifs de gestion et relatives aux dates limites des tâches ou à la durée totale d'un projet. On a surtout :
  - La date de disponibilité : avant laquelle la tâche ne peut pas commencer.
  - La date d'échéance : avant laquelle la tâche doit être achevée.
- **les contraintes d'enchainement :** une contrainte d'enchainement, ou de succession, est une contrainte qui lie le début et la fin de deux activités par une relation linéaire. Ce sont des contraintes imposées par la cohérence technologique qui décrivent le positionnement relatif de certaines tâches par rapport à d'autres. Il existe d'autres contraintes plus spécifiques entre deux ou plusieurs tâches comme la synchronisation (Esquirol & LOPEZ, 1999) .

2.4.4 Les objectifs (les critères d'optimisation) :

Il existe plusieurs types d'objectifs :

- **Liés au temps :**
  - Le temps total d'exécution ou le temps moyen d'achèvement d'un ensemble de tâches.
  - Différents retards (maximum, moyen, somme, nombre, etc.,…) par rapport aux dates limites fixées.

- **Liés aux ressources :**
  - La quantité totale ou pondérée de ressources nécessaires pour réaliser un ensemble de tâches.
  - La charge de chaque ressource.
- **Liés à une énergie ou à un débit**
- **Liés aux coûts** de lancement, de production, de transport, etc (HABIBA, 2012).

3 Le problème d'ordonnancement d'atelier :

Dans un problème d'ordonnancement d'atelier, il existe deux notions principales : les ressources et les tâches.

- ✓ ***Les ressources :*** sont des machines de type Ressources disjonctives (ne pouvant réaliser qu'une opération à la fois) et sont renouvelables.
- ✓ ***Les tâches*** : chaque tâche est composée de plusieurs entités appelées opérations.

Une tâche ne pouvant se trouver simultanément en deux lieux distincts, donc elle ne peut être exécutée qu'à raison d'une opération à la fois sur une seule des machines.

3.1 Classification des problèmes d'ordonnancement :

Généralement on classifie les problèmes d'ordonnancement selon l'organisation de l'atelier. Les modèles d'ateliers sont :

- Les organisations à une machine.
- Les organisations à machines parallèles.
- Les organisations à machines multiples.

3.1.1 Les organisations à une machine :

Dans ce cas, l'ensemble des tâches à réaliser est fait par une seule machine, les tâches sont alors effectuées lors d'une opération.

Les organisations à ressource unique sont un cas que l'on peut rencontrer rarement dans les entreprises industrielles, par contre l'une des situations intéressantes où on peut rencontrer ce genre de configuration est le cas où on est devant un Système de Production comprenant une machine goulot qui influence l'ensemble du processus, ou bien les systèmes informatiques partagés où l'on dénote souvent la présence d'une seule ressource : une imprimante, un seul microprocesseur, etc... avec plusieurs usagers (TAGHEZOUT, 2011).

3.1.2 Les organisations à machines parallèle :

Dans ce cas, on dispose d'un ensemble de machines identiques pour réaliser les travaux. Ces dernières se composent d'une seule opération, et une tâche exige une seule machine. L'ordonnancement s'effectue en deux phases :

- La première phase consiste à effectuer les travaux sur machines
- La deuxième phase consiste à établir la séquence de réalisation sur chaque machine (TAGHEZOUT, 2011).

3.1.3 Les organisations a machine multiple :

Dans les organisations à machines multiples, il existe plusieurs machines sur lesquelles seront exécutées plusieurs tâches. Ce sont des organisations énormément répandues dans le domaine de la production industrielle, constituant ainsi une catégorie différente d'ateliers.
Trois types ont une importance particulière dans les problèmes d'ordonnancement : le flow shop, le job shop et l'open shop.

3.1.3.1 Atelier a acheminement unique : le flow shop :

Dans le cas des organisations de type Flow Shop Simple (Hentous & Guinet, 1997)les tâches ont un même cheminement. Une tâche est ainsi constituée d'opérations visitant différentes machines et enchaînées de manière linéaire suivant une chaîne (Esquirol & LOPEZ, 1999)

Ce sont des ateliers où une ligne de fabrication est constituée de plusieurs machines en série et où toutes les opérations de toutes les tâches passent par les machines dans le même ordre. Autrement dit, on a *m* machines en série et chaque tâche doit être exécutée sur chacune de ces machines.

Ainsi, toutes les tâches doivent s'exécuter sur la machine 1, puis sur la machine 2,…, et ainsi de suite dans le même ordre. Après la fin d'exécution sur une machine, la tâche rejoint la file d'attente de la machine suivante, et ne passe qu'une seule fois sur la machine.

3.1.3.2 Atelier a acheminement libre : open shop :

Dans les organisations de type Open Shop Simple ou Hybride, les tâches ont une gamme non précisée (libre), autrement dit il n'existe aucun ordre d'exécution des opérations. Dans les problèmes d'ordonnancement de production dits à cheminement libre (ou sans contraintes

d'enchaînement), ce type d'atelier est moins contraint que celui de type flow shop ou de type job shop.

Les problèmes d'ordonnancement consistent, d'une part, à déterminer le cheminement de chaque tache et d'autre part à ordonnancer les taches en tenant compte des gammes trouvées. Chaque produit à fabriquer doit subir diverses opérations sur des machines, mais dans un ordre totalement libre (Esquirol & LOPEZ, 1999).Open shop n'est pas couramment utilisé dans les entreprises.

3.1.3.3 Atelier a acheminement multiple : Job shop :

Les ateliers à cheminement multiple (Job-shop) sont des unités manufacturières traitant une variété de produits individuels dont la production requiert divers types de machines dans des séquences variées (TAGHEZOUT, 2011). Les opérations sont réalisées selon un ordre total bien déterminé, variant selon la tâche à exécuter.

Contrairement au type d'atelier précédent, l'atelier job shop se caractérise par un cheminement multiple puisque les opérations de chaque job peuvent emprunter divers chemins.

L'objectif le plus considéré dans le cas d'un atelier à cheminements multiples est le même que celui considéré pour un atelier à cheminement unique. L'objectif est de trouver une séquence de tâches sur les machines qui minimise le temps total de production (ESQUIROL & LOPEZ, 1999).

La figure suivante représente un atelier de type job shop :

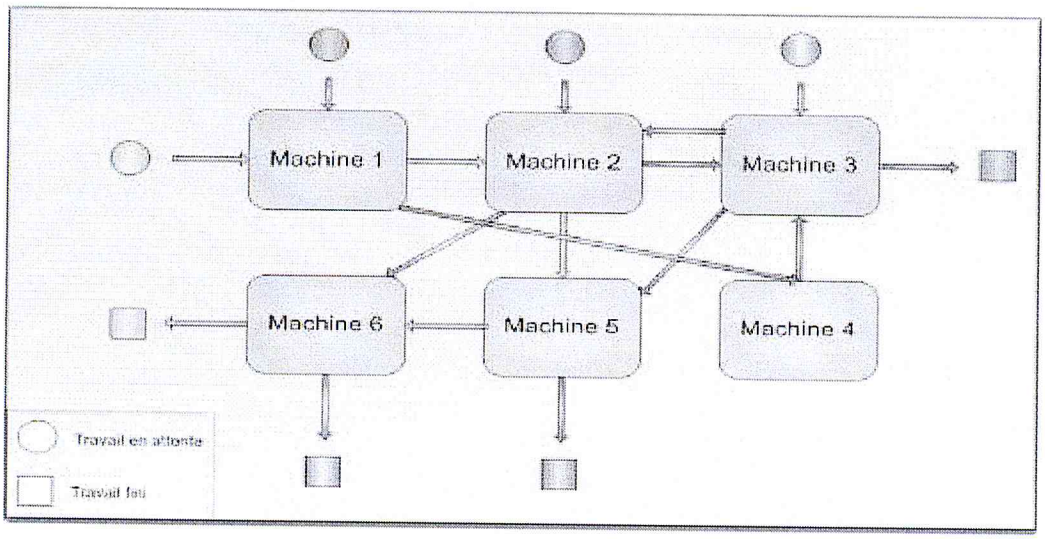

Figure 2 : Organisation d'atelier à cheminements multiples (Job Shop)

Le job hop flexible :

Le job shop flexible est une extension du modèle job shop classique. La flexibilité est due à l'attribution d'un sous-ensemble de ressources (machines) pour le traitement de chaque opération de telle sorte que le temps opératoire de chaque opération dépende de la machine sélectionnée (Meziane amine, 2011). Sa particularité essentielle est que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations. Plus précisément, une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération.

4 Problème de job shop flexible :

4.1 Définition :

Le problème Job Shop Flexible **(FJSP)** est une généralisation du problème de Job Shop classique. Dans ce type de problème, une opération nécessite une machine pour être réalisée, et cette machine peut être choisie dans un ensemble défini dans un ensemble de machines a priori. Le problème d'ordonnancement consiste alors à affecter une machine à chaque opération, et à déterminer la séquence des opérations sur les machines dans le but de minimiser un critère donné.

4.2 Formulation des problèmes job shop :

Le problème de job shop flexible constitue parmi les problèmes d'ordonnancement les plus étudiés.

Les problèmes d'ateliers Job Shop Flexibles peuvent être formulés comme suit : (Vacher, 2000)

- Soit un ensemble de n produits J_j, $j=1, \ldots, n$ indépendants à réaliser sur m machines M_k, $k = 1, \ldots, m$.
- Chaque produit $J_j$est constitué d'une séquence de $n_j$opérations$O_{i,j}$, $i = 1, 2, \ldots, n_j$, à exécuter selon un ordre bien défini.
- L'exécution de chaque opération i d'un job $J_j$nécessite une machine 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 $O_{i,j}$à une machine $M_k$entraîne l'occupation de cette machine durant tout le temps d'exécution de l'opération, noté $p_{i,j,k}$.

4.3 Les contraintes :

Les contraintes du problème représentent les contraintes technologiques auxquelles sont soumises les tâches et les machines. Elles concernent l'utilisation des machines et les liens existants entre les opérations.

- Les machines sont indépendantes les unes des autres (pas d'outils communs par exemple).
- Les machines sont disponibles jusqu'à la fin de l'ordonnancement, en particulier, les pannes des machines ne sont pas prises en compte.
- Une opération ne peut être en état d'exécution que sur une seule machine à la fois.
- Les tâches sont indépendantes les unes des autres, il n'existe aucun ordre de priorité attaché aux tâches.
- Deux opérations de la même tâche ne peuvent être exécutées simultanément.
- Une machine ne peut exécuter qu'une seule opération à un instant donné.
- Seulement le temps d'exécution, proprement dit, est pris en compte. Les temps de transport d'une machine à l'autre, de préparation,... ne sont pas considérés.
- Les tâches sont autorisées d'attendre les machines autant qu'il faut. Il n'y a pas de date d'échéance.
- Une opération en cours d'exécution ne peut être interrompue.
- Une opération peut avoir un ou plusieurs choix de machines avec une durée opératoire donnée.
- Les machines peuvent exister en plusieurs exemplaires non-reliées, c'est-à-dire les durées opératoires dépendent de la ressource choisie.
- Une opération donnée sera exécutée par une seule machine.

4.4 Les objectifs :

L'objectif du problème d'ordonnancement est de fixer les dates de début des opérations. Pour cela il faut déterminer à la fois l'affectation des machines et l'ordre de passage de l'ensemble des opérations sur chaque machine, en respectant les contraintes.

Le but est de minimiser ou maximiser la fonction objectif pour trouver la ou les meilleures (s) solutions (s). Cette fonction objectif appelée aussi critère de performance, critère d'évaluation ou objectif d'ordonnancement.

5 La complexité des problèmes et des algorithmes :

La complexité algorithmique est un concept fondamental qui permet de mesurer les performances d'un algorithme et permet aussi de le comparer avec d'autres algorithmes réalisant les mêmes fonctionnalités.(OURARI, 2011)

La classe P regroupe les problèmes résolubles pour des algorithmes polynomiaux. Un algorithme est polynomial lorsque son temps d'exécution borné par O(p(x)) où p est un polynôme et x la largeur d'entrée d'une instance du problème. (Simon, 1982)

Les algorithmes dont la complexité ne peut pas être bornée polynomiale correspondent à la classe NP (non-déterministe polynomiale)

Parmi les classes de NP on trouve les problèmes NP-difficile qui représentent une large classe de NP.

Les problèmes d'ordonnancement sont en général NP-difficiles. Même s'il est possible de résoudre de manière optimale des problèmes de petite taille, il devient difficile de trouver une bonne solution admissible (respectant toute les contraintes du problème) pour un grand problème.

6 Conclusion :

Dans ce chapitre, nous avons tout d'abord défini ce qu'est un système de production, son pilotage, ensuite nous avons décrit les caractéristiques générales d'un problème d'ordonnancement, tout en précisant notamment les différents éléments qui le déterminent ainsi que la classification de ces problèmes et les modèles d'organisation les plus importants.

Nous avons accordé une attention particulière à la fin du chapitre au problème de type job shop flexible qui est l'objet de notre étude.

Dans la partie qui va suivre, nous allons aborder les méthodes et les techniques existantes utilisées afin de résoudre ce problème.

La résolution optimale d'un problème d'ordonnancement consiste en la détermination d'une séquence d'exécution des tâches sur les machines de sorte qu'un certain critère d'optimisation atteigne sa valeur optimale.

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, théorie des graphes, …). Ces méthodes garantissent en général l'optimalité de la solution fournie.

Mais vu la complexité des problèmes d'ordonnancement, ces méthodes deviennent très vite inadaptées lorsque la taille du problème croit, d'où la nécessité d'utiliser des méthodes de résolution approchée, qu'on appelle méthodes heuristiques, généralement efficaces pour les problèmes NP-difficiles.

Nous présentons dans ce qui suit les méthodes de résolution les plus connues.

1 Problème d'optimisation multi-objectif :

Un problème d'optimisation mono-objectif en général est défini par un espace de recherche S et une fonction objectif F, le but étant de trouver la solution s∈S de meilleure qualité F(s). Donc on cherche soit à maximiser soit à minimiser la fonction objectif selon le problème.

Dans la plupart des problèmes d'optimisation, plusieurs critères sont à prendre en considération afin d'obtenir une solution satisfaisante. L'optimisation multi objectif a donc pour but d'optimiser plusieurs objectifs simultanément.

Généralement, les objectifs sont en conflit (l'amélioration d'un objectif provoque la détérioration d'un autre objectif). Dans le cas multi-objectif, le concept d'optimisation est différent que dans le cas mono-objectif. En effet, on n'est plus ici à la recherche d'un optimum global, mais plutôt d'une surface de solution qui offre un bon compromis entre les différents objectifs.

2 Les méthodes de résolution d'un problème d'optimisation multi objectif :

2.1 Les méthodes exactes :

Une méthode exacte garantie de trouver la solution optimale et de la prouver. Ces méthodes sont basées sur une résolution algorithmique.

Le temps nécessaire pour trouver la solution optimale augmente avec la taille du problème. Les problèmes de petite ou moyenne taille peuvent être résolus de façon optimale par des algorithmes exacts.

Bien que les méthodes exactes aient l'avantage d'obtenir des solutions optimales garanties, leur inconvénient est le temps de résolution qui est lié à la taille du problème.

Il existe plusieurs méthodes exactes, nous citons : programmation dynamique, programmation linéaire, recherche arborescente, procédure par séparation.

2.2 Les méthodes approchées :

L'utilisation de méthodes exactes n'est pas toujours possible pour résoudre un problème donné à cause d'un certain nombre de contraintes, telles que le temps d'exécution qui est souvent important ou bien la difficulté surtout lorsque la taille du problème devient importante. La recherche d'une solution optimale dans un espace important devient difficile

CHAPITRE 2

Les méthodes de résolution d'un problème d'optimisation multi objectifs

surtout en un temps raisonnable car l'exploration complète de l'espace de la solution prend un temps.

Pour cela, les chercheurs ont adopté des méthodes de résolution approchées qui sont basées sur des techniques d'exploration de l'espace plus rapide que les méthodes exactes. Les méthodes approchées peuvent fournir des solutions optimales en un temps raisonnable. Les méthodes approchées sont considérées pour les problèmes d'ordonnancement dans lesquelles il est difficile de trouver une solution optimale en un temps raisonnable.

2.2.1 Les méthodes heuristiques :

Une méthode heuristique est une méthode approchée se voulant simple et rapide. Généralement les méthodes heuristiques utilisent des règles simples pour optimiser un ou plusieurs critères. Le principe général de cette catégorie est l'intégration des stratégies de décision pour avoir une solution approchée de l'optimal en un temps raisonnable.

Les méthodes heuristiques sont adaptées à un problème donné. L'inconvénient de ces méthodes approchées est le fait qu'elles n'offrent aucune garantie de la solution optimale. Ce défaut n'est pas toujours un problème mais la plupart des heuristiques ont été conçues spécialement pour un problème donné. (Kacem, 2003).

2.2.2 Les méthodes métaheuristiques :

Contrairement aux heuristiques, les métaheuristiques visent à résoudre une large gamme de problèmes différents sans changements majeurs dans l'algorithme (Dréo, 2004).

Les métaheuristiques permettent de réduire d'une manière considérable l'espace de recherche. Ces méthodes fournissent une solution quasi-optimale (une solution qui s'approche de l'optimum tout en réduisant au maximum les couts).

2.3 Classe des approches de résolution multi objectif :

2.3.1 Les approches à base de transformation :

Ces approches transforment le problème initial afin de le ramener à la résolution d'un ou plusieurs problèmes mono-objectifs. Parmi ces méthodes on peut citer les méthodes d'agrégation et les méthodes de programmation par but.

Les méthodes à base de transformation nécessitent une connaissance du problème et ne fournissent qu'une seule solution.

2.3.2 Les approches Non Pareto :

Ces approches ne sont ni à base de transformation, ni agrégative, ni Pareto, elles transforment le problème d'origine et elles effectuent leur recherche en traitant indépendamment chacun des objectifs et d'une façon équivalente (collette & siarry, 2002).

Parmi ces approches, les approches à sélection parallèle VEGA (Vector Evaluated Genetical Algorithm) qui optimise en parallèle chaque objectif sur un ensemble de solutions indépendamment des autres objectifs. Les approches a sélection lexicographique où les objectifs sont considérés rangés par ordre d'importance par le décideur ; la solution optimale est obtenue en minimisant le premier objectif d'abord, le deuxième, et ainsi de suite jusqu'au dernier objectif.

L'inconvénient de ces approches est qu'elles ont souvent du mal à trouver la solution de compromis parce qu'elles se focalisent sur les parties extrêmes du front.

2.3.3 Les approches Pareto :

Ces approches sont basées sur la notion de dominance pour comparer les solutions. Contrairement aux méthodes précédentes, celles-ci ne font subir aucune transformation au problème multi-objectif ni une préférence pour un objectif par rapport à un autre.

Les objectifs sont traités de la même façon et les solutions optimales sont celles qui ne sont pas dominées au sens de Pareto. (Goldberg, 1989).

2.4 Dominance au sens Pareto :

- **Définition** :

Un point $x \in E$ domine $x' \in E$ si $\forall i, fi(x) \leq fi(x')$ avec au moins un i tel que $fi(x) < fi(x')$ (Berro, 2008)

- ***Optimalité de Pareto :***

Les solutions qui dominent les autres et qui ne se dominent pas entre elles sont appelées solutions optimales au sens de Pareto (solution non dominées)

- ***Frontière de Pareto :***

La frontière ou bien le front Pareto, est l'ensemble des points Pareto optimaux, La figure suivante montre un exemple de ça :

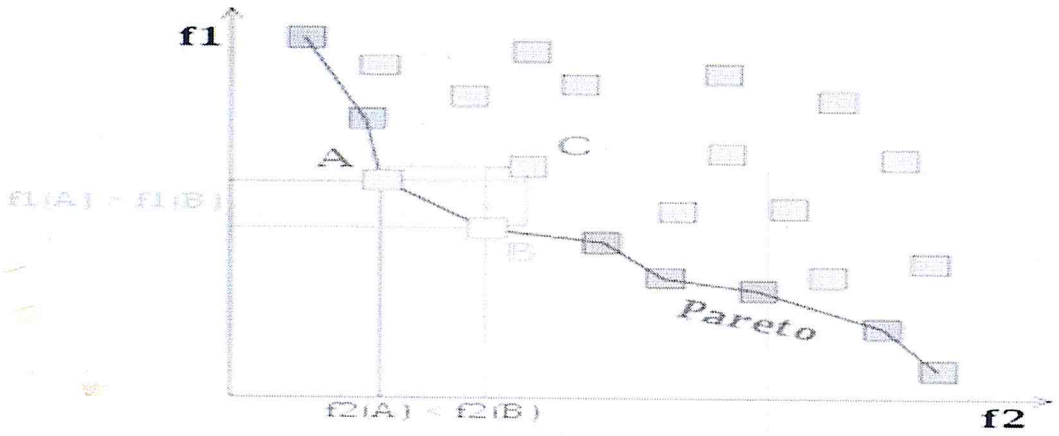

Figure 3 : le front Pareto.

3 Les algorithmes évolutionnaire :

3.1 Définition :

Les algorithmes évolutionnaires sont des algorithmes d'optimisation stochastique se basant sur le principe de l'évolution naturelle darwinienne des populations biologiques.
Un algorithme évolutionnaire a pour but d'optimiser une fonction réelle et de faire évoluer un ensemble de solutions (population) vers une solution qui approche a l'optimale.

Les algorithmes évolutionnaires ont été appliqués avec succès à de nombreux problèmes où les algorithmes classiques d'optimisation sont incapables de produire des résultats satisfaisants.

3.2 Principe de fonctionnement :

Les algorithmes évolutionnaires reposent sur une vision darwinienne relativement simpliste et une optimisation stochastique, la fonction à optimiser, appelée aussi performance, est définie sur un espace de recherche. L'algorithme fait évoluer une population, un sous-ensemble de l'espace de recherche. Cette évolution résulte d'une part d'un darwinisme artificiel, qui se manifeste par la sélection et le remplacement et ne dépend que de la performance, d'autre part de l'effet du stochastique, qui s'exprime dans l'initialisation et les opérateurs de variation et ne dépend que de la représentation de l'espace de recherche.

L'idée fondamentale est que la sélection favorise les individus qui optimisent la performance et que les variations font apparaitre dans la population sélectionnée des individus que l'on peut espérer meilleurs au regard de la performance. Dans cette évolution, les générations successives de la population restent à taille constante et l'aspect stochastique ne dépend que de la génération précédente (schoenauer, 2011).

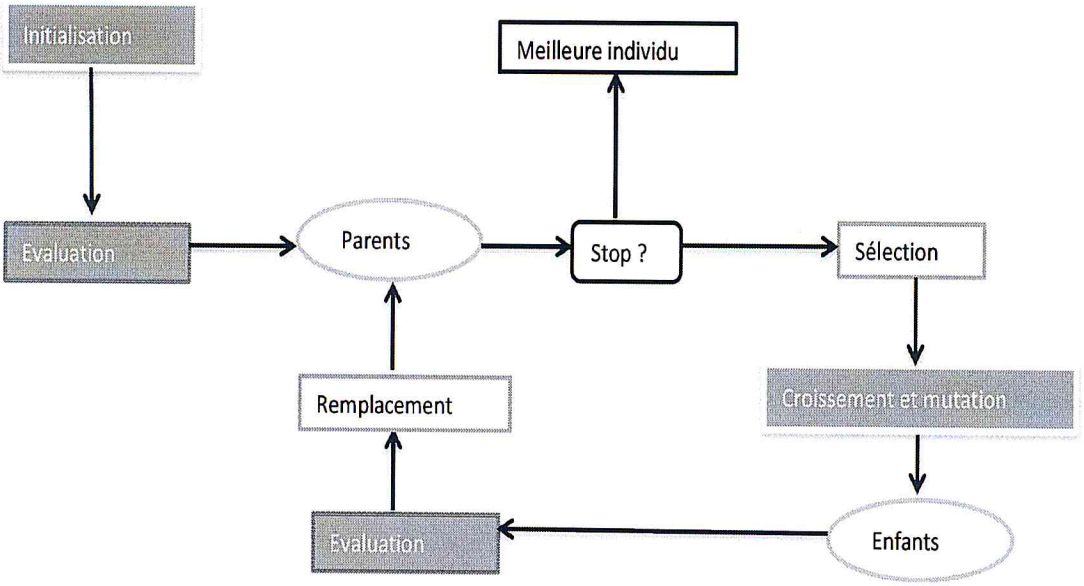

Figure 4 : principe de fonctionnement d'un algorithme évolutionnaire.

Algorithme évolutionnaire

1- Initialisation :

- Initialiser le compteur de génération t=0
- Initialiser aléatoirement une population p(t) de taille fixée N.
- Evaluer chaque individu de p(t).

2- Sélectionner les individus les plus performants de la population, les recopier pour former une nouvelle population de même taille.

3- Créer une nouvelle population en appliquant les opérateurs de variation :

- Opérateur de croisement : combiner deux individus pour obtenir deux nouveaux.
- Opérateur de mutation : modifier un individu

4- Evaluer le fitness de chaque individu de la nouvelle population.

5- Remplacer certains individus de l'ancienne population par les meilleurs individus de la nouvelle population.

6- Si la condition d'arrêt n'est pas vérifiée, aller vers l'étape 2, sinon retourner le meilleur individu de p (t).

3.3 Exploitation et exploration :

L'utilisation de l'exploitation et l'exploration revient à effectuer une recherche locale dans le voisinage de meilleurs individus. L'idée est d'exploiter efficacement l'information obtenue précédemment par les meilleures solutions pour spéculer sur la position de nouveaux points avec l'espoir d'améliorer la performance.
Toutefois, l'exploitation toute seule ne permet pas de préserver la diversité génétique. La population devient homogène, et dans ce cas l'évolution d'une population risque de se résumer à l'évolution d'un seul individu dominant.

L'exploration consiste à diriger la recherche de façon à préserver une population diversifiée. L'exploration des nouvelles régions de l'espace de recherche permet d'introduire dans la population des informations innovatrices. Cependant, l'excès d'exploration conduit à une quasi recherche aléatoire qui empêche la convergence. Il est important de réaliser un compromis entre l'exploitation et l'exploration. Il faut donc maintenir un équilibre entre l'exploitation de bonnes solutions rencontrées et l'exploration de zones inconnues de l'espace de recherche pour garantir l'efficacité de l'algorithme (Roudenko, 2004).

3.4 Représentation des individus :

On peut distinguer plusieurs types de représentation des individus, parmi les plus connues : la représentation binaire et la représentation réelle.

La représentation (codage) des individus doit :

- Etre manipulable par des opérateurs de variation (transformations génétiques : mutation, croisement).
- Permettre une transition simple et facile et efficace vers l'espace de recherche.

3.4.1 Codage réel :

Le principe de cette représentation est de coder directement les variables du problème dans l'individu. Les individus sont donc représentés par des vecteurs réels.

L'avantage de cette représentation est de conserver les variables du problème, ce qui lui permet une meilleure prise en compte de la structure du problème et permet d'avoir une grande précision dans les résultats. Malgré cela, l'inconvénient de cette représentation est de définir de nouveaux opérateurs de variation génétique bien spécifiques. (Roudenko, 2004).

3.4.2 Codage binaire :

Les algorithmes évolutionnaires utilisent habituellement des chaines de bits pour représenter les individus. Le codage binaire est généralement utilisé dans les algorithmes évolutionnaires, où chaque individu est représenté par un vecteur binaire (chaine de bits) et où chaque élément prend la valeur 1 ou 0.

Malgré son efficacité ce type de codage présente des inconvénients pour des problèmes où on veut une grande précision dans les résultats, dans ce cas, le codage binaire devient inadapté (GOLDBERG & DAVID, 1989).

3.5 Evaluation des individus :

Chaque fois qu'un nouvel individu a été créé, il faut lui associer une valeur appelée fitness, ou fonction d'évaluation, qui mesure la qualité de la solution.

L'évaluation représente donc la performance de l'individu vis-à-vis du problème à résoudre.

Cette valeur sera utilisée par les processus de sélection pour favoriser les individus les mieux adaptés(les meilleures solutions au problème).

3.6 Opérateurs génétique :

3.6.1 Sélection :

La sélection est un opérateur essentiel dont le principe consiste à permettre aux meilleurs individus d'une population de se reproduire.

La sélection détermine combien de fois un individu participe à la reproduction ou sera reproduit en une génération. Les individus ayant les meilleures performances sont sélectionnés plus souvent que les autres.

Les individus sélectionnés, appelés parents, sont ensuite disponibles pour la phase de variation. Il existe différents types de sélection :

- ***Sélection proportionnelle :***

Plusieurs stratégies de sélection des individus sont basées sur le principe de sélection proportionnelle, la chance qu'a un individu d'être sélectionné est proportionnelle à sa valeur de fitness, on trouve dans cette famille :

- ***La sélection par roulette :***

Dans la sélection par roulette, la population est représentée comme une roue de roulette, et où chaque individu est représenté par une portion qui correspond proportionnellement à sa valeur de fitness. La sélection d'un individu se fait en tournant la roue en face d'un pointeur fixe.

L'un des inconvénients de ce type de sélection est de choisir presque toujours le même individu s'il en existe un bien meilleur que les autres, ce qui cause une perte de diversité dans la population (Goldberg, 1989).

- ***La sélection par rang :***

La sélection par rang se compose de deux étapes :

La première étape consiste à ranger tous les individus de la population selon leur fitness : Le rang est fait dans l'ordre décroissant (ou croissant), selon si on veut minimiser (ou maximiser) la fonction de fitness. Normalement, les individus de moins bonne qualité obtiennent un rang faible (à partir de 1).Dans la deuxième étape on choisit le nombre d'individus à sélectionner et on sélectionne selon l'ordre.

3.6.2 Croisement :

Le croisement est un mécanisme qui consiste à appliquer des procédures avec une certaine probabilité qui s'appelle le taux de croisement sur les individus sélectionnés pour donner naissance à un ou plusieurs enfants. Ces derniers doivent hériter, par croisement de certaines caractéristiques des parents.

Le croisement a un rôle d'exploration, car la combinaison des solutions produit des solutions placées aux endroits non encore visités pendant la recherche, et en même temps un rôle de l'exploitation lorsqu'à la fin de la recherche, les parents sont similaires, ce qui produit des enfants similaires (Benlahrache, 2007).

3.6.3 Mutation :

L'opérateur de mutation modifie de manière complètement aléatoire les caractéristiques d'un ou plusieurs individus de la population. Il modifie un ou plusieurs gènes pour passer d'une solution à une autre solution de forme similaire mais qui peut avoir une évaluation totalement différente. L'individu muté ne sera pas forcément meilleur ou moins bon, mais il apportera des possibilités supplémentaires qui pourraient être utiles pour la création de bonnes solutions (BENLAHRACHE, 2007).

Cet opérateur permet d'explorer l'espace de recherche en évitant à l'algorithme de converger trop rapidement vers un optimum local. Donc il peut :

- ✓ Favoriser l'exploration lorsque l'individu muté est éloigné de l'individu original
- ✓ Favoriser l'exploitation lorsque l'individu muté est proche de l'individu original (ABDELLAH, 2005).

3.7 Type d'algorithme évolutionnaire :

3.7.1 Les algorithmes génétiques :

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique (la survie des individus les mieux adaptés à un environnement et de l'évolution naturelle : croisements, mutations, sélection, etc.)

Ils ont été exposés par (Holland 1975) puis vulgarisés par (Goldberg, 1989).

Les algorithmes génétiques sont les plus populaires des algorithmes évolutionnaires.

Algorithme génétique

1. Créer la population initiale (de taille N)
2. Tant que le critère d'arrêt n'est pas vérifié faire
   a. Evaluation de chaque individu de la population
   b. Sélection proportionnelle des parents compte tenu de leurs fitness.
   c. Reproduction :
      - Combiner les parents afin de produire une (ou plusieurs) nouvelle(s) solution(s) (croisement)
      - Appliquer un opérateur génétique qui crée des enfants dont une partie des gènes sont aléatoires ; (mutation)
   d. Remplacement : sélectionner N solutions pour constituer une nouvelle génération de taille N.
3. Fin du tant que.

3.7.2 Programmation évolutive :

La programmation évolutive s'appuie sur un codage approprié du problème à résoudre et sur les opérations de mutation adaptées pour le codage.

Ce modèle évolutionnaire utilise la mutation et n'utilise pas dans sa version originale la recombinaison des individus par croisement. Développé à l'origine pour l'évolution des

automates à état fini, ce modèle est souvent appliqué à la résolution de problèmes d'optimisation à variables réelles dans des espaces de recherche très variés.

L'idée consiste à faire subir des mutations importantes aux mauvais individus et des mutations faibles aux bons individus. L'opérateur de sélection est de type probabiliste (DUPAS, 2004).

3.7.3 Les stratégies d'évolution :

Elles ont été développées pour résoudre des problèmes d'optimisation à variables réelles posés au milieu industriel.

Ce sont les meilleurs algorithmes pour les problèmes purement numériques. Dans ce modèle de stratégies d'évolution, les seuls mécanismes utilisés sont la mutation et la sélection. Ce type d'algorithmes utilise le principe de mutation avec un taux de mutation plus grand. Le but d'augmenter le taux pour diversifier la population pour éviter que l'espace de recherche exploré ne soit limitée autour d'un optimum local.

Ces approches utilisent un opérateur de sélection de type déterministe. Donc les solutions dont la fonction d'adaptation (fitness) est mauvaise sont éliminées de la population (DUPAS, 2004).

4 Techniques avancés pour la résolution des problèmes multi objectif

4.1 Décomposition d'un problème multi objectif

4.1.1 Somme pondérée :

Comme son nom l'indique, cette approche considère une simple combinaison linéaire des différents objectifs. Soit $\lambda = (1;...; M)^T$ un vecteur de poids, P tel que $\lambda i \geq 0$ pour tout

$i = 1;... ; M$, et

$\sum \lambda i = 1$

Nous définissons alors le problème d'optimisation scalaire (à un seul objectif) suivant :

$$argmax \left(g^{ws}(x|\lambda) := \sum_{m=1}^{m} \lambda_i f_m(x) \right)$$

Où x désigne une solution, g^{ws} (x/ λ) désigne une fonction scalaire définie par rapport au coefficient au vecteur de poids λ donné en paramètre. Afin de générer un ensemble de solutions du front de Pareto, une idée consiste à utiliser différents vecteurs de poids λ. Cette approche ne permet pas cependant d'obtenir tous les vecteurs Pareto optimaux. (MARQUET, 2014).

4.1.2 Approche de Tchebycheff (Te) :

Dans cette approche, la fonction scalaire et le problème scalaire qui lui est associé sont définis de la façon suivante :

$$argmin \left(g^{te}(x|\lambda, z^*) := \max_{1 \leq m \leq M} \{\lambda_m \cdot |f_m(x) - z_m^*|\} \right)$$

Dans ce problème, la fonction g^{te} est paramétrée par le vecteur λ, mais aussi par un point $z^* = (z_1^* ; ... ; z_M^*)^T$ appelé point de référence. Au contraire de la fonction g^{ws} utilisée dans l'approche précédente, on sait démontrer que pour toute solution Pareto optimale x*, il existe un vecteur de poids λ tel que x* est une solution optimale de l'équation ; et inversement chaque solution optimale de l'équation est une solution optimale Pareto.

Par conséquent, on est capable d'obtenir différentes solutions Pareto optimales en modifiant le vecteur de poids. Une faiblesse de cette approche repose sur sa fonction d'agrégation non linéaire dans le cas d'un problème d'optimisation multi-objectif continu (MARQUET, 2014).

4.2 MOEA/D :

MOEA/D (Multi Objective Evolutionary Algorithme based on Decomposition) est un cadre de l'algorithme générique. Il décompose un problème d'optimisation multi-objectif en un certain nombre de différents sous-problèmes d'optimisation mono-objectif simples puis utilise une méthode basée sur la population pour optimiser ces sous-problèmes simultanément.

MOEA/D nécessite la définition de P vecteurs de poids uniformément distribués dans l'espace objectif. Ces vecteurs de poids définissent en fait des directions de recherche dans lesquelles l'algorithme va tenter de chercher des solutions.

L'algorithme maintient aussi une population de solutions de taille P également. En effet, chaque vecteur de poids est utilisé pour définir un sous-problème mono-objectif en utilisant les fonctions scalaires.

Une solution est alors attachée à chaque fonction scalaire. Le but de MOEA/D est de trouver les meilleures solutions par rapport à chacun des vecteurs de direction. Pour cela, l'originalité de MOEA/D consiste à définir la notion de voisinage de chaque sous-problème scalaire. Étant donné un vecteur de direction λi et le sous-problème scalaire sous-jacent pour

tout i ∈ {1, . . ., P}, on définit l'ensemble B(i) des voisins de i comme étant les sous-problèmes correspondants aux vecteurs de direction les plus proches de λ.

L'algorithme parcourt les sous-problèmes de façon itérative, l'un après l'autre. Pour chaque problème, en sélectionnant des solutions parmi les voisins sont sélectionnées. Ces solutions permettent alors de produire une nouvelle solution, notée y', en utilisant des opérateurs de variations classiques, comme la mutation et le croisement. Une fois la solution y' produite, elle est comparée aux solutions des voisins. Si y' permet d'améliorer la solution d'un voisin alors elle remplace cette solution et ainsi de suite pour tous les voisins. Ce mécanisme est répété pour toutes les directions jusqu'à ce qu'une condition d'arrêt soit satisfaite, par exemple un nombre maximum d'itérations, L'ensemble des solutions calculées par rapport à toutes ces directions est alors retourné en sortie (MARQUET, 2014).

4.3 Modèle parallèle des algorithmes évolutionnaire :

Pour des problèmes d'optimisation difficiles, exécuter le cycle de reproduction d'un AE standard sur de larges instances du problème et/ou sur de grandes tailles de population nécessite des ressources considérables en termes de calcul des ordinateurs.

Par conséquent, une variété de difficultés algorithmiques doit être étudiée pour concevoir des algorithmes évolutionnaires efficaces. Ces difficultés consistent habituellement à définir de nouveaux opérateurs, des algorithmes hybrides, des modèles parallèles, etc…une autre approche consiste à utiliser le parallélisme. Il existe trois modèles parallèles majeurs qui peuvent être distingués dans la recherche sur les algorithmes évolutionnaires.

4.3.1 Modèle d'évaluation parallèle de la population :

Généralement l'utilisation de ce modèle est intéressante et d'une grande importance lorsque la fonction d'évaluation peut être elle-même parallélisée dans la mesure où l'évaluation serait coûteuse en temps de calcul et/ou en paramètres d'entrées/sorties. Dans ce cas, la fonction peut être pensée/visualisée comme un groupement d'un certain nombre de fonctions partielles (sous-fonctions).

Ce modèle est utilisé lorsque l'évaluation de la population est la phase la plus lourde en temps de calcul. Lorsque la durée d'un calcul de performance varie d'un calcul à l'autre (que ce soit en raison de conditions de calcul différentes, ou de par l'hétérogénéité des processeurs utilisés), l'avantage apporté par ce schéma de parallélisation peut toutefois se dégrader ; l'ensemble des processeurs devront alors attendre le plus lent d'entre eux. (BERACHI, 2010).

4.3.2 Population distribuée :

Ce modèle de parallélisassions consiste à distribuer une unique population sur l'ensemble des processeurs. Sur chaque processeur "vivent" alors quelques individus (souvent un seul), et les opérations de sélection/remplacement et de croisement se font entre individus "voisins" pour la topologie du réseau de processeurs. (BERACHI, 2010)

A noter qu'il existe en général dans ce modèle un processeur superviseur (root), qui reçoit des informations de l'ensemble des processeurs et permet de suivre précisément l'évolution de l'ensemble en affichant diverses statistiques. C'est en particulier lui qui décidera de l'arrêt de l'algorithme.

4.3.3 Modèle en ile synchrone/asynchrone :

Le modèle suivant de parallélisassions des algorithmes évolutionnaires consiste à diviser la population en petites sous-populations, chaque sous-population évoluant sur un processeur, suivant un schéma traditionnel auquel vient s'ajouter une étape de migration (communication).

Dans ce modèle particulier des algorithmes évolutionnaires, plusieurs AEs sont simultanément exécutés dans un environnement de collaboration afin de trouver de meilleures solutions. (BERACHI, 2010).

La figure suivante représente le model en iles :

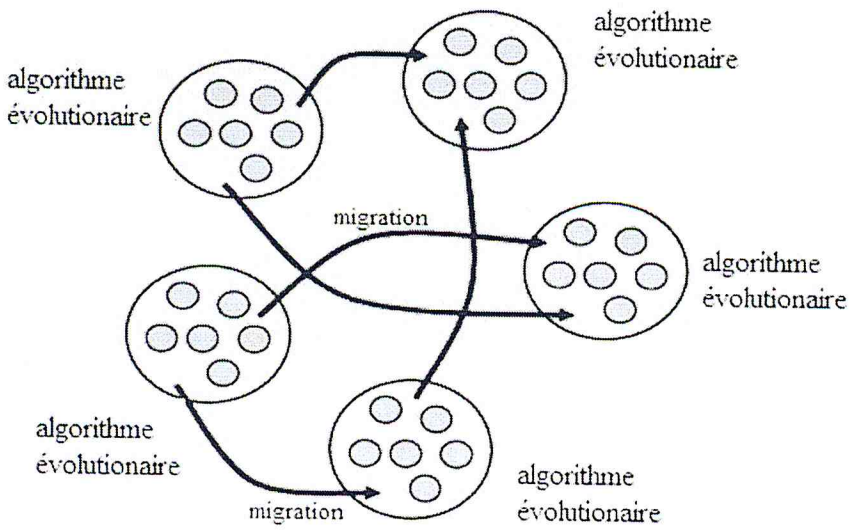

Figure 5 : un modèle en îles d'algorithmes évolutionnaires

5 EDA (Estimation Distribution Algorithm) :

5.1 Définition :

Les algorithmes à estimation de distribution (Estimation of Distribution Algorithms EDA) forment une famille de métaheuristiques inspirée des algorithmes génétiques. Ils sont utilisés pour résoudre des problèmes d'optimisation, via la manipulation d'un échantillonnage de la fonction décrivant la qualité des solutions possibles. Comme toutes les métaheuristiques utilisant une population de points, ils sont itératifs.

Les algorithmes à estimation de distribution ont été conçus à l'origine comme une variante des algorithmes évolutionnaires. Dans les versions les plus connues de type EDA, il n'y a pas d'opérateurs de croisement ou de mutation. En effet, la population des nouveaux individus est tirée au hasard, selon une distribution estimée depuis des informations issues de la population précédente. Dans les algorithmes évolutionnaires, la relation entre les différentes variables est implicite alors que, dans les algorithmes EDA, le cœur de la méthode consiste justement à estimer ces relations, à travers l'estimation de la distribution de probabilité associée aux individus. (DREO & SIARRY).

5.2 Historique :

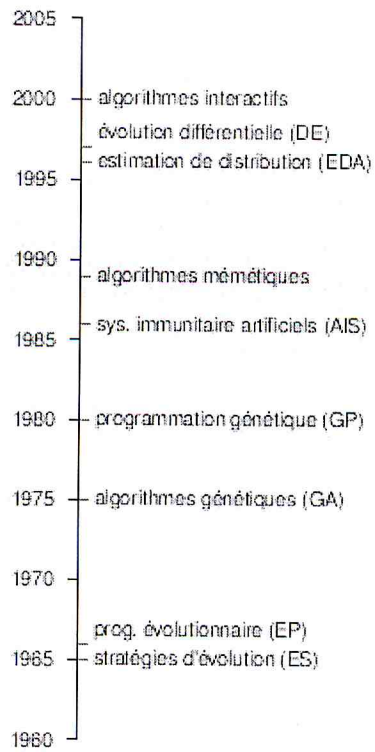

Figure 6 : Chronologie des principales métaheuristiques.

5.3 Comportement :

Le comportement d'un algorithme à estimation de distribution est de passer d'une population de solution très dispersée (A) à une population plus centrée sur l'optimum trouvé (B).

Il a été démontré (généralement à l'aide de modèles de Markov ou de systèmes dynamiques) que la plupart des versions pour l'optimisation combinatoire sont convergentes (c'est-à-dire qu'elles peuvent trouver l'optimum en un temps fini).

5.4 Algorithme :

Le vocabulaire lié aux algorithmes à estimation de distribution est emprunté à celui des algorithmes évolutionnaires, on parlera donc de « population d'individus » plutôt que d'« échantillon de points », ou de « fitness » plutôt que de « fonction objectif ». Néanmoins, tous ces termes ont la même signification.

L'algorithme de base procède de la manière suivante pour chaque itération :

Algorithme EDA

D0 ➡ Engendrer M individus aléatoirement

i = 0

Tant que critère d'arrêt :

i = i + 1

- ✓ D_{i-1}^{S} ➡ Sélectionner N <= M individus dans Di−1 grâce à la méthode de sélection Sélectionnés
- ✓ pi(x) = p (x| D_{i-1}^{S}) ➡ Estimer la distribution de probabilité des individus
- ✓ Di ➡ Échantillonner M individus depuis la vecteur de probabilité p(x)

Fin

5.3 Comportement :

Le comportement d'un algorithme à estimation de distribution est de passer d'une population de solution très dispersée (A) à une population plus centrée sur l'optimum trouvé (B).

Il a été démontré (généralement à l'aide de modèles de Markov ou de systèmes dynamiques) que la plupart des versions pour l'optimisation combinatoire sont convergentes (c'est-à-dire qu'elles peuvent trouver l'optimum en un temps fini).

5.4 Algorithme :

Le vocabulaire lié aux algorithmes à estimation de distribution est emprunté à celui des algorithmes évolutionnaires, on parlera donc de « population d'individus » plutôt que d'« échantillon de points », ou de « fitness » plutôt que de « fonction objectif ». Néanmoins, tous ces termes ont la même signification.

L'algorithme de base procède de la manière suivante pour chaque itération :

Algorithme EDA

D0 ⟹ Engendrer M individus aléatoirement

i = 0

Tant que critère d'arrêt :

i = i + 1

- ✓ D_{i-1}^{S} ⟹ Sélectionner N <= M individus dans Di−1 grâce à la méthode de sélection
  Sélectionnés
- ✓ pi(x) = p (x| D_{i-1}^{S}) ⟹ Estimer la distribution de probabilité des individus
- ✓ Di ⟹ Échantillonner M individus depuis la vecteur de probabilité p(x)

Fin

5.5 Avantages et inconvénients :

- Les Avantages :
    - meilleur contrôle de l'algorithme
    - moins de paramètres de réglage
    - plus de référence à la métaphore biologique de la sélection naturelle
    - réduction des besoins en mémoire
    - réduction du temps de calcul

- Les inconvénients :
    - Difficulté pour estimer les paramètres du modèle pour de grandes populations
    - Offre juste une approximation de la solution et non pas la solution exacte
    - Difficulté d'apprendre un modèle probabiliste adéquat

6 Conclusion :

Nous avons consacré ce chapitre à la présentation des méthodes de résolution des problèmes d'ordonnancement, plus précisément les métaheuristiques. Dans la première partie de ce chapitre nous avons introduit les approches métaheuristiques dans le cadre de la résolution des problèmes d'optimisation combinatoire.

La deuxième partie consistait en une présentation générale des métaheuristiques les plus connues. A la fin du chapitre, nous avons fait un état de l'art sur les approches de résolution du job shop flexible. Une attention particulière a été donnée aux approches MOAE/D et EDA que nous allons utiliser par la suite.

Le chapitre suivant décrit les techniques que nous avons mises en œuvre pour implémenter ces approches.

CHAPITRE 3

Adaptation des approches EDA et MOEA/D pour le job shop flexible

Dans le but de résoudre le problème complexe d'ordonnancement des ateliers de type Job Shop Flexible par une approche métaheuristique, nous avons choisi :

L'EDA, l'algorithme à estimation de distribution pour sa facilité de calcul d'une part et d'autre part pour qu'il soit intégré dans l'approche MOEA/D ; qui elle servira à résoudre le problème d'ordonnancement toujours, mais cette fois avec plusieurs objectifs ou autrement dit multi-objectifs, L'objectif principal de notre approche sera de réduire le temps et les ressources nécessaires à la résolution du problème job shop flexible.

La méthode de résolution que nous proposons consiste à hybrider l'EDA avec le MOEA\D afin d'obtenir un algorithme qui est simple de calcul et traite le cas multi-objectifs.

L'idée principale de notre approche est d'utiliser des probabilités pour trouver des solutions qui varient de la solution courante afin d'éviter que la recherche de solution ne reste bloquée sur un optimum local de mauvaise qualité.

Nous présentons dans ce chapitre, Les principales clés qui ont servi à la mise en œuvre des éléments nécessaires à l'implémentation de cette méthode :

- ✓ Le codage des solutions.
- ✓ l'Adaptation de l'EDA
- ✓ L'Adaptation de MOEA\D.
- ✓ Utilisation de la plateforme JADE.
- ✓

1 Codage des solutions :

Le codage choisi pour la représentation d'une solution est un codage basé sur l'affectation d'une règle de priorité à chaque machine et produit. Ces règles de priorité sont des heuristiques qui permettent à la machine de choisir l'opération à réaliser et au produit la machine sur laquelle s'exécuter.

Ce codage est de type « indirect », appelé « codage indirect basé sur les règles de priorité », consiste à établir une liste de priorités pour l'ensemble des opérations fournies. Cette liste ne décrit pas la séquence des opérations sur les machines mais l'ordre dans lequel les machines vont choisir les opérations à exécuter et les machines auxquelles les produits vont s'exécuter. Ainsi, à chaque étape, devant chaque machine et produit, il existe une file

d'opérations qui attendent l'exécution, la machine et le produit doivent alors prendre d'elles suivant leur liste de priorité.

Dans notre travail un individu est codé sur deux parties. Une partie pour coder les opérations : « Opération sélection **OS** », et un autre partie pour le codage des choix de machines pour chaque opération : « Machine sélection **MS** ».

Nous allons représenter cet individu sous forme d'un vecteur de m éléments (où m est le nombre de machines et de produit fois 2. Chaque case est codée par le numéro de la règle choisi parmi l'ensemble de règles.

Et pour faciliter plus la tâche nous avons choisi un codage binaire afin que le calcul de vecteur de probabilité soit plus facile et plus simple, dans ce cas les valeurs du vecteur représentées dans les cases varieront entre 0 et 1 (voir tableau 2).
Pour mieux comprendre on va illustrer le vecteur à m valeurs en supposant que la partie machine prend n valeurs et la partie produit prend p valeurs donc m = n+p * 2 (voir tableau 1).

Partie machine				Partie produit			
Règle N1	règle N2	. . .	Règle Nn	Règle N1	règle N2	. . .	Règle Np

Tableau 1 : représentation de vecteur de codage indirect choisit

Regle num : Un numéro varie de **1** à **4** qui correspond au numéro de la règle choisie .bien sur les règles décrites seront de type binaire, la figure suivante le montre :

01	10	00	...	01
Règle 1	Règle 2	Règle 3	...	Règle N

Tableau 2 : représentation des valeurs binaires des règles.

Les tableaux suivants énonçant les 8 règles utilisées pour les produit et machine :

1. la règle SPT (Shortest Processing Time) : la prochaine opération ordonnancée est celle dont la durée est inférieure à celles des autres opérations non encore ordonnancées.
2. la règle LPT (Longest Processing Time) : la prochaine opération ordonnancée est celle dont la durée est supérieure à celles des autres opérations non encore ordonnancées.
3. la règle SPT global : la prochaine opération ordonnancée est celle qui a le plus petit temps d'exécution de la tâche.
4. la règle FIFO (First Int First Out) : la prochaine opération ordonnancée est celle qui est la 1ére dans la liste d'attente.

Tableau 3 : Liste des règles de priorité utilisées pour la machine

1. la règle FOPNR (Fewest Number of Operations Remaining) : le plus petit nombre d'opération
2. la règle SPT global : la prochaine opération ordonnancée est celle qui a le plus petit temps d'exécution de la tâche.
3. la règle SPT (Shortest Processing Time) : la prochaine opération ordonnancée est celle dont la durée est inférieure à celles des autres opérations non encore ordonnancées.
4. la règle LPT (Longest Processing Time) : la prochaine opération ordonnancée est celle dont la durée est supérieure à celles des autres opérations non encore ordonnancées.

Tableau 4 : Liste des règles de priorité utilisées pour le produit

2 Adaptation de l'EDA pour le job shop flexible :

L'organigramme ci-dessous décrit comment nous avons pu adapter l'approche de distribution et estimation dite EDA pour notre problème :

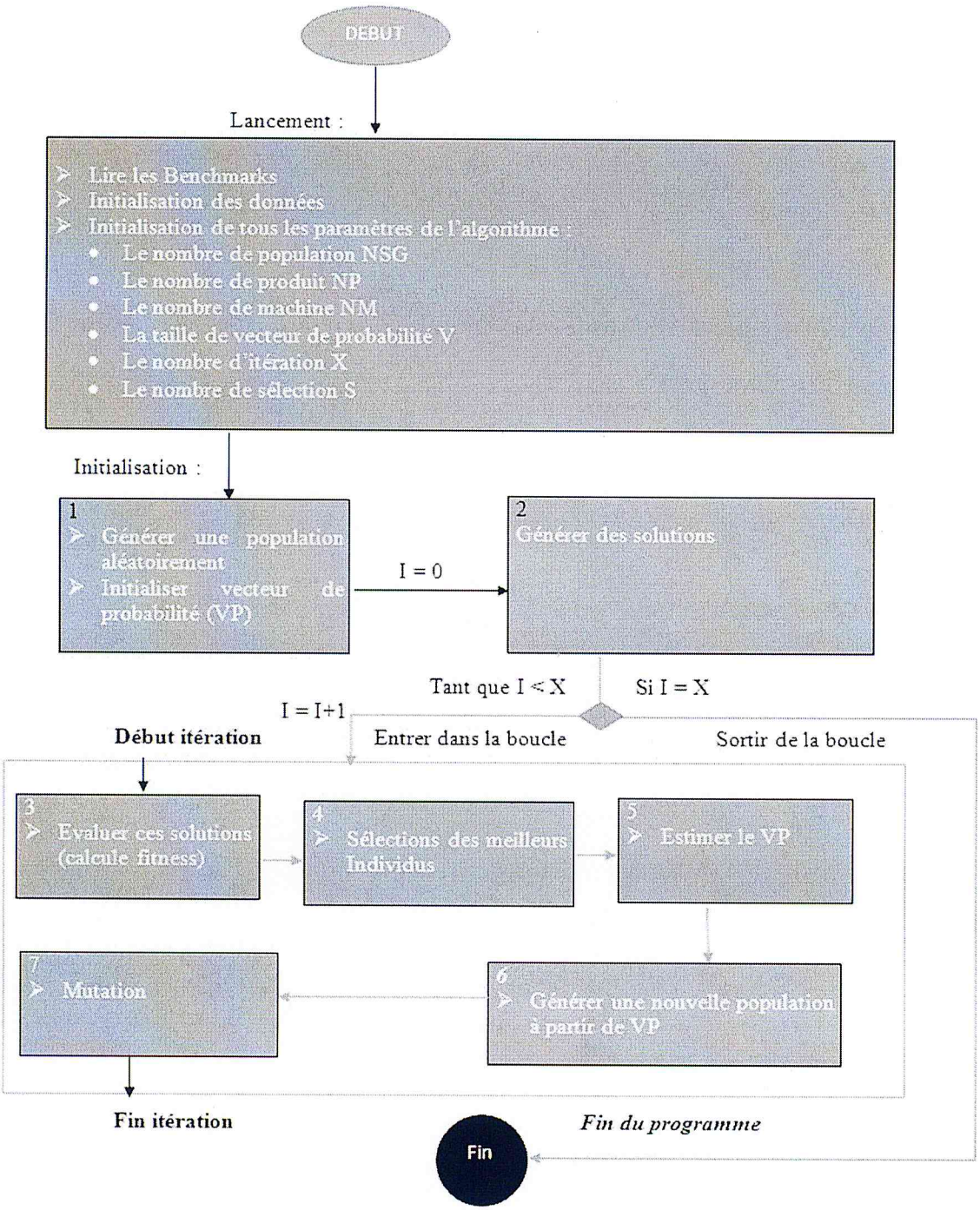

Figure 7 : organigramme de l'algorithme EDA

Remarque

On Remarque dans l'organigramme l'apparition de l'étape mutation or que dans les EDA la notation mutation et croisement n'existent pas.

Pourquoi alors utiliser la mutation ?

Enfaite, les métaheuristiques utilisent l'exploitation et l'exploration afin de trouver l'optimum, dans notre cas l'algorithme à estimation de distribution se contente de l'exploitation seulement, ce qui le rend convergent et qui bloque dans un ensemble de solution qui peut être très loin de l'optimum.

Donc, pour éviter ce blocage nous avons intégré la notion de mutation afin que notre algorithme utilise l'exploitation et l'exploration en même temps.

2.1 Les paramètres de l'algorithme :

- ***Taille de population :***

Ceci est un paramètre clé dans l'algorithme car il représente la taille de population ou autrement dit le nombre d'individus qu'on aura à évaluer et à traiter comme solution.

- ***Nombre de produit :***

C'est l'indice qui définit combien on aura de produit à traiter et à affecter dans des machines.

- ***Nombre de machine :***

Il définit le nombre de machine qu'on a et à combien de machines on peut affecter les produits.

- ***Taille de vecteur de probabilité :***

Elle désigne la taille que devra prendre le vecteur de probabilité, elle prend comme valeur la somme multiplie fois 2 du nombre de machines et produits afin de créer le vecteur de probabilité qui contiendra toutes les informations de cette population.

- ***Nombre d'itération :***

Désigne le nombre de fois qu'on aura à faire boucler notre algorithme, et le nombre final désigne le test d'arrêt de notre algorithme.

- ***Nombre de sélection :***

Désigne le nombre des meilleurs individus qu'on aura à sélectionner parmi toute la population, ou littéralement dit le paramètre de sélection.

2.2 Les étapes de l'algorithme :

2.2.1 Générer une population aléatoire :

Cette étape a pour but d'initialiser la population du départ en prenant compte de la taille déclarée auparavant, elle génère toute une population de m vecteurs (individus) contenant des valeurs binaires tirées au hasard soit 0 soit 1.

Cette étape ne sera plus utilisée à l'avenir vu que les prochaines populations générées ne seront plus aléatoires mais générées à partir du VP (vecteur de probabilité).

La figure suivante représente un exemple de population.

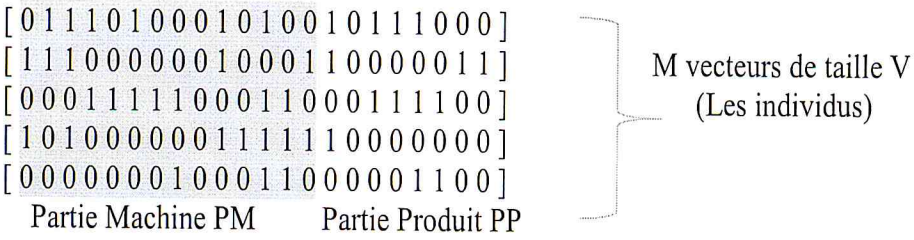

Figure 8 : Exemple de population

2.2.2 Initialiser le vecteur de probabilité :

Cette étape consiste à créer le vecteur de probabilité qui contiendra toutes les informations sur la population, en prenant compte de la taille calculée auparavant, le VP se calcule d'une manière assez simple :

Une fois la population initialisée on génère le VP avec comme valeurs 0.5 (probabilité d'avoir soit 1 soit 0) de ce fait, Le VP crée contient uniquement la valeur 0.5, oui vous l'aurez surement remarqué les valeurs que contient le VP sont des réels et non pas des entiers, on reviendra sur ça et sur son utilité plus tard.

La figure suivante représente cette étape :

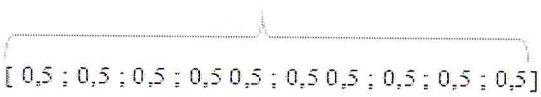

Figure 9 : vecteur initiale de probabilité

2.2.3 Générer une solution :

Cette étape est cruciale dans notre algorithme car elle permet de générer des solutions à partir de la population crée.

Son principe est de convertir les vecteurs de la population représentée sous forme binaire en des vecteurs avec des valeurs décimales afin qu'ils soient envoyés à la fonction d'évaluation qui va poursuivre la prochaine étape.

[0 1 1 1 0 1 0 0 0 1 0 1 0 0 1 0 1 1 1 0] ⟹ [2 4 2 1 2 2 2 1 3 4 3]

Figure 10 : La conversion d'un vecteur de population

2.2.4 Evaluer une solution :

Comme son nom l'indique le but principal de cette étape est d'évaluer les solutions qu'on lui aura proposées, les résultats obtenus à partir de cette étape déterminent les meilleurs individus parmi la population testée.

L'évaluation se fait en testant les solutions proposée (les propriétés) sur des benchmark ou autre exemple qui traite le problème d'ordonnancement, et qui nous donnera comme résultats le temps, la charge et le poids total de chaque affectation avec chaque solution proposé

Et pour évaluer une solution on se base sur notre codage qui représente les règles des machines et des produits à choisir et donc la fonction se base sur ces règles de priorité pour effectuer l'affectation de tous les produits et machines et une fois cette affectation terminée la fonction renvoie avec le temps la charge et le poids que ça a couté chaque affectation, et donc après l'obtention des résultats on passe à l'étape suivante.

2.2.5 Sélection des meilleurs individus :

Une fois les résultats de l'évaluation obtenus, le rôle de cette étape devient alors crucial car on va traiter ces résultats et classer les individus selon leurs fitness, du coup on aura toute une liste d'individus classés du meilleur au plus mauvais, il nous reste alors qu'à sélectionner les meilleurs (avec la valeur de sélection déclarée auparavant), ces individus vont servir à estimer le nouveau vecteur de probabilité.

L'organigramme suivant montre un exemple de ça :

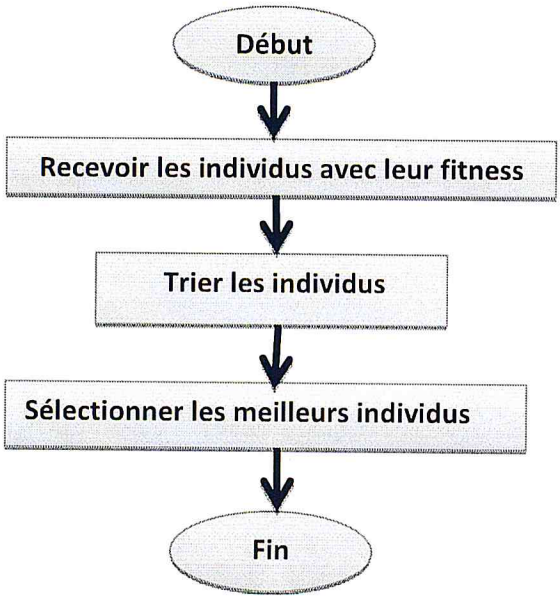

Figure 11 : l'organigramme de la sélection

2.2.6 Estimer le vecteur de probabilité (VP) :

Cette étape consiste à estimer un nouveau VP à partir des individus sélectionnés, son calcul se base sur la probabilité d'apparition de l'individu, il se fait de la manière suivante :

On parcourt chaque premier élément de chaque individu (vecteur) et à chaque fois qu'on croise un 1 alors on additionne et au final on divise le total sur le nombre d'individus c.-à-d. la taille de la population et on ajoute la valeur obtenue dans la 1er case du VP et on continue comme ça avec le reste jusqu'à ce que tous les vecteurs seraient parcourus.

La figure suivant représente un exemple de calcul de vecteur de probabilité.

[0 1 1 1 0 1 0 1]
[1 1 1 0 0 0 0 0] ⟹ [0,33 ; 0,66 ; 0,66 ; 0,66 ; 0 ,33 ; 0 ,66 ; 0 ,0 ; 0 ,66]
[0 0 0 1 1 1 0 1]
1 /3

Figure 12 : calcul de la probabilité

2.2.7 Générer une nouvelle population :

Cette étape va nous permettre de créer une nouvelle population générée à partir du vecteur de probabilité estimé auparavant, lequel contient les informations sur l'ancienne population plus précisément sur les meilleurs individus de l'ancienne population.la figure suivante représente l'organigramme de cette étape :

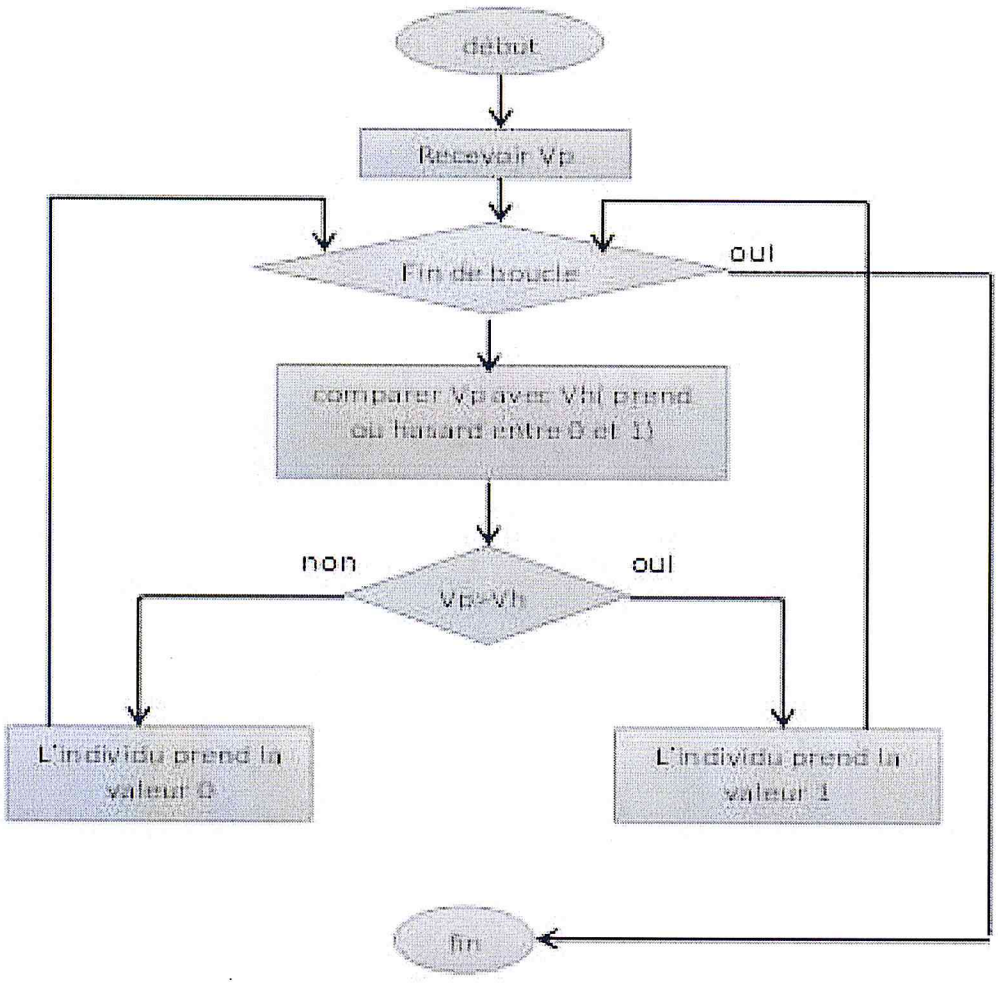

Figure 13 : l'organigramme de générer une population à partir de VP

2.2.8 La mutation :

Son but sert à éviter que le programme converge vers une solution non optimale et qu'il bloque dedans, du coup on utilise la notion de mutation pour diversifier l'exploitation.

Cette mutation se fait avec une valeur de convergence qui a tendance à se rapprocher de celle de l'individu le plus adapté, et dépendamment de cette valeur on inverse un individu par un autre.

Une fois mutée on revient à l'étape qui génère des solutions.

L'organigramme suivant représente la partie mutation avec un taux TM et un nombre de mutation nb.

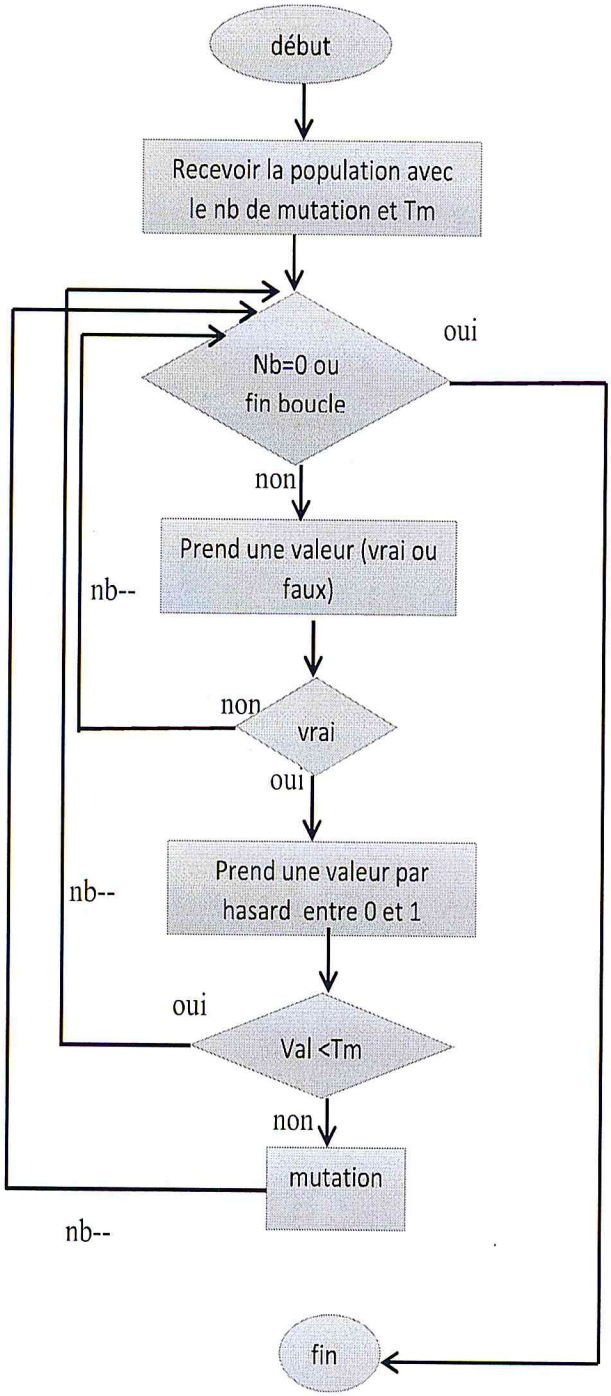

Figure 14 : l'organigramme de mutation

2.3 Les objectifs :

2.3.1 Cmax (bestfit) :

Le Cmax, appelé aussi Makespan, est la durée totale de l'ordonnancement. C'est une valeur détermine le temps que l'affection a pris avec l'algorithme utilisée. Bien sur cette valeur ne peut pas être toujours idéale, tout dépend du déroulement de l'algorithme, mais en général elle se rapproche beaucoup de l'optimum.

Dans chaque itération si une nouvelle fitness obtenue est inférieure à la précédente, alors le Cmax prend cette nouvelle valeur comme valeur, et ça continue comme ça tout au long de la boucle, une fois la boucle terminée, la dernière valeur obtenue sera alors l'optimum vu que c'est le meilleur fitness obtenu avec notre algorithme.

La durée totale de l'ordonnancement est égale à la différence entre la date d'achèvement de la tâche la plus tardive et la date de départ de la première tâche.

La minimisation de cette durée est le critère le plus souvent rencontré puisque ça conduit inévitablement à une utilisation efficace des ressources.

2.3.2 La charge critique :

Le but de cet objectif est de calculer la charge des machines qui réalisent la solution optimale, la somme de ces charges constitue la charge critique, autrement dit c'est le poids minimal du meilleur chemin à suivre pour atteindre le meilleur fitness.

2.3.3 La charge totale :

Le but de cet objectif est de calculer la somme non pas de la charge critique seulement mais les charges critiques de tous les individus, la charge minimale obtenue dans chaque itération représente la meilleure charge critique obtenue.

2.4 Représentation des résultats :

La présentation la plus courante pour l'ordonnancement est le diagramme de Gant celui_ci représente une opération par un segment ou une barre horizontale, L'axe des abscisses représente le temps et l'axe des ordonnées représente l'ensemble des machines. Pour chaque machine, on représente la séquence de tâches effectuées dans le temps.

L'organigramme suivant représente un exemple de représentation d'ordonnancement avec diagramme de Gantt :

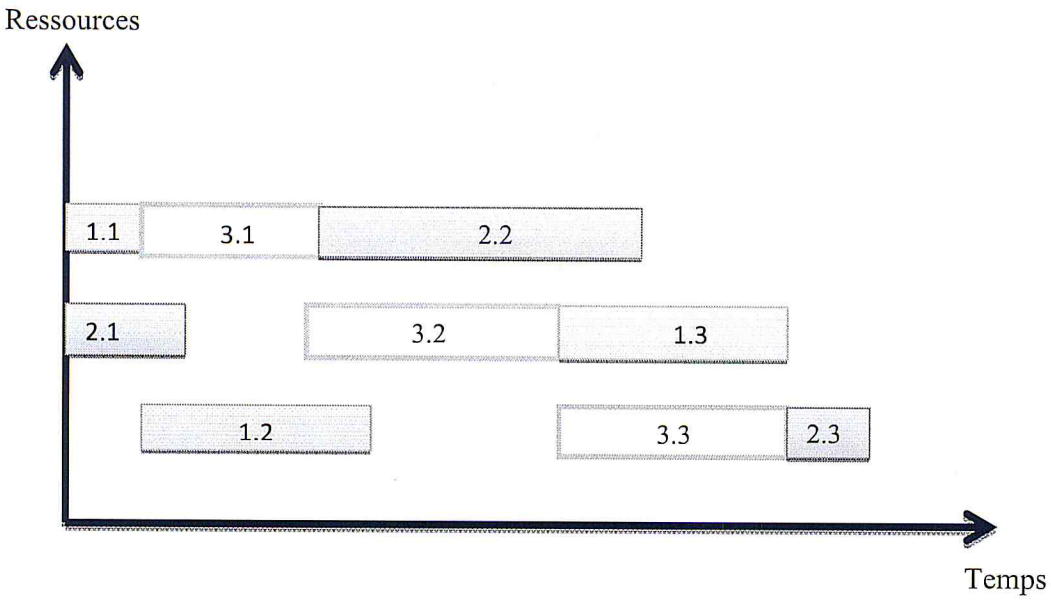

Figure 15 : Exemple d'une représentation d'une solution sous forme de diagramme de Gantt.

3 Adaptation de l'MOEA/D :

Contrairement au mono-objectif dans le cas multi-objectifs le but est non pas de trouver une solution optimale mais tout un ensemble de solution qui satisfait les contraintes qu'on souhaite trouver.

L'approche MOEA\D utilise le parallélisme distribué ce qui nous a permis d'introduire l'EDA qui est une approche de distribution avec une petite modification dans la partie d'évaluation et un nouveau paramètre appelé vecteur de poids (pondération) qui servira dans la parallélisation.

L'organigramme suivant représente comment nous avons procéder afin de réaliser cette approche :

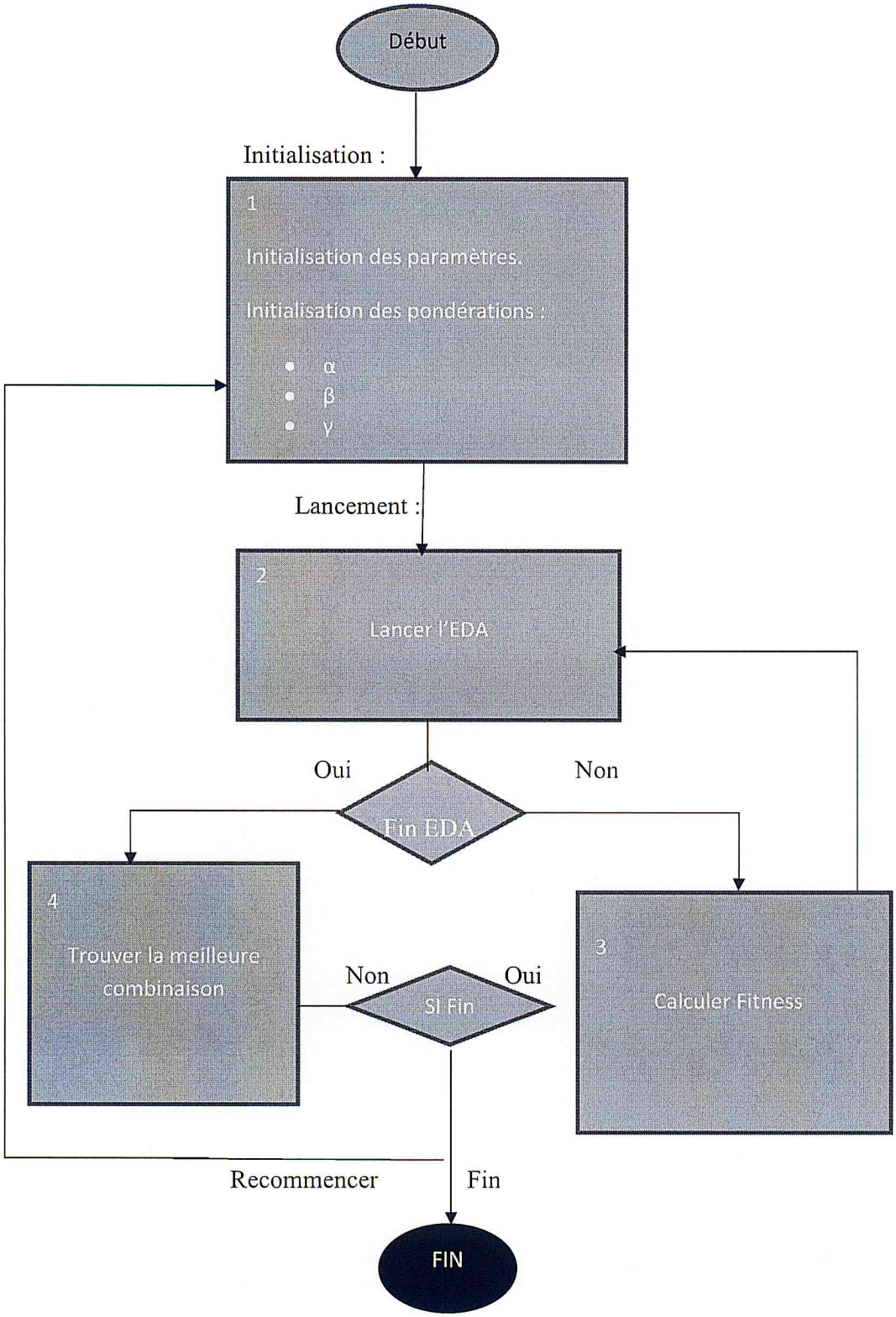

Figure 16 : l'organigramme de MOAE/D

3.1 Les pondérations :

C'est un vecteur de poids de la taille des objectifs, ces valeurs sont des variables de petite taille avec comme somme la valeur 1, elles seront utilisées comme contraintes afin de vérifier à la fin si les solutions trouvées font partie de la dominance de Pareto et si elles atteignent le front de Pareto ou non.

Exemple : voilà une illustration d'un vecteur de poids avec 3 objectifs

On suppose qu'on a 3 objectifs f1, f2 et f3, la taille du vecteur dépend de la taille de ces objectifs

α	β	γ
0.5	0.33	0.17

Telle que α est la pondération de l'objectif f1, β pour f2 et γ pour f 3

Et **($\sum (\alpha + \beta + \gamma) = 1$) .**

3.2 Les étapes :

- ***Initialisation :***

Le but de cette étape est de déclarer les paramètres généraux de l'algorithme entre autres les trois pondérations avec lesquels nous traiterons nos trois objectifs

Une fois ces pondérations déclarées on passe à l'étape suivante.

- ***Lancer EDA :***

Dans cette étape nous allons lancer notre algorithme EDA avec une fonction évaluation modifiée pour nous renvoyer cette fois 'Cmax, Charge critique et Charge totale' qui serviront dans le calcul du meilleur fitness dans l'étape suivante.

- ***Calculer fitness :***

Cette étape est cruciale car c'est dans cette dernière qu'on va utiliser la notion de multi-objectifs, comme on le sait, le but est de trouver la meilleure combinaison entre les différents objectifs qu'on a, ce qui nous pousse alors à calculer un fitness représentant la meilleure combinaison entre les objectifs de chaque individu, le calcul de ce fitness se base sur l'approche de Tchebycheff (te) et se calcul de la manière suivante :

Le min (de max entre $\sum$ pondération * | objectifs – point idéal z |).

C.-à-d. dans notre cas :

Min {max ((α * | Cmax – z1 |), (β * | charge critique – z2 |) , (γ * | charge total – z3 |) }

La valeur calculée représente la meilleure fitness de la population, une fois calculée on se base sur cette valeur pour ordonner et sélectionner les meilleurs individus de cette population.

- ***Trouver la meilleure combinaison :***

Une fois la fitness calculée, les individus sélectionnés, le programme continue son déroulement jusqu'à la fin de l'itération ou on stock la meilleure combinaison trouvée, faire ceci pour toutes les itérations jusqu'à l'arrivée du test d'arrêt (fin de l'EDA) là où on tire la meilleure combinaison possible pour les 3 objectifs parmi ceux de chaque itération.

La combinaison finale représente la solution trouvée pour la pondération qu'on a fixée dans ce cas ; Il ne reste plus qu'à interpréter cette solution.

4 Utilisation de la plateforme JADE :

Comme on l'a vu précédemment, le MOEA\D est une approche multi objectifs basé sur la parallélisation, cette parallélisation se fait par l'affectation d'un vecteur de poids pour l'approche qui serait utilisée comme indice dans l'espace de recherche dans lequel l'algorithme va être orienté.

Pour y arriver nous avons adapté le MOEA\D avec un seul vecteur de poids et donc un seul espace de recherche, et avec l'utilisation de la plateforme JADE multi-agents nous avons intégré l'approche MOEA\D à plusieurs agents avec des pondérations différentes pour diversifier l'espace de recherche d'une part, s'échanger des messages durant les itérations d'autre part, et pour finir d'interpréter les résultats obtenus pour vérifier si les résultats réalisent le front de Pareto ou non.

L'organigramme qui va suivre résume la méthode adaptée pour la plateforme multi agents :

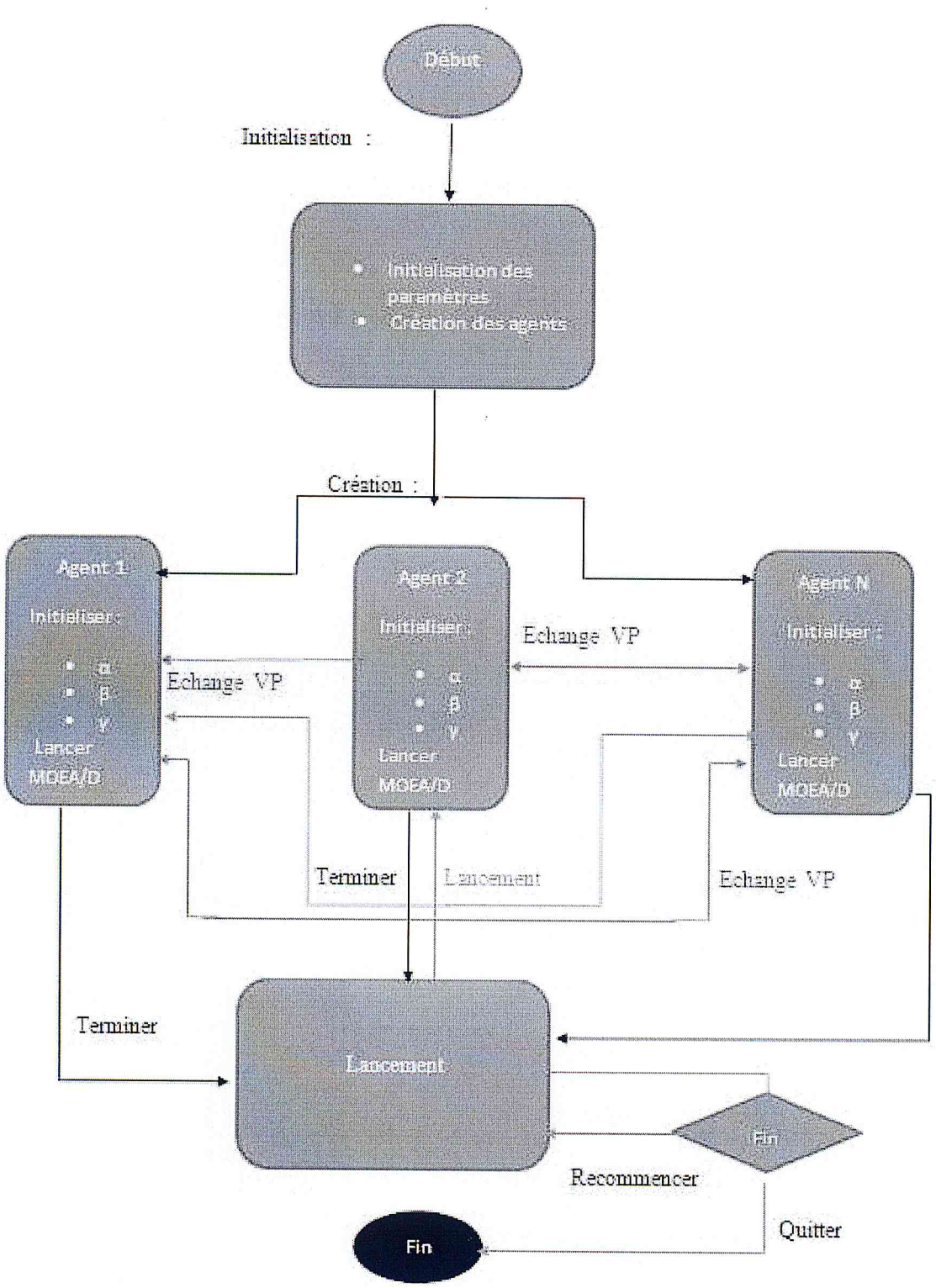

Figure 17 : organigramme de la plateforme.

Les Paramètres :

Les paramètres de cet algorithme ne sont que les anciens paramètres généraux de l'EDA et de MOEA/D récupérés et utilisés dans cette plateforme multi-agents.

4.1 Les agents :

Du point de vue de la programmation concurrente, un agent est un objet actif, ayant un thread de contrôle. Ainsi que JADE utilise un modèle de programmation concurrente "un thread-par-agent" au lieu d'un modèle "un thread-par-comportement" pour éviter une augmentation du nombre de threads d'exécution exigés sur la plate-forme d'agents.

Les agents JADE sont actifs (ils peuvent dire Non et ils sont faiblement couplés), communiquent et interagissent avec les autres agents grâce à des messages et rendent des services.

D'un point de vue plus technique, Jade est orienté vers une programmation concurrente où chaque agent est en « compétition ». Les concepteurs de Jade ont ainsi fait le choix de créer un thread (code de parallélisassions) par agent plutôt qu'un thread par comportement. Néanmoins, via le planificateur de tâches, un comportement peut se bloquer lui-même pour éviter de gaspiller du CPU (Ex : pendant qu'il attend des messages).

4.2 Les étapes :

4.2.1 Initialisation :

Cette étape consiste à :

- Initialiser les paramètres :

  Ce but est d'initialiser tous les paramètres dont l'algorithme aura besoin d'utiliser dans son déroulement.

4.2.2 Création des agents :

Une fois les agents crées, chacun aura à sa disposition son propre MOEA\D avec sces propres pondérations et qui attend un signal de lancement.

Et vu qu'on va utiliser des échanges de messages entre les voisins les plus proches (entre agent et agent) la communication doit être synchronisée entre eux, le lancement de ces agents doit être en même temps afin d'éviter qu'un agent soit décalé par rapport à un autre.

4.2.3 Lancement :

Voilà une fois lancés, chaque agent déroule son MOEA/D vu que les pondérations sont différentes, les résultats seront de même, pour mieux gérer et avoir de meilleurs résultats une communication entre ces agents est nécessaire (dans le schéma s'a fait référence à échange VP) cette communication aura pour but de s'échanger le vecteur de probabilité entre chaque agent durant le déroulement de chacun avec un nombre précis d'itérations pour s'envoyer ces VP.

4.2.4 L'échange des messages :

L'échange des vecteurs de probabilité va servir à améliorer de plus en plus la population à générer, dans son contexte le nouveau vecteur de probabilité reçu servira à générer une petite partie de la population et le reste sera générer par le VP source . Pour réaliser tout ça nous avons transformé le VP en messages afin qu'il soit envoyé et reçu par les autres agents.

L'échange de messages se fait à un nombre précis d'itérations, et le choix du voisinage revient à choisir le plus proche voisin, cette méthode a était calculée par la distance euclidienne des pondérations entre chaque agent, les agents les plus proches entre eux sont considérés comme des voisins et l'envoi et la réception des messages entre tous les agents constitue un automate. L'organigramme qui va suivre représente un échange de messages entre l'agent 1 et l'agent 2 à l'itération = 100 :

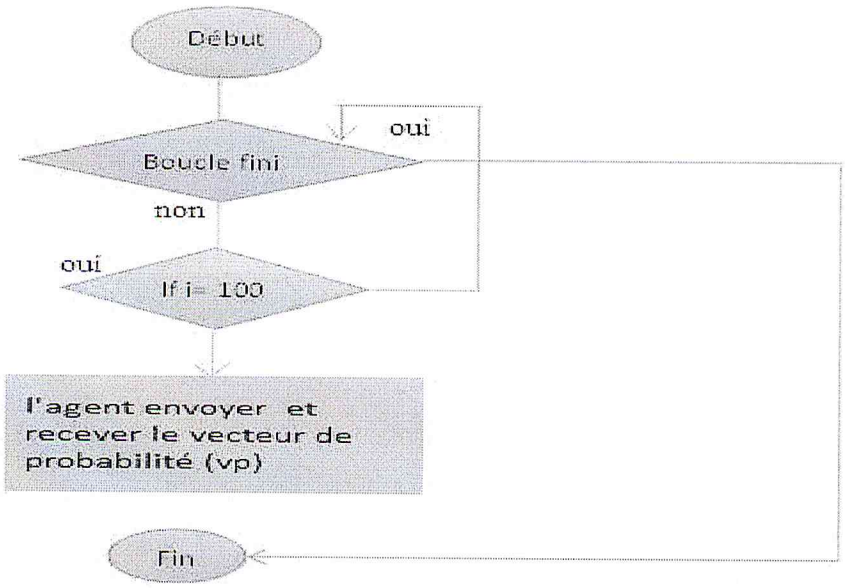

Figure 18 : échange du message du vecteur de probabilité

Une fois les messages reçus, la génération générée à nouveau, l'algorithme continue son déroulement en revenant à l'étape mutation et recommence à nouveau tout au long de la boucle.

Le critère d'arrêt est dans notre cas la fin de la boucle d'exécution de l'algorithme MOEA.D définit tout au début comme paramètre.

Voilà, le programme finit, il nous reste qu'à collecter les résultats obtenus, les étudier, les comparer avec des résultats d'autres méthodes et tirer une conclusion.

5 Conclusion :

Dans ce chapitre, nous avons développé une métaheuristique basée sur l'hybridation des algorithmes EDA et MOEA\D pour résoudre le problème étudié. Les différentes étapes de l'application de cette approche (le codage utilisé, l'adaptation de l'EDA, l'adaptation de MOEA\D,…) ont été présentées dans ce chapitre.

Maintenant vient l'implémentation de cette dernière, le dernier chapitre illustre cette implémentation, et donne une description du logiciel développé, ainsi que les résultats obtenus, les comparaisons de ses résultats et une conclusion de notre travail, c'est ce qu'on va aborder dans le prochain chapitre.

CHAPITRE 4

Implémentation Test et Résultat

L'objectif de ce présent chapitre est de donner une synthèse des résultats obtenus par notre approche à la résolution d'un ensemble de problèmes tests « benchmarks » de Job Shop Flexible.

Ce chapitre est divisé en trois parties. La première est consacrée à la définition des outils de développement utilisés pour l'implémentation de notre méthode et à la présentation de l'application. L'objectif de la deuxième partie est de donner le réglage optimum des paramètres d'EDA. La troisième partie présente des expérimentations numériques visant à illustrer et comparer la méthode implémentée.

1 Le langage de programmation choisi :

Pour la réalisation de notre projet nous avons utilisé le langage de programmation Java(SUN), sous l'éditeur Eclipse (https://eclipse.org).

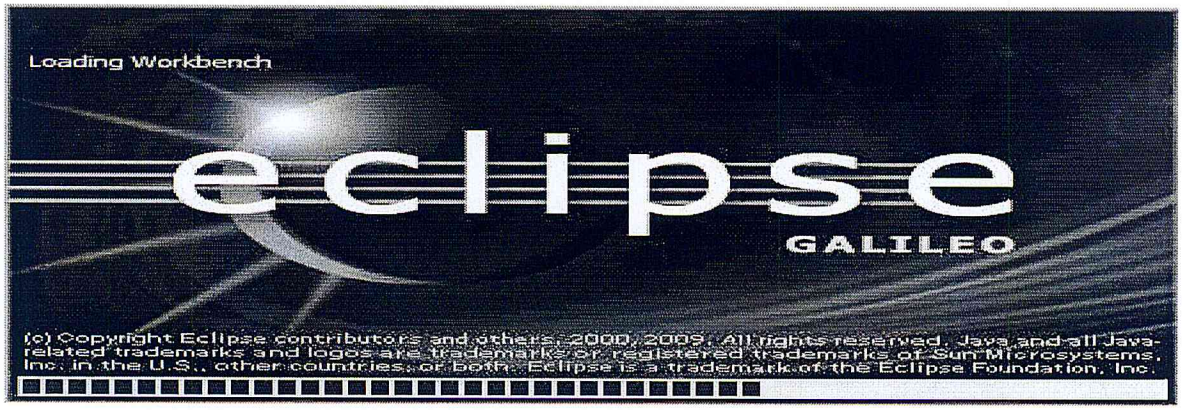

Figure 19 : logo d'Eclipse

Nous avons choisi le langage JAVA pour les raisons suivantes :

- Nous sommes bien familiarisés avec le langage JAVA ;
- L'application peut s'exécuter sur n'importe quel système d'exploitation à condition d'avoir la machine virtuelle java installée sur la machine (portabilité).
- La disponibilité de la documentation et de l'assistance (forums).
- JAVA est un langage de programmation moderne développé par Sun Microsystems ; c'est un langage de programmation orienté objet basé sur le langage C++ mais avec des fonctionnalités qui rendent la programmation plus simple et plus sûre. Il bénéficie d'une grande bibliothèque qui met à la disposition du développeur plusieurs paquetages prêts à l'utilisation, nous citons parmi les principaux qu'on a utilisés :

- **La bibliothèque Swing :**

Cette librairie offre un ensemble de composants, graphiques pour la construction des interfaces graphiques ; parmi ces composants on a utilisé : JFrame, JPanel, JDialog, JButton, JTabbedPane, …

- **Java2D :**

C'est une librairie qui offre un ensemble d'outils de dessin pour implémenter des graphiques qu'on a utilisés spécialement pour le dessin des diagrammes.

- **Java.io :**

Comme tout langage de programmation, java fournit cette bibliothèque de manipulation des flux de données que nous avons utilisée pour manipuler des fichiers Texte (création, lecture, écriture).

- **Java. util :**

Nous avons utilisé cette bibliothèque pour manipuler les structures de données (les vecteurs) et certaines fonctions mathématiques (random (), sqrt (), abs(), …)

- **la plateforme multi agents (jade) :**

Nous avons utilisé cette plateforme pour créer les agents pour utiliser le concept de parallélisme.

2 Présentation de l'algorithme :

2.1 Interface principale :

Lors du lancement du programme, l'interface principale apparaît, portant le thème de notre projet, nos noms respectifs, ainsi qu'un bouton de lancement de l'application dans le cas de l'EDA ou MOEA\D, et un bouton de lancement des agents dans le cas de la plateforme JADE.

Les figures qui vont suivre représentent l'interface utilisée pour l'EDA, MOEA\D et les Agents. Elles contiennent les boutons suivants :

❖ **Bouton Lancer L'EDA** :

Ce bouton sert à lancer notre algorithme EDA

❖ **Bouton Lancer L'MOEA\D** :

Ce bouton sert à lancer notre algorithme MOEA\D

❖ **Bouton Lancer les Agents** :

Ce bouton sert à lancer simultanément tous les agents de la plateforme, afin que chaque agent déroule son MOEA.D

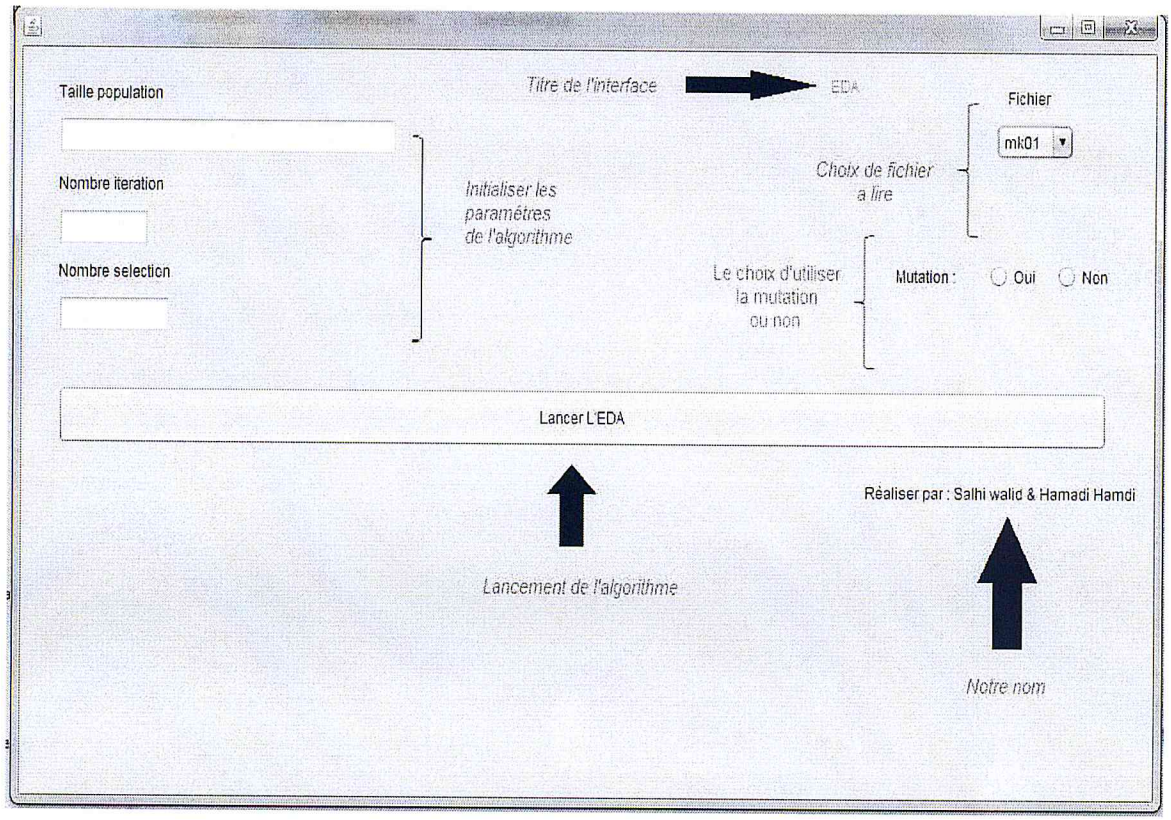

Figure 20 : interface principale d'EDA

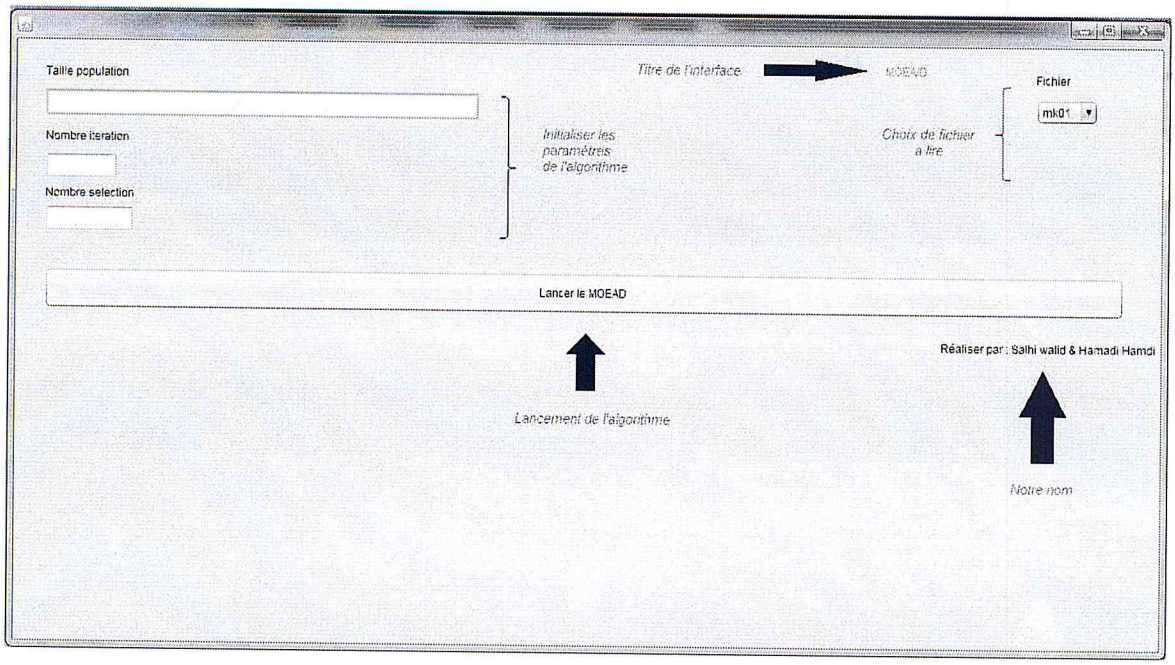

Figure 21 : interface principale de MOEA/D

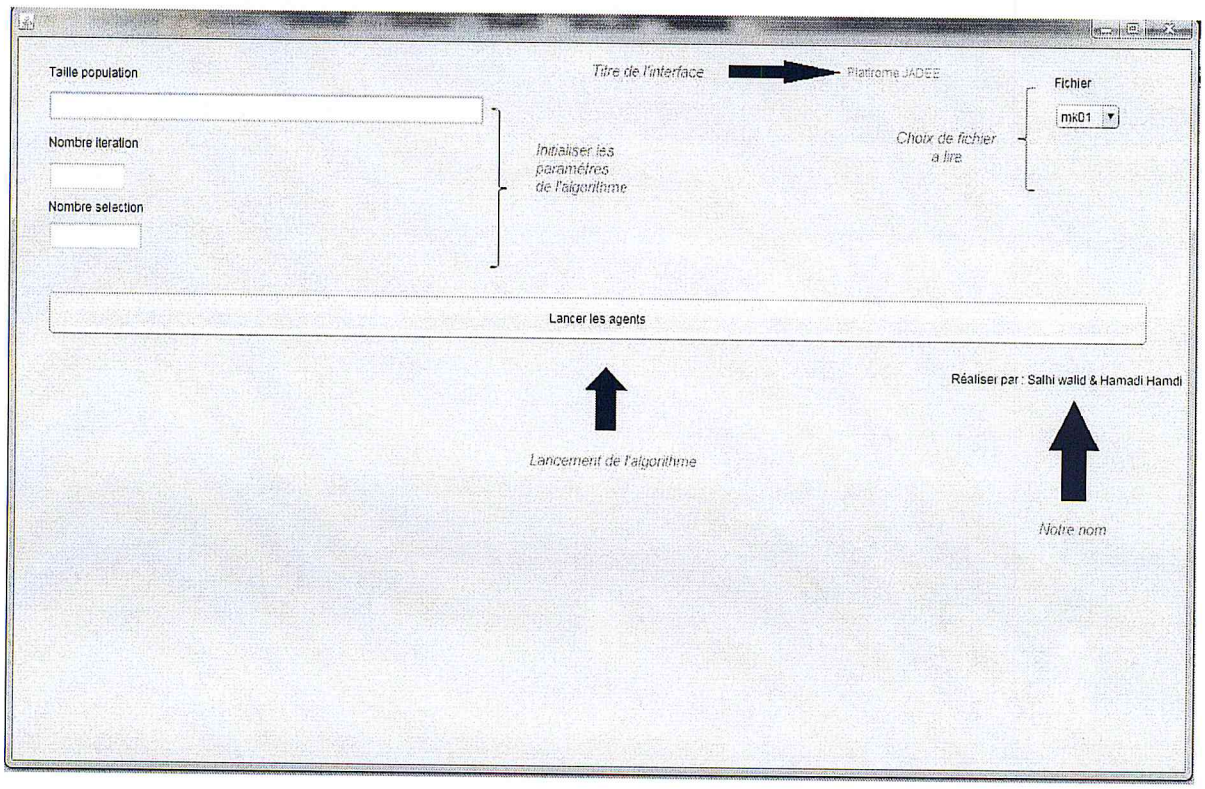

Figure 22 : interface de lancement des agents

2.1.1 Les paramètres :

- **Taille population :**

Nous permet d'indiquer la taille du problème à traiter en définissant la taille de la population et le nombre d'individus.

- **Nombre itération :**

Nous permet d'indiquer le nombre d'itérations que l'algorithme aura à dérouler.

- **Nombre sélection :**

Nous permet d'indiquer le nombre d'individus qu'on aura à sélectionner pour l'estimation et la généralisation d'un nouveau vecteur de probabilité

- **Fichier :**

Nous permet de choisir avec quel fichier Benchmarks ou autre avec lequel on va tester notre algorithme

- **Mutation :**

Cette case nous permet de choisir si oui ou non on veut utiliser la mutation dans notre algorithme EDA.

Dans le tableau qui suit l'ensemble des paramètres utilisés pour le programme avec quelques captures d'écran des résultats et des déroulements :

Paramètre	Définition	type	Valeur
NSG	Taille de la population	Int	Définit par l'utilisateur
I	Indice itération	Int	Dynamique
S	Nombre de sélection	Int	Définit par l'utilisateur
X	Nombre d'itérations	Int	Définit par l'utilisateur
TM	Taux de mutation	Float	0.2
NBRM	Nombre d'individu a muté	Int	2
muter	Indice de mutation	Boolean	Définit par l'utilisateur
Fichier	Le fichier à lire	String	Définit par l'utilisateur
N_P	Nombre de produit	Int	dynamique

N_M	Nombre de machine	Int	Dynamique
V	Taille de l'individu	Int	N_P + N_M x 2
Cmax	le Makespan	Int	Dynamique
Charge	La charge critique	Int	Dynamique
Total	La charge totale	Int	Dynamique
bestind	le meilleur individu	Vector	Dynamique

Tableau 5 : les paramètres utilisés

2.2 Aperçu du programme :

- ***Fenêtre des résultats :***

La figure suivante montre la fenêtre d'un résultat de l'EDA :

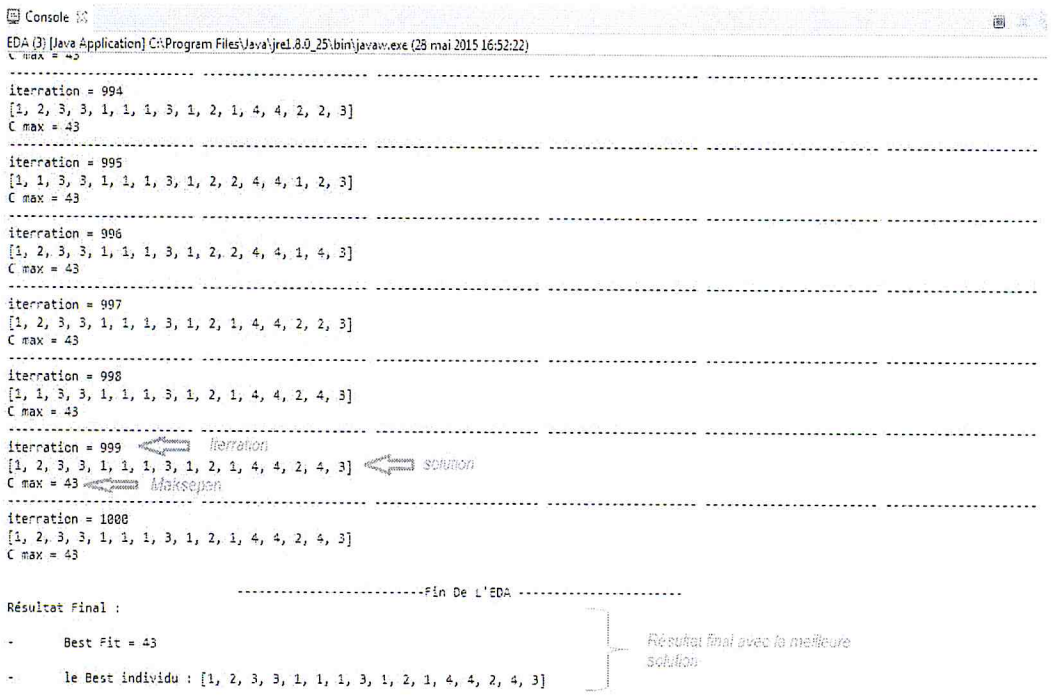

Figure 23 : la fenêtre de résultat d'EDA

La figure suivante montre la fenêtre d'un résultat du MOEA\D :

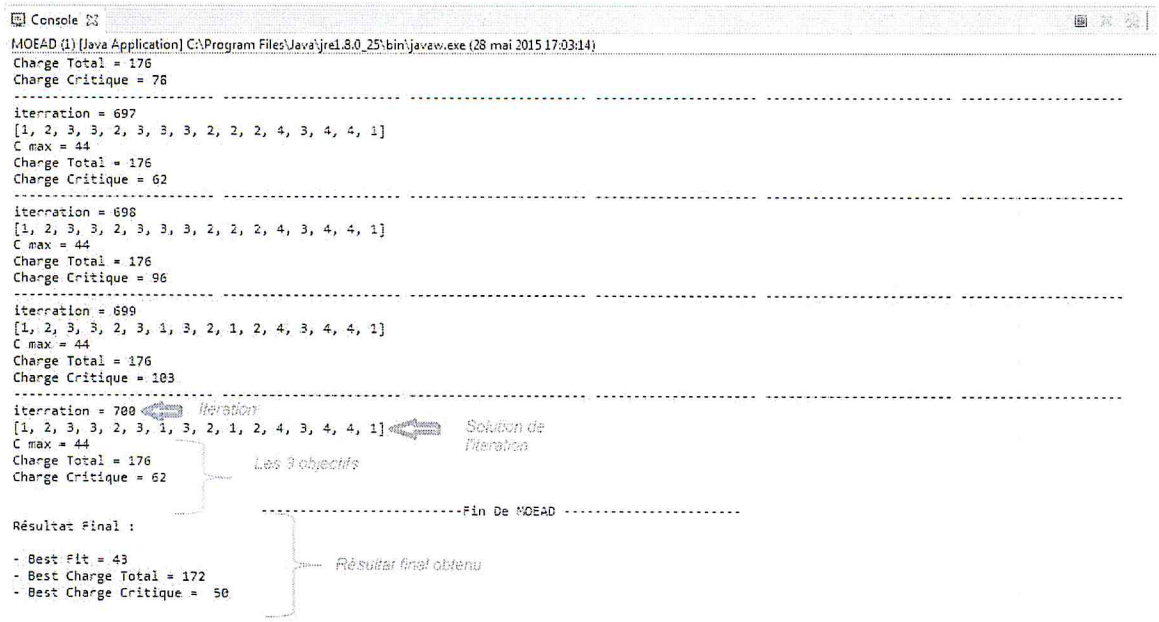

Figure 24 : la fenêtre de résultat de MOEA/D

Interprétation des résultats par un diagramme de Gantt pour L'EDA :

La figure suivante montre le résultat obtenu ou bien la solution trouvée par un diagramme de Gantt :

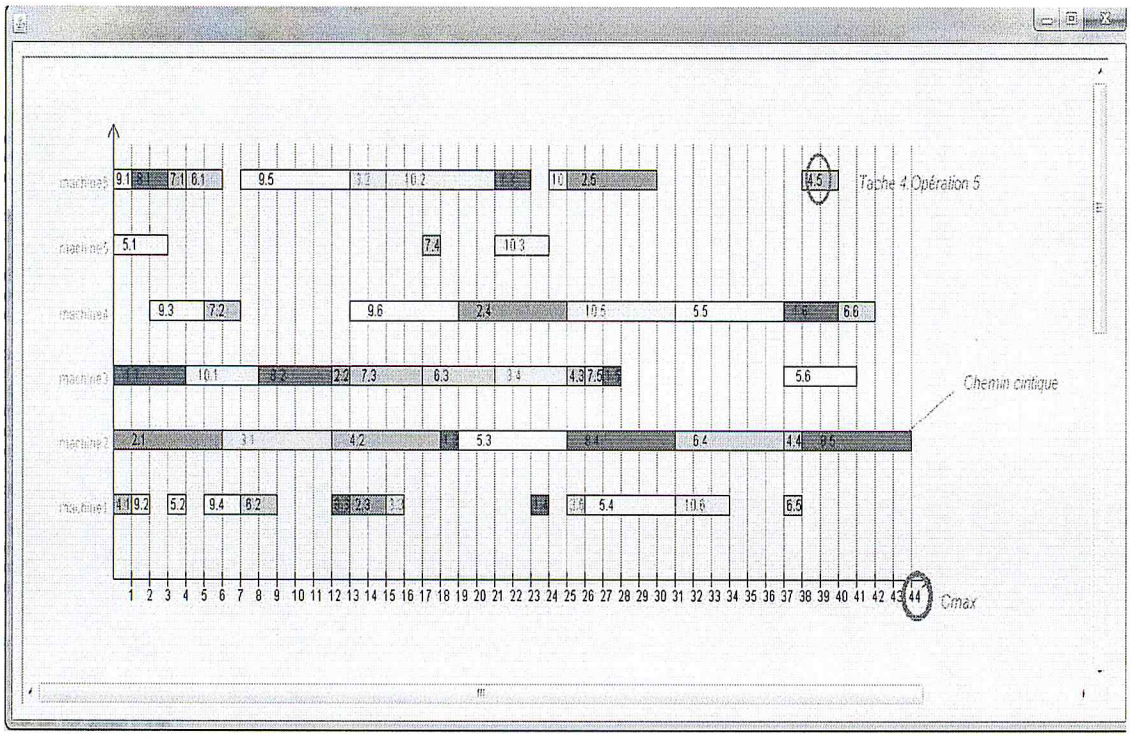

Figure 25 : la représentation de la solution trouvée par un diagramme de Gantt

- **La population de départ :**

La figure suivante est un exemple de population de départ tirée aléatoirement

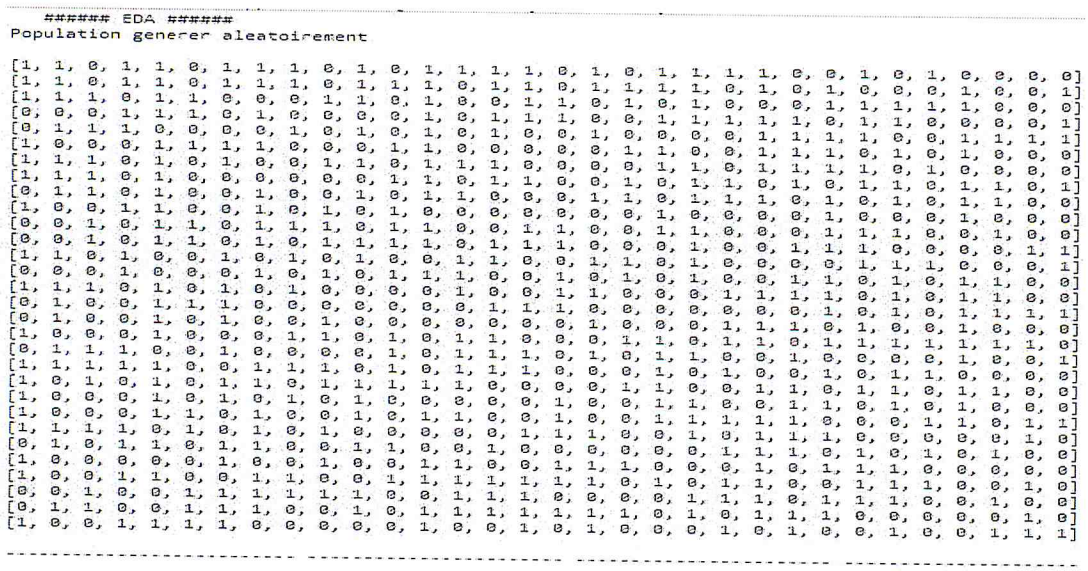

Figure 26 : un exemple de population de 30 individus

3 Validation de l'approche :

3.1 Les benchmarks :

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

Les benchmarks de Job Shop Flexible présentent les caractéristiques suivantes :

- Chaque benchmark est décrit par un ensemble de **n** tâches et **m** machines. Ce sont des problèmes **n** x **m**
- Chaque tâche comprend un certain nombre d'opérations qui dépend de la tache.
- Chaque opération à son ensemble de choix de machine.
- Les durées opératoires des opérations sont comprises entre 1 et 100.
- A part le temps opératoire, il n'existe pas d'autres contraintes temporelles (telles que la date d'échéance, la durée de préparation, la date de disponibilité).
- Les solutions optimales de ces benchmarks sont connues ce qui constitue un bon outil pour les expérimentations et les comparaisons.

La figure suivante montre un exemple de benchmarkMK01 sur laquelle nous constatons les caractéristiques générales des benchmarks :

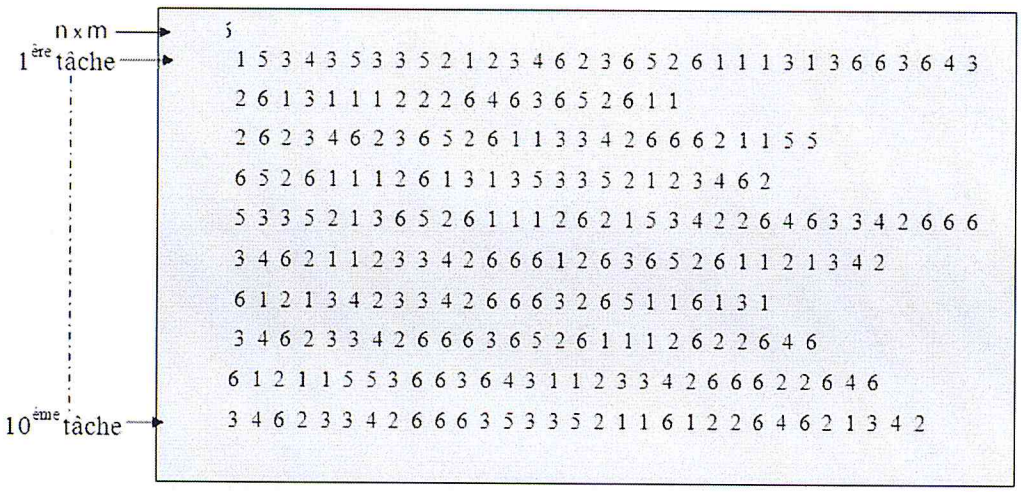

Figure 27 : Exemple d'interprétation de MK01

Exemple d'interprétation de la tache 2 :

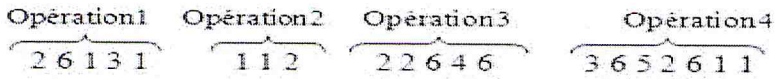

Figure 28 : interprétation de la tache 2.

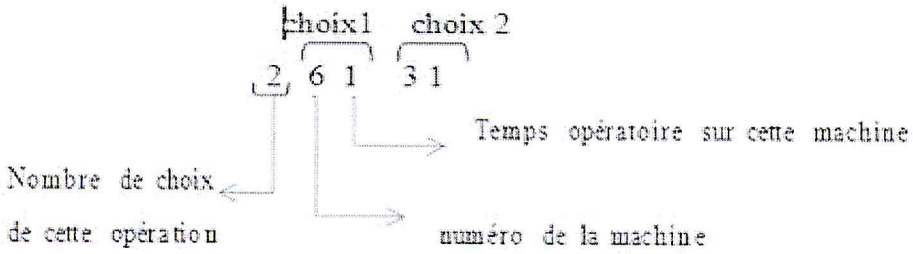

Figure 29 : interprétation deopération1 de la tache 2.

Dans notre étude, nous utilisons un échantillon de ces benchmarkscommedes problèmes tests pour la résolution du problème de Job Shop Flexible.

3.2 EDA :

Pour valider notre EDA, nous avons tout d'abord confirmé l'utilité de l'étape de mutation en faisant une série de tests sur les 10 Benchmarks avec comme paramètres : Population = 30, itérations = 500 et 4 sélection.

Le tableau ci-dessous montre les résultats obtenus :

Problème	Sans mutation		Avec mutation	
	La moyenne	Meilleur	La moyenne	Meilleur
Mk 01	46	45	43	42
Mk 02	35	33	34	33
Mk 03	226	222	215	213
Mk 04	78	73	70	68
Mk 05	187	183	184	181
Mk 06	99	95	93	88
Mk 07	172	165	165	160
Mk 08	530	523	523	523
Mk 09	357	346	346	339
Mk 10	288	278	275	263

Tableau 6 : les résultats de EDA avec /sans mutation

Pour mieux interpréter ces résultats nous présentons une illustration de ce tableau par des histogrammes pour mieux faciliter la tâche. Les figures suivantes représentent des histogrammes des résultats obtenus dans ce tableau :

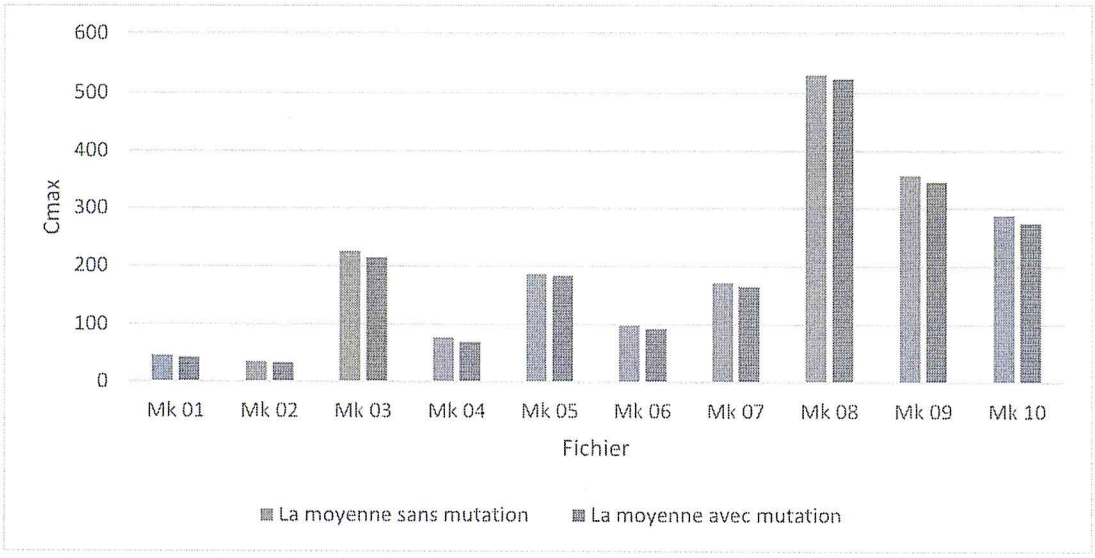

Figure 30 : comparaison EDA avec mutation/sans mutation moyen Cmax.

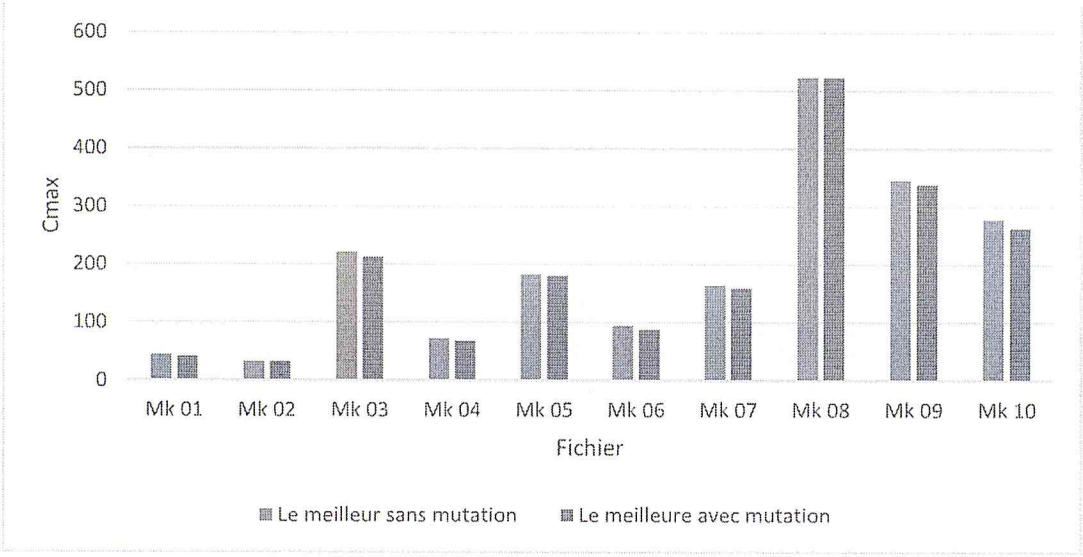

Figure 31 : comparaison EDA avec mutation/sans mutation best Cmax

3.2.1 Interprétation des résultats :

Comme on le voit, les résultats obtenus avec la mutation sont de loin meilleurs que ceux obtenus sans celle-ci. On peut conclure alors que notre EDA doit se dérouler avec la mutation.

Voilà maintenant la mutation confirmée il nous faut maintenant trouver le meilleur paramétrage possible pour l'algorithme, et pour y parvenir nous avons fait une série de tests sur les 10 Benchmarks avec comme paramètres I pour itérations, S pour sélection et NSG pour nombre de population. Le tableau suivant montre les résultats obtenus :

Nombre itération	paramètres	Mkn01		Mk 02		Mk 03		Mk 04		Mk 05	
		moy	min	moy	min	moy	min	moy	min	moy	min
50	S=3,nsg=30	45	43	35	34	225	222	78	77	194	188
	S=5,nsg=50	44	42	34	32	223	220	76	73	183	187
100	S=3,nsg=30	46	44	34	31	222	216	77	74	191	87
	S=5,nsg=50	44	43	34	33	219	213	73	75	187	183
300	S=3,nsg=30	43	42	34	34	220	214	74	73	185	182
	S=5,nsg=50	43	43	34	34	215	213	71	69	183	181
500	S=3,nsg=30	43	42	34	33	219	213	75	71	186	184
	S=5,nsg=50	44	43	34	33	213	213	71	69	187	184

Tableau 7 : EDA avec les différents paramètres pour problèmes mk01...mk05

Nombre itération	paramètres	Mkn06		Mk 07		Mk 08		Mk 09		Mk 10	
		moy	min	moy	min	moy	min	moy	min	moy	min
50	S=3,nsg=30	98	95	177	167	530	523	355	352	291	280
	S=5,nsg=50	97	93	175	157	523	523	352	349	282	271
100	S=3,nsg=30	99	95	180	169	528	523	355	343	275	285
	S=5,nsg=50	92	87	173	163	523	523	347	344	277	273
300	S=3,nsg=30	92	90	173	162	525	523	352	345	291	284
	S=5,nsg=50	96	94	171	169	523	523	340	334	277	275
500	S=3,nsg=30	89	88	172	157	523	523	341	334	274	271
	S=5,nsg=50	96	90	164	159	523	523	340	337	279	266

Tableau 8 : EDA avec les différents paramètres pour problèmes mk06...mk10

La figure suivante représente la courbe des résultats avec différents paramètres :

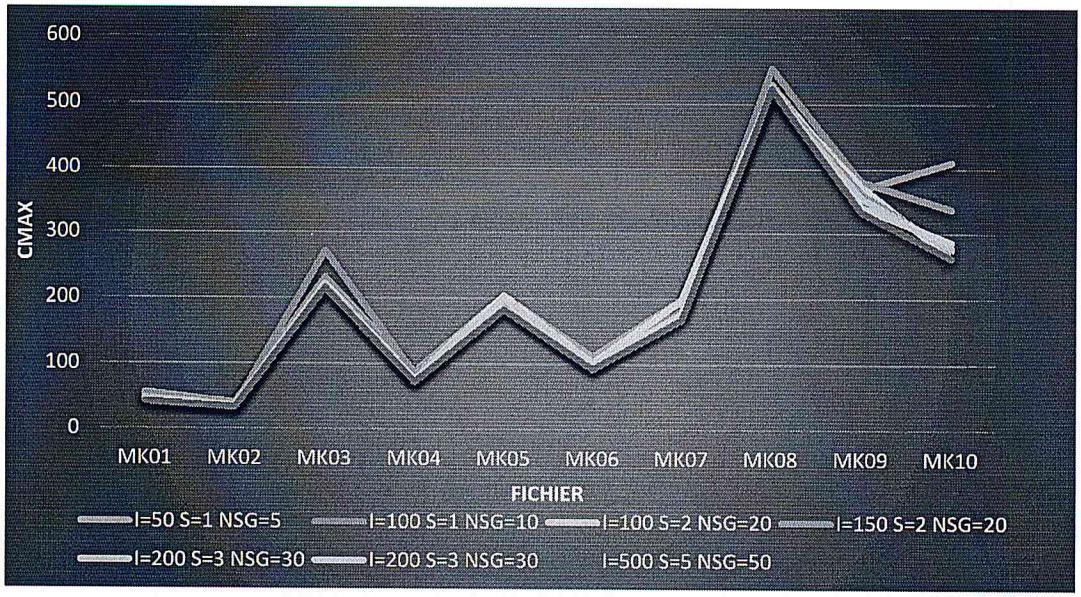

Figure 32 : les résultats des différents Benchmarks

Après interprétation de ces résultats, on remarque que le paramétrage du programme joue un rôle important sur les résultats obtenus à la fin, donc après plusieurs essais on peut dire que le meilleur paramétrage possible pour les 10 Benchmarks est de prendre comme paramètres :

I = 500 S=5 et NSG = 50.

Maintenant qu'on est sûr d'utiliser la mutation et qu'on connait le meilleur paramétrage, on passe à l'étape de validation de l'EDA en comparant les résultats obtenus avec ceux d'autres algorithmes.

Le tableau ci-dessous nous donne la comparaison suivi d'une figure représentant la courbe des résultats obtenus dans le tableau :

Fichier	GENACE	Pezz	EDA	Brandimart
Mk01	41	40	42	42
Mk02	29	26	31	32
Mk03	204	204	213	211
Mk04	67	60	68	81
Mk05	176	173	181	186
Mk06	68	63	87	86
Mk07	148	139	157	157
Mk08	523	523	523	523
Mk09	328	311	339	369
Mk10	231	212	263	296

Tableau 9 : comparaison d'EDA avec les algorithmes de littérature

La figure suivante représente la courbe de ce résultat :

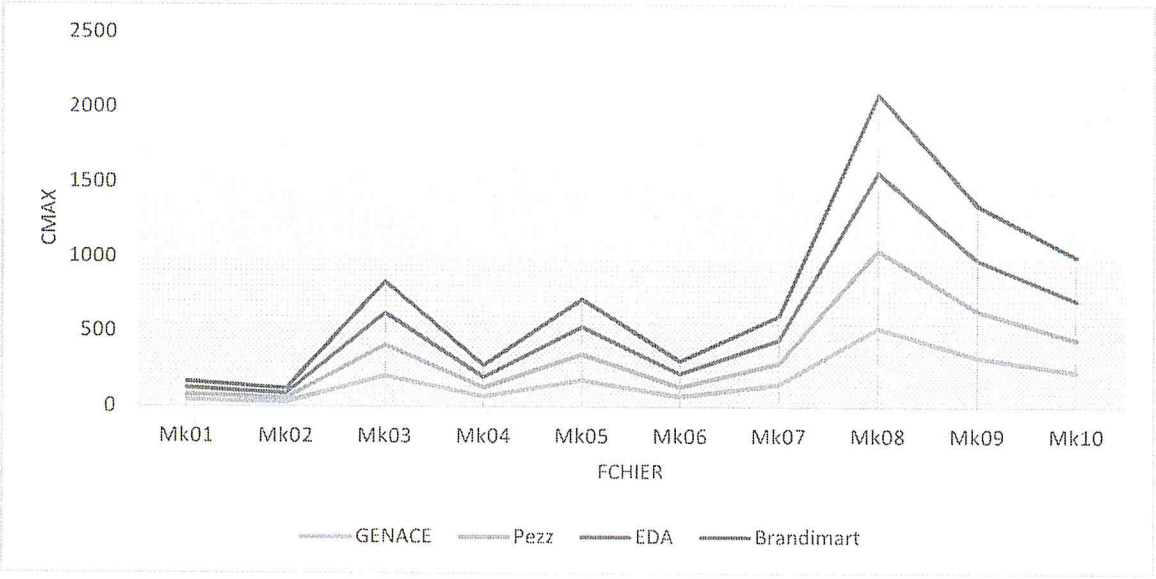

Figure 33 : comparaison EDA avec les algorithmes de littérature

La courbe ci-dessus montre que les résultats obtenus sont compris entre les résultats des autres algorithmes ; ce qui prouve que notre méthode a pu atteindre des résultats satisfaisants avec une approche distribuée qui consomme moins de calcul. Notre EDA se rapproche des méthodes exactes existantes.

3.3 MOEA/D avec l'utilisation de la plateforme JADE :

Afin de valider notre MOEA\D en utilisant le concept de parallélisme, nous avons utilisé la plateforme JADE pour lancer 10 agents sur la plateforme tout en respectant un certain paramétrage qu'on va définir dans ce qui va suivre :

3.3.1 Les pondérations :

Les pondérations utilisées dans chaque agent sont la clé de notre parallélisme car chaque agent va dérouler son MOEA\D avec les pondérations fixées pour lui.

Le tableau suivant représente les 10 pondérations utilisées :

agent	α	β	γ
1	0.33	0.33	0.33
2	0.20	0.40	0.40
3	0.14	0.43	0.43
4	0.40	0.20	0.40
5	0.29	0.29	0.43
6	0.33	0.50	0.17
7	0.43	0.14	0.43
8	0.50	0.33	0.17
9	0.38	0.38	0.25
10	0.80	0.05	0.15

Tableau 10 : les pondérations utilisées pour chaque agent

Comme mentionné auparavant, chaque pondération est fixée pour un agent, l'affectation de ces pondérations se fait de manière manuelle.

Une fois les pondérations fixées, on lance les agents avec le bouton Lancer les agents dans l'interface principale.

Une fois les messages reçus, chaque agent démarre son MOEA\D indépendamment des autres en échangeant des messages avec les agents les plus proches.

3.3.2 L'échange des messages :

Le tableau qui va suivre donne le chemin suivi pour le choix des agents récepteurs et expéditeurs du VP entre les plus proches voisins :

	1	2	3	4	5	6	7	8	9	10
1		0.1634	0.2368	0.1634	0.1148	0.2334	0.2368	0.2334	0.1933	0.5759
2	0.1634		0.0734	0.2828	0.14525	0.2824	0.3484	0.3844	0.2351	0.7388
3	0.2368	0.0734		0.3484	0.2051	0.3295	0.4101	0.4551	0.3041	0.8114
4	0.1634	0.2828	0.3484		0.1452	0.3844	0.0734	0.2824	0.2351	0.4949
5	0.1148	0.14525	0.2051	0.1452		0.3366	0.2051	0.3366	0.2204	0.6293
6	0.2334	0.2824	0.3295	0.3844	0.3366		0.4551	0.0578	0.1526	0.6509
7	0.2368	0.3484	0.4101	0.0734	0.2051	0.4551		0.32954	0.3041	0.4726
8	0.2334	0.3844	0.4551	0.2824	0.3366	0.0578	0.32954		0.1526	0.4108
9	0.1933	0.2351	0.3041	0.2351	0.2204	0.1526	0.3041	0.1526		0.5434
10	0.5759	0.7388	0.8114	0.4949	0.6293	0.5609	0.4726	0.4108	0.5434	

Tableau 11 : la distance euclidienne entre les agents

Les résultats obtenus nous ont permis de suivre le cheminement suivant sur lequel se base l'échange des messages entre les voisins :

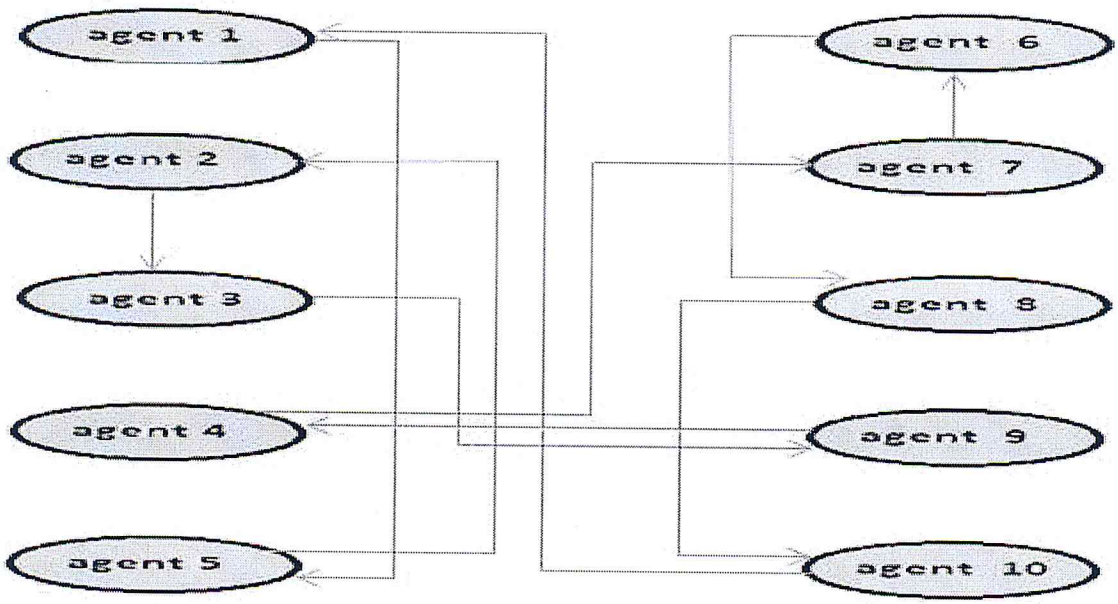

Figure 34 : le cheminement de l'échange des messages entre les voisins

Exemple : la figure ci-dessous représente un échange de message entre les agents.

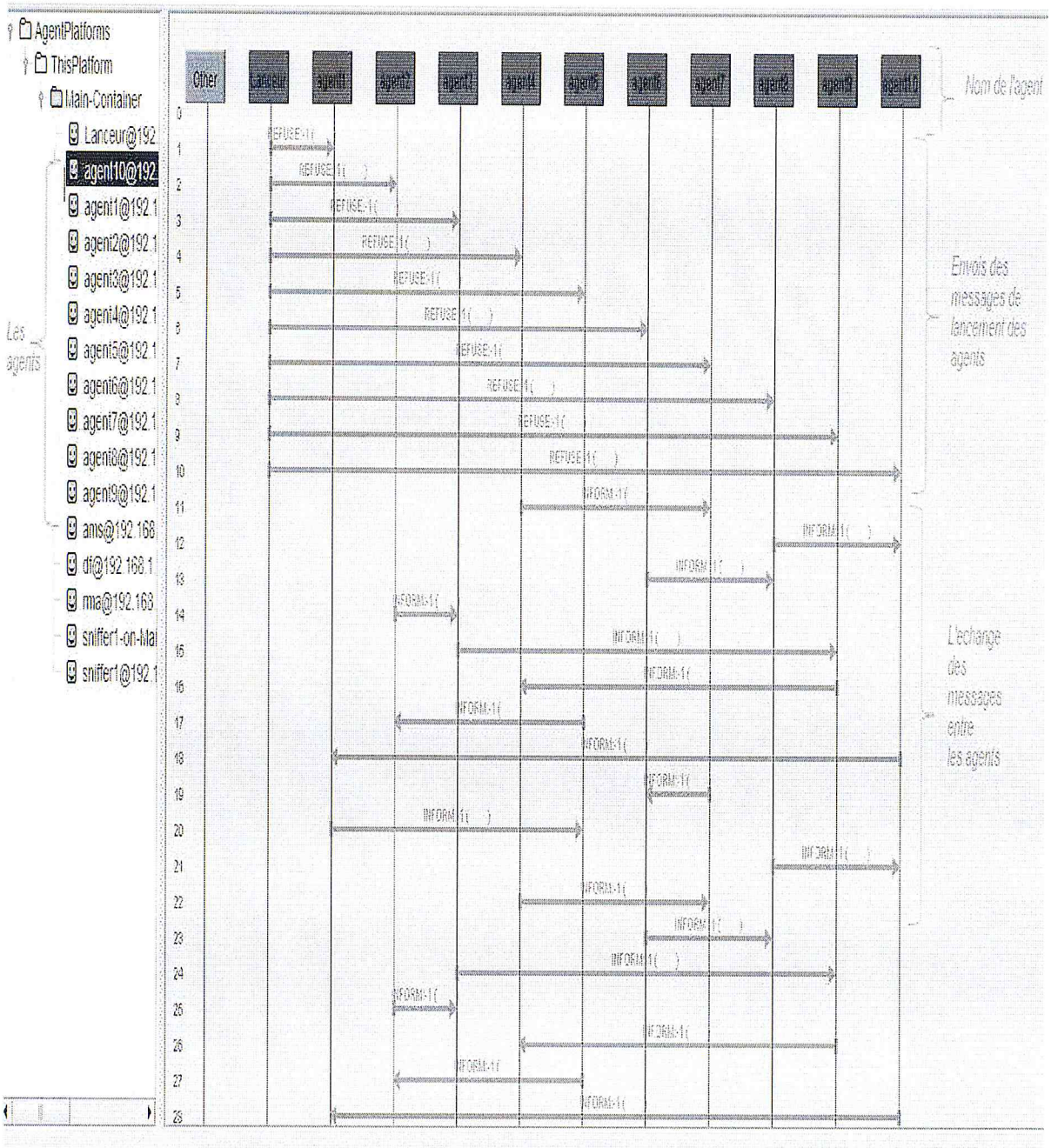

Figure 35 : échange des messages entre les agents

Une fois la plateforme et les paramètres fixés, on passe à l'étape de validation. Des tests sur le Benchmark mk01 nous ont permis d'obtenir des résultats qu'on peut interpréter et comparer avec un algorithme connu pour le multi objectifs NSGII

3.3.3 Aperçu des résultats :

La figure suivante contient une capture d'écran d'un résultat obtenu lors d'une utilisation de la plateforme avec 10 agents :

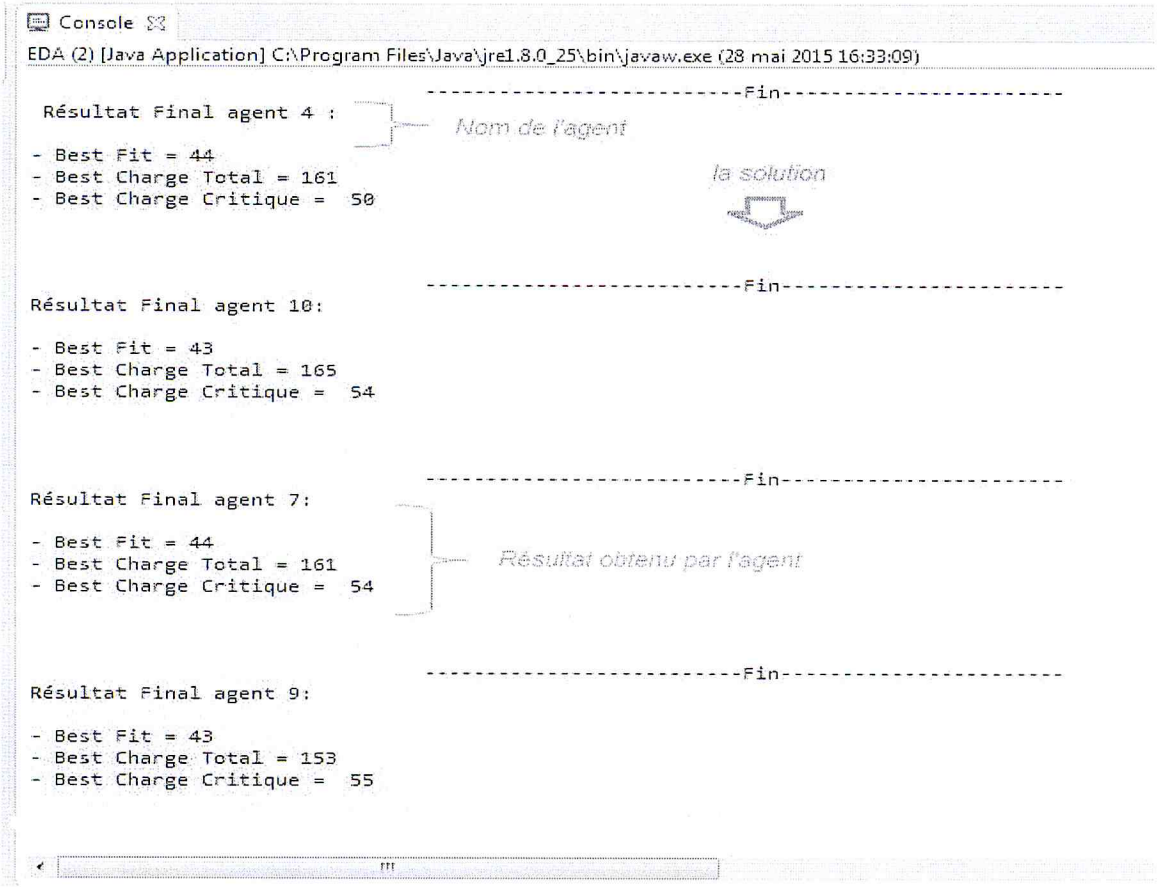

Figure 36 : les résultats des différents agents

3.3.4 Interpréter le front de Pareto :

Pour vérifier la validité de MOEA\D, nous devons confirmer que les résultats obtenus peuvent être dans le front de Pareto. Pour y arriver nous avons testé notre programme sous la plateforme JADE avec les pondérations décrites dans Tableau 10 et le schéma adapté dans la figure 35 ainsi que le paramétrage suivant :

Nombre d'itérations = 500 ; nombre de sélection = 5, taille de population = 50, temps d'envoi et de réception de message = chaque 100 itérations, nombre d'agents = 10.

Les figures suivantes montrent le nuage de points pour les résultats obtenus avec MOEA\D d'une part, et du NSGII d'autre part afin qu'on puisse comparer entre eux par la suite :

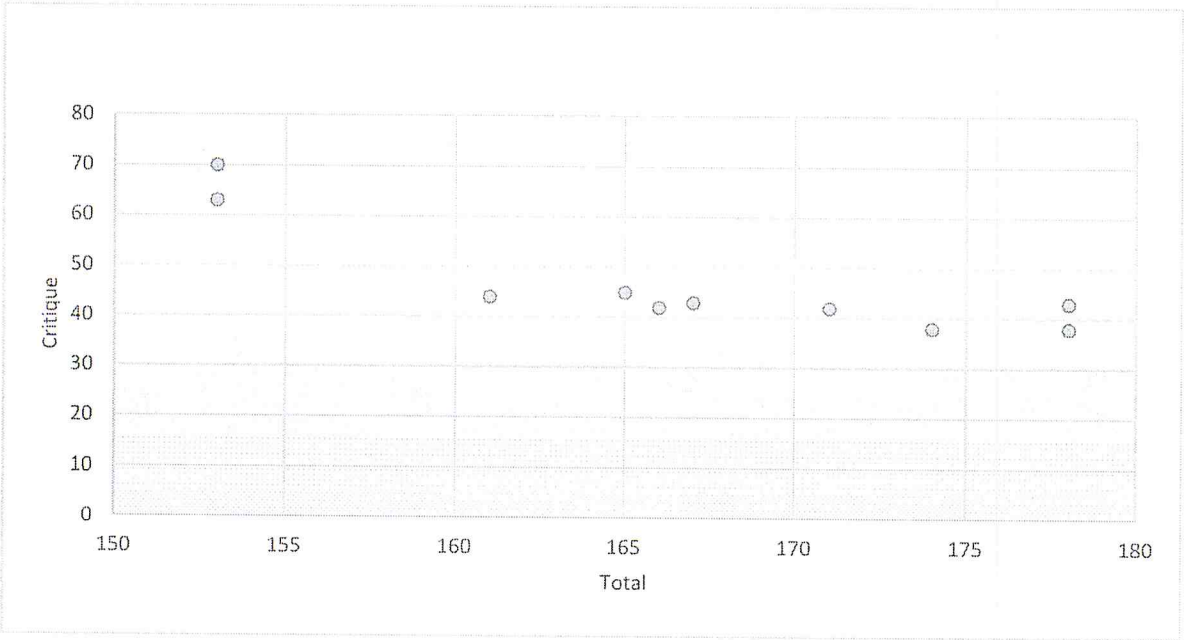

Figure 37 : le résultat de MOEA/D pour la charge critique et la charge totale

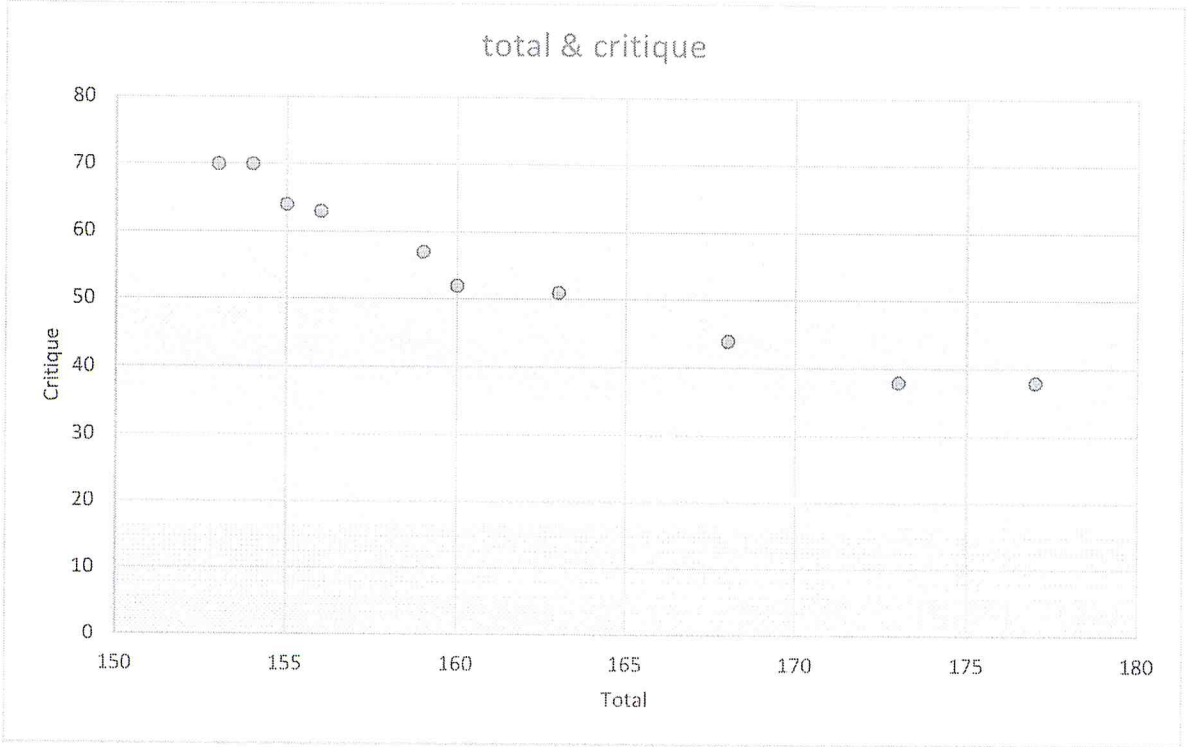

Figure 38 : le résultat de NSGA II pour la charge critique et la charge totale

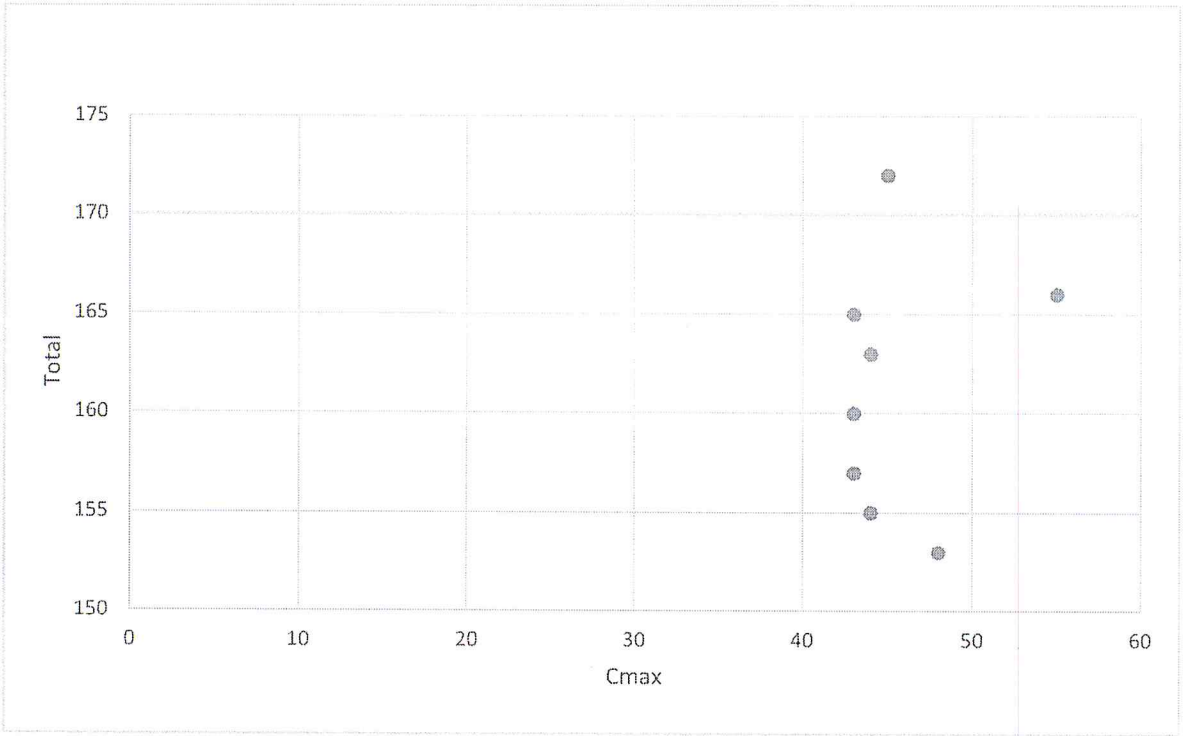

Figure 39 : le résultat de MOEA/D pour la charge totale et Cmax

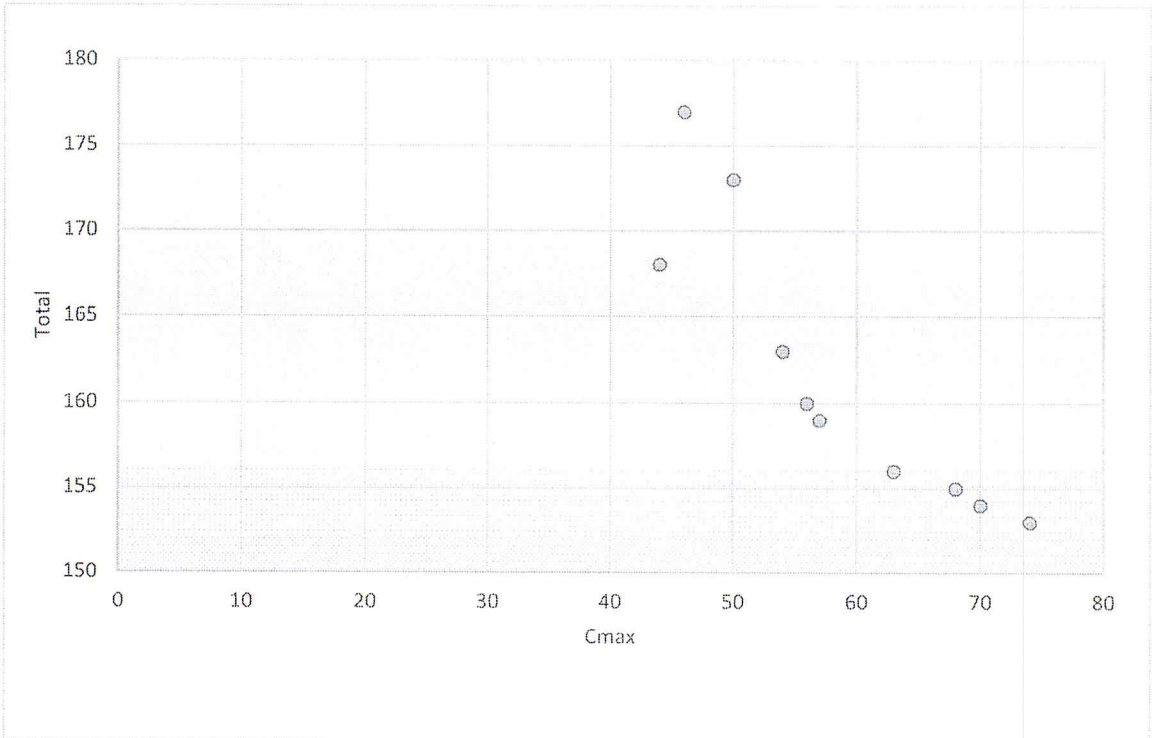

Figure 40 : le résultat de NSGA II pour la charge totale et la Cmax

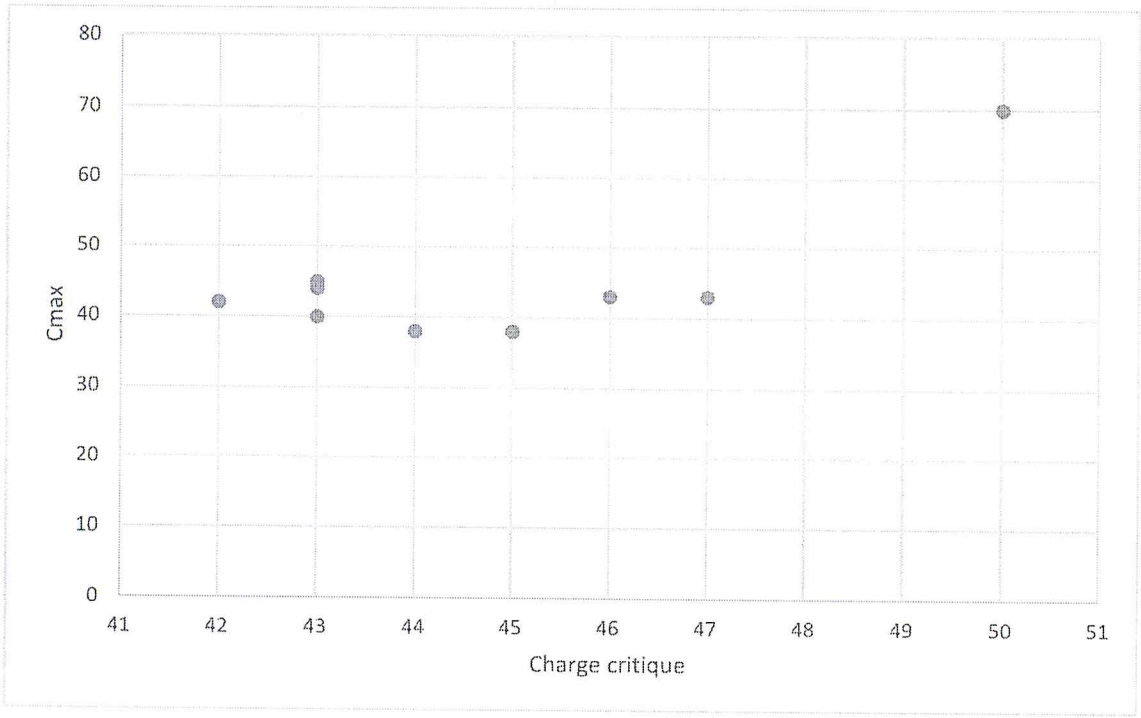

Figure 41 : le résultat de MOEA/D pour Cmax et la charge critique

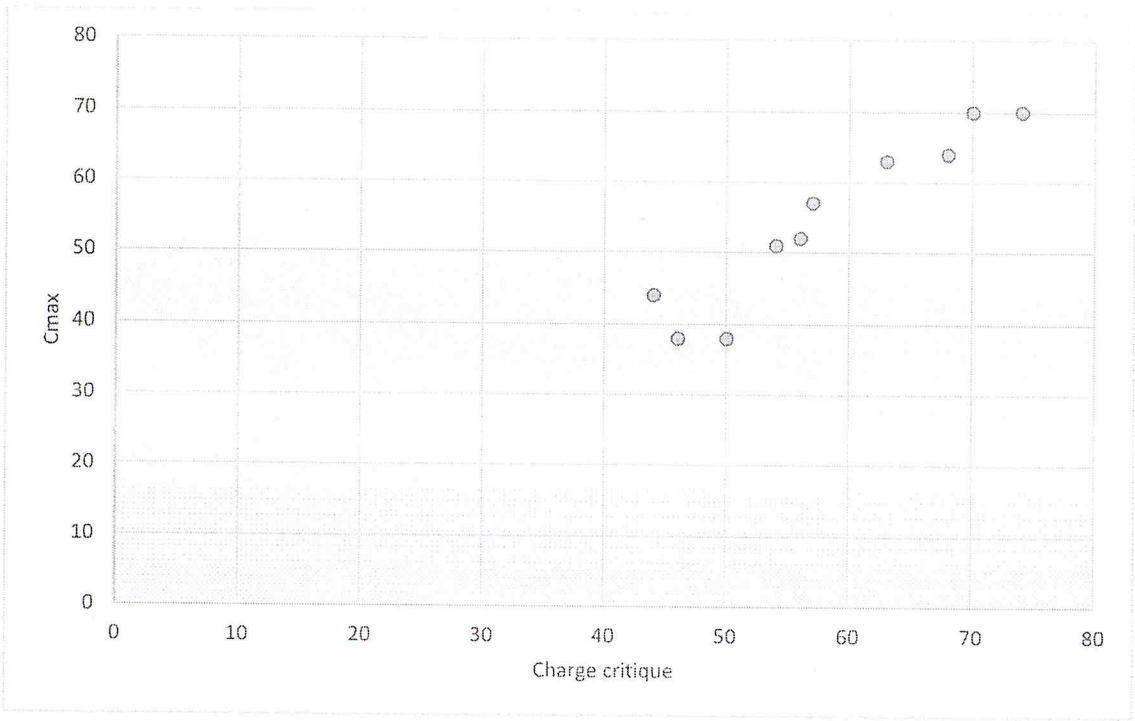

Figure 42 : le résultat de NSGA II pour Cmax et la charge critique

3.4 Interprétation des résultats :

Comme on le remarque dans les différentes figures 37,39 et 41 les résultats obtenus sont assez fiables qui peuvent s'approcher du front de Pareto, et si on compare ces résultats avec ceux de l'algorithme NSGII qui est d'ailleurs très efficace dans le cas multi objectifs, on remarque bien l'apparition du front de Pareto dans les 3 figures, ce qui confirme que les résultats obtenus sont satisfaisants et que l'approche adaptée pour le MOEA\D donne des résultats valides.

Donc on peut également conclure que **notre MOEA\D est une approche intéressante et prometteuse pour le problème de job shop flexible**.

4 Conclusion :

Nous avons présenté dans ce chapitre les résultats des différentes expérimentations réalisées sur un ensemble de problèmes tests avec des Benchmarks. L'objectif a été d'explorer les performances de l'approche qu'on a utilisée, d'un côté, et de valider notre implémentation avec l'EDA pour le mono-objectif et le MOEA\D pour le multi objectifs et pour finir l'hybridation des deux au problème de Job Shop Flexible.

L'exploration des résultats nous a permis d'aboutir aux conclusions suivantes :

- Notre approche donne les mêmes résultats que des approches proposés dans la littérature. Nous constatons d'une part que l'EDA proposé est valable.
- Le MOEA\D proposé est efficace pour plusieurs objectifs, simple à appliquer et ne requiert pas un temps de résolution élevé et donne des résultats qui font partie du front de Pareto.

Et pour finir l'utilisation de la plateforme JADE nous a permis de créer une plateforme avec plusieurs agents qui exécutent en parallèle notre algorithme avec diverses pondérations, ce qui nous a permis de constater la dominance de Pareto dans les résultats obtenus.

CONCLUSION GÉNÉRALE

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

Le problème d'ordonnancement des ateliers de type job shop flexible est un problème d'ordonnancement extrêmement complexe. Il est classé parmi les problèmes combinatoires difficiles au sens fort. Cette complexité est due à l'explosion combinatoire du nombre de solutions qui croît exponentiellement avec la taille du problème. De ce fait, l'utilisation de méthodes exactes en vue de l'obtention de solutions optimales semble non réaliste. Le recours à des méthodes approchées comme les heuristiques est donc devenu incontournable. Parmi ces méthodes, le paradigme des Métaheuristiques s'impose comme une approche très prometteuse. En effet, en plus de leur adaptabilité aux différents problèmes combinatoires, les métaheuristiques 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.

Nous nous sommes intéressés dans ce travail à l'adaptation des métaheuristiques au problème d'ordonnancement de type job shop flexible, avec l'adaptation de l'EDA algorithme à estimation de distribution pour le cas mono objectif, et l'adaptation de MOEA\D pour le cas multi objectifs.

Un cadrage théorique de notre thème consiste à présenter en premier lieu les problèmes d'ordonnancement en général, les problèmes d'ordonnancement des ateliers de job shop flexible. En deuxième lieu, les méthodes de résolution existantes pour le problème de type job shop flexible. Et en troisième lieu les techniques adaptées pour la réalisation des approches en se basant sur des principes proposés par plusieurs chercheurs. Finalement des essais numériques ont été ensuite réalisés afin d'évaluer et de comparer les performances des différentes approches implémentées et de valider notre application.

Ce travail nous a permis de constater que ces méthodes sont intéressantes pour la résolution du problème d'ordonnancement de type job shop flexible.

En ce qui concerne le cas mono objectif, l'application de l'EDA proposé nous a permis de constater les remarques suivantes :

- L'algorithme à estimation de distribution est très facile et simple à mettre en œuvre.
- Son principe de distribution nécessite moins de coût que les algorithmes classiques.
- Sa légèreté en termes de consommation le rend très rapide en termes de calcul.
- Les paramètres de dimensionnement sont un déterminant important de la méthode.
- La détermination empirique de ces paramètres, est une étape très importante avant l'utilisation de ces approches.
- L'intégration de la mutation afin d'introduire l'exploitation à donner de meilleurs résultats.
- Le principe de résolution de cette approche se base sur l'estimation d'un vecteur de probabilité pour la population actuelle, et le distribuer pour la population nouvelle.
- L'EDA développé est très performant pour la résolution des problèmes d'ordonnancement de type job shop flexible. Son utilisation pour la résolution de benchmarks connus a donné de bons résultats comparativement à ceux de la littérature. Son implémentation est de ce fait réussie.

Dans le MOEA\D proposé pour le cas multi objectifs, une hybridation avec l'EDA a été faite. L'approche réalisée nous a permis de constater ce qui suit :

- L'intégration de l'EDA pour le MOEA\D était une bonne idée vu que l'EDA traite le mono objectif, ainsi que le MOEA\D qui traite le cas multi objectifs à base de décomposition.
- Le parallélisme du MOEA\D se base sur les pondérations fixées au début ; diversifier ces pondérations revient à diversifier l'espace de recherche.
- Afin de réaliser cette parallélisation, il nous a fallu utiliser la plateforme multi agents JADE.
- Dans la plateforme, chaque agent a ces propres pondérations et donc son propre espace de recherche. Les résultats obtenus de chaque agent forment le résultat de tout l'espace de recherche.

- L'interprétation du résultat se fait par la dominance de Pareto en vérifiant si le front est réalisé ou non.
- Les résultats étudiés sur des Benchmarks montrent que le front de Pareto a été approché avec les pondérations qu'on a utilisées.
- Un front de Pareto représente les résultats qui dominent les autres.
- Le choix de l'utilisation de la plateforme JADE est dû au fait que celle-ci est simple et rapide à mettre en œuvre avec une possibilité de visualiser et communiquer entre les différents agents.
- Le MOEA\D développé est acceptable pour la résolution des problèmes multi objectifs, pour le problème d'ordonnancement de type job shop flexible. Son implémentation est de ce fait réussie.

On peut conclure que les approches proposées sont encore très récentes, et que l'adaptation de ces approches est nouvelle dans le domaine, ce qui nous pousse à dire que même si les résultats obtenus sont pour le moins satisfaisants pour l'instant, il y aura toujours des améliorations à faire afin de raffiner de plus en plus l'approche adapté que ce soit pour l'EDA ou bien pour le multi objectifs.

Bibliographie

ABDELLAH, B. (2005). *optimisation multi objectif évolutionnaire.* Tunisie: Ecole Polytechnique.

BAPTISTE, P. (1998). *une étude théorique et expérimentale de la propagation des contraintes de ressources.* université de technologie de compiègne.

BENLAHRACHE, N. (2007). *optimisation multi objectif pour l'alignement multiple de séquence.* Université Mentouri de Constantine.

BERACHI, M. (2010). *Algorithme Evolutionnaire a Etats pour l'Optimisation Diffcile. Data Structures and Algorithms.* france: Université Nice Sophia Antipolis.

BERRO, A. (2008). Algorithme évolutionnaire pour l'optimisation multiobjectif. *Séminaire du 4 novembre 2008 - LAAS.* toulouse: Laboratoire d'analyse et d'architucture des systèmes.

COLLETTE, Y., & SIARRY, P. (2002). *optimisation multiobjectif.* paris: Eyrolles.

DRéO, J. (2004). *Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical.* université Paris12.

DREO, J., & SIARRY, P. (s.d.). *Métaheuristiques d'optimisation vues sous l'angle de l'échantillonnage de distribution.*

DUPAS, R. (2004). *Amélioration de performance des systèmes de production:apport des algorithmes évolutionnistes aux problèmes d'ordonnancement cycliques et flexibles.* Université d'Artois.

ESQUIROL, &., & LOPEZ. (1999). *l'ordonnancement.* paris: Economica.

GIARD, V. (1988). *gestion de la production.* paris: economica.

GOLDBERG, & DAVID, E. (1989). *genetic algorithms in search optimization and machine learning.*Addison wisley.

HABIBA, H. (2012). *Planification et Ordonnancement en temps réel d'un Job shop en utilisant l'Intelligence Artificielle.* Université de Tlemcen.

HENTOUS, H. (1999). *contribution au pilotage des sysèmes du production du type job shop.* lyon.

HENTOUS, H., & GUINET. (1997). *A new heuristic to minimize completion.* lyon: International conference on industrial engineering and production.

KACEM, I. (2003). *ordonnancement multi critère des job shop flexible :formulation bornes inferieurs et approche evolutionniste coopérative.* université lille.

KONE, O. (2009). *Nouvelles approches pour la résolution du problème.* Université Toulouse.

LETOUZEY, A. (2001). *Ordonancement interactif basé sur des indicateurs:Applications a la gestion de commandes incertaines et a l'affectation ds opérateurs.* institut nationale polytechnique de toulouse.

MARQUET, G. (2014). *Stratégies scalaires et parallèles en optimisation multi-objectifs.* Université Lille 1.

MEZIANE, A. (2011). *optimisation par phases pour le problème d'ordonnacement des ateliers de type job shop totalement flexibles.* universite d'oran.

OURARI, S. (2011). *De l'ordonnancement déterministe à l'ordonnancement.* UNIVERSITÉ DE TOULOUSE.

ROUDENKO, O. (2004). *Application des algorithmes 'evolutionnaires aux problemes d'optimisation multi-objectif avec contraintes.* paris: Ecole Polytechnique.

SCHOENAUER, M. (2011). *projet fractale.* france.

SENECHAL, O. (2004). *Pilotage des systèmes de production vers la performance globale.*UNIVERSITE DE VALENCIENNES ET DU HAINAUT CAMBRESIS.

SIMON. (1982). *sequencing and scheduling an introduction to the mathematics of the job-shop.* new york: wiley.

TAGHEZOUT, N. (2011). *Conception et Développement d'un système multi-agent d'Aide à la Décision pour la gestion de production dynamique.* Université Toulouse.

TRENTESAUX. (2002). *pilotage hétérarchique des système de production.* université de valencienne.

VACHAR, J. (2000). *un système adaptatif par agent avec utilisation des algorithmes génétiques muliti objectifs :application a l'ordonnacement d'atelier de type job shop.* université de Havre.