Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques : optimisation par algorithmes génétiques et essaims particulaires

Hela Boukef Ben Othman Boukef

N° d'ordre : 98 **ÉCOLE CENTRALE DE LILLE**

UNIVERSITÉ DE TUNIS EL MANAR

ÉCOLE NATIONALE D'INGÉNIEURS DE TUNIS

THÈSE

présentée en vue
d'obtenir le grade de

DOCTEUR

en Automatique et Informatique Industrielle

par

Hela BOUKEF BEN OTHMAN

Doctorat délivré conjointement par l'École Centrale de Lille et l'École Nationale d'Ingénieurs de Tunis

Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques Optimisation par algorithmes génétiques et essaims particulaires

soutenue le 3 Juillet 2009, devant le jury d'examen composé de :

MM.	**Noureddine ELLOUZE**	**Président**
	Abdellah EL MOUDNI	**Rapporteur**
	Noureddine LIOUANE	**Rapporteur**
	Imed KACEM	**Examinateur**
	Mohamed BENREJEB	**Co-Directeur de Thèse**
	Pierre BORNE	**Directeur de Thèse**

Thèse en cotutelle préparée au Laboratoire d'Automatique, Génie Informatique et Signal de l'Ecole Centrale de Lille et à l'Unité de Recherche LARA Automatique de l'Ecole Nationale d'Ingénieurs de Tunis

Avant Propos

Ce présent travail a été effectué au sein de l'Unité de Recherche LARA Automatique de l'Ecole Nationale d'Ingénieurs de Tunis (ENIT) et du Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS) de l'Ecole Centrale de Lille (EC-Lille).

Nous sommes particulièrement sensibles au grand honneur que Monsieur le Professeur Noureddine ELLOUZE, Directeur de l'Unité de Recherche LSTS de l'Ecole Nationale d'Ingénieurs de Tunis, nous fait en acceptant de présider notre Jury d'Examen. Qu'il trouve ici l'expression de notre profonde reconnaissance.

C'est un agréable devoir pour nous d'exprimer notre très vive reconnaissance à Monsieur le Professeur Mohamed BENREJEB, Directeur de l'Unité de Recherche LA.R.A. Automatique, et à Monsieur Pierre BORNE, Professeur à l'Ecole Centrale de Lille pour nous avoir guidé durant toute l'élaboration de ce mémoire avec le sérieux et la compétence qui les caractérisent. Qu'ils trouvent ici le témoignage de notre très profonde gratitude.

Nous tenons à remercier vivement Monsieur Abdellah EL MOUDNI, Professeur à l'Université de Technologie Belfort-Monbéliard et Monsieur Noureddine LIOUANE, Maître de Conférences à l'Institut Supérieur des Sciences Appliquées et Technologies de Gafsa et Directeur de l'Institut Supérieur des Sciences Appliquées et Technologies de Kairouan, d'avoir bien voulus accepté de rapporter sur notre travail. Qu'ils trouvent ici, le témoignage de notre profonde reconnaissance.

Nos remerciements s'adressent également à Monsieur Imed KACEM, Professeur à l'Université Paul Verlaine-Metz; pour l'intérêt qu'il a bien voulu porter à nos travaux en acceptant de participer à notre Jury d'Examen.

Nous tenons, enfin à remercier tous les chercheurs de l'Unité de Recherche LARA Automatique de l'ENIT et du Laboratoire d'Automatique, Génie Informatique et Signal de l'EC-Lille pour leur amicale présence et la sympathie qu'ils nous ont constamment témoignées. Nous leur exprimons, ici, toute notre gratitude.

Table des Matières

Table des Figures

Liste des Tableaux

Introduction générale

Parmi les problèmes rencontrés par le chercheur et l'ingénieur, les problèmes d'optimisation occupent à notre époque une place de choix. Formuler les problèmes d'optimisation et tenter de les résoudre représentent l'objectif principal de nombreux chercheurs.

Comprendre, analyser et formuler un problème d'optimisation nécessitent d'abord une définition des paramètres, des variables, de l'espace de recherche ainsi que des fonctions à optimiser.

Une fois la (ou les) fonction(s) à optimiser définie(s), une méthode adaptée pour la résolution du problème posé est choisie.

A ce niveau, la taille et la complexité du problème entrent en compte pour le choix de la méthode d'optimisation. Si le problème est de petite taille et de complexité réduite, la mise en œuvre d'une méthode exacte peut suffire et aboutir à une solution optimale.

Dans le cas de problèmes de tailles importantes, les méthodes approchées constituent le moyen le plus efficace de se rapprocher le plus possible de la solution optimale.

Qu'il s'agisse d'une optimisation mono ou multi-objectifs entre également en ligne de compte. Dans le cas d'un objectif unique, la définition de la fonction fitness, f, ne pose généralement pas de problème. Par exemple, si l'on se fixe l'objectif de minimiser un coût C, la fonction fitness sera égale à C.

Certains problèmes d'optimisation doivent satisfaire des objectifs multiples, souvent concurrents, ce qui nécessite parfois la recherche d'un compromis. Une méthode classique, en présence de fonctions objectifs f_i, consiste à les combiner en effectuant, par exemple, une somme pondérée des fonctions objectifs, $f = \sum_i \alpha_i f_i$, ramenant ainsi un problème multi-objectifs à un problème mono-objectif.

C'est à l'utilisateur de fixer convenablement les poids des objectifs, tenant compte de leur importance ou de les adapter, parfois, par tâtonnement.

Les problèmes d'ordonnancement dans le secteur industriel, sont parmi les problèmes d'optimisation les plus étudiés. Améliorer le rendement des ressources et minimiser les coûts

de production sont devenus les leitmotivs des industriels. Chercher le meilleur moyen de maximiser son profit est aujourd'hui l'un des objectifs principaux de toute entreprise.

C'est dans ce contexte qu'entre nos travaux de recherche. Ils concernent la résolution de problèmes multi-objectifs d'ordonnancement en industries pharmaceutiques. Au niveau de ce type d'industries, assurer la production en quantité et surtout de qualité irréprochable, dans les délais impartis et tenant compte des différentes saisons tout en minimisant les coûts, représente un challenge de tous les jours.

Sachant que les problèmes de production dans les industries pharmaceutiques sont complexes et nécessitent la prise en compte de plusieurs facteurs essentiellement liés au respect de l'hygiène ainsi que de l'assurance et du contrôle de la qualité, nous nous orientons pour leur résolution vers le choix des méthodes approchées.

Deux méthodes sont donc utilisées tout au long de ce rapport. La méthode des algorithmes génétiques et la méthode d'optimisation par essaim particulaire qui sont des méthodes évolutionnistes.

Les algorithmes évolutionnistes doivent leur nom à l'analogie avec les mécanismes d'évolution des espèces vivantes. Un algorithme évolutionniste est composé de trois éléments essentiels : une *population* constituée de plusieurs individus représentant des solutions potentielles pour problème posé, un *mécanisme d'évaluation* de l'adaptation de chaque individu de la population à l'égard de son environnement et un *mécanisme d'évolution* composé d'opérateurs permettant d'éliminer certains individus et de produire de nouveaux individus à partir des individus sélectionnés.

Un algorithme évolutionniste débute, donc par la création d'une population initiale souvent générée aléatoirement et répète ensuite un cycle d'évolution composé de trois étapes essentielles qui sont la *mesure de la qualité* de chaque individu de la population par le mécanisme d'évaluation, la *sélection* des individus pour une éventuelle évolution et la *génération* de nouveaux individus par recombinaisons d'individus sélectionnés. Une condition d'arrêt indique la fin de ce processus.

Le premier chapitre de ce mémoire propose, dans un premier temps, une vue d'ensemble sur les problèmes d'ordonnancement des systèmes de production et sur leur complexité. Ainsi les différentes composantes de l'ordonnancement sont présentées et les types d'ateliers pouvant les caractériser introduits. Différentes représentations possibles des problèmes d'ordonnancement sont par la suite proposées. Dans un deuxième temps, nous focalisons notre attention sur la présentation des différentes méthodes d'optimisation allant des méthodes

exactes aux méthodes approchées indiquant celles qui sont les plus utilisées dans la littérature. En conclusion à ce chapitre, la problématique relative à l'ordonnancement en industries pharmaceutiques et des critères à minimiser est présentée.

Dans la première partie du deuxième chapitre, les spécificités et les différents problèmes rencontrés dans un atelier de conditionnement en industries pharmaceutiques sont introduits. Les lignes de conditionnement composant le poste en question sont détaillées nous amenant ainsi à nous intéresser aux ateliers de type flow-shop dont elles font partie.

La deuxième partie, quant à elle traite de la résolution du problème multi-objectifs posé en utilisant la méthode des algorithmes génétiques.

Un Codage spécifique est recherché pour permettre la meilleure représentation possible du problème traité.

Deux exemples sont par la suite traités, et les résultats relatifs consignés pour leur comparaison ultérieure avec la méthode d'optimisation par essaim particulaire au niveau du chapitre suivant.

Dans le troisième chapitre, la méthode d'optimisation par essaim particulaire est introduite et son utilisation dans le cas continu présentée. La formulation de cette méthode est, par la suite, modifiée pour permettre son adaptation au cas discret.

Dans une première partie, trois exemples traitant de l'ordonnancement job-shop flexible ont été traités et comparés avec des résultats obtenus par utilisation des algorithmes génétiques. Dans une deuxième partie, nous revenons au problème d'ordonnancement en industries pharmaceutiques traité au deuxième chapitre pour effectuer une comparaison des résultats obtenus.

Chapitre I

Ordonnancement : spécificités, ateliers, méthodes et complexité

I.1 - Introduction

La réalisation d'un projet nécessite souvent une succession de tâches auxquelles s'attachent certaines contraintes :

- de temps, relatives aux délais à respecter pour l'exécution des tâches,
- d'antériorité**,** où certaines tâches doivent s'exécuter avant d'autres,
- de production, concernant le temps d'occupation du matériel ou des hommes qui l'utilisent, ...

Les techniques d'ordonnancement dans le cadre de la gestion d'un projet ont pour objectif de répondre au mieux aux besoins exprimés par un client, au meilleur coût et dans les meilleurs délais, en tenant compte des différentes contraintes.

L'ordonnancement se déroule en trois étapes qui sont:

- la planification, qui vise à déterminer les différentes opérations à réaliser, les dates correspondantes, et les moyens matériels et humains à y affecter.
- l'exécution, qui consiste à mettre en œuvre les différentes opérations définies dans la phase de planification.
- le contrôle, qui consiste à effectuer une comparaison entre planification et exécution, soit au niveau des coûts, soit au niveau des dates de réalisation.

Ainsi, le résultat d'un ordonnancement est un calendrier précis de tâches à réaliser qui se décompose en trois importantes caractéristiques :

- l'affectation, qui attribue les ressources nécessaires aux tâches,
- le séquencement, qui indique l'ordre de passage des tâches sur les ressources,
- le datage, qui indique les temps de début et de fin d'exécution des tâches sur les ressources.

Dans ce chapitre, quelques généralités sur les problèmes d'ordonnancement dans les ateliers de production dont les spécificités : les types d'ateliers, les critères d'optimisation et la complexité sont introduites. Dans un deuxième temps, une description des principales méthodes d'optimisation utilisées dans la littérature est réalisée nous permettant ainsi de présenter celles que nous utiliserons dans la suite de ce rapport.

I.2 - Généralités sur l'ordonnancement

L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activités de l'économie, depuis l'industrie manufacturière [Pinedo, 55] jusqu'à l'informatique [Blazewicz et al, 96].

Ordonnancer le fonctionnement d'un système industriel de production consiste à gérer l'allocation des ressources au cours du temps, tout en optimisant au mieux un ensemble de critères [Rodammer et al, 88]. C'est aussi programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution [Carlier et al, 88].

Ordonnancer peut également consister à programmer l'exécution des opérations en leur allouant les ressources requises et en fixant leurs dates de début de fabrication.

D'une manière plus simple, un problème d'ordonnancement consiste à affecter des tâches à des ressources à des instants donnés pour répondre au mieux aux besoins exprimés par un client, au meilleur coût et dans les meilleurs délais, tout en tenant compte des contraintes.

Les problèmes d'allocation des ressources, d'organisation des tâches, de respect des délais et de prise de décision en temps requis constituent autant de difficultés qu'il est nécessaire de surmonter dans la gestion des systèmes de production en milieu industriel.

Au niveau de l'entreprise, l'ordonnancement concerne plusieurs postes : les ventes, la production, la maintenance, etc. Son rôle est de plus en plus important, car il permet une gestion de ces différents postes qui peut être optimale.
Pour la bonne gestion de ces postes ainsi que des contraintes pouvant y être reliées, il est nécessaire :

- de déterminer les différentes opérations à réaliser, les dates correspondantes, les moyens matériels et humains à y affecter,
- d'exécuter ces opérations et de contrôler les coûts qui en découlent.

C'est ainsi que l'ordonnancement intervient pour permettre la meilleure gestion possible du système de production.

I.3 - Formulation d'un problème d'ordonnancement

Les problèmes d'ordonnancement apparaissent dans tous les domaines : informatique, industrie, construction, administration, etc [Carlier, 88].
Les différentes données d'un problème d'ordonnancement sont les tâches, les ressources, les contraintes et les critères.
Ainsi, étant donnés un ensemble de tâches et un ensemble de ressources, il s'agit de programmer les tâches et affecter les ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), en respectant un ensemble de contraintes.

I.3.1 - Les tâches

Une tâche est une entité élémentaire localisée dans le temps, par une date de début et/ou de fin, et dont la réalisation nécessite une durée préalablement définie.
Elle est constituée d'un ensemble d'opérations qui requiert, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif.
On distingue deux types de tâches :

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

I.3.2 - Les ressources

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

- *les ressources renouvelables,* qui, après avoir été allouées à une tâche, redeviennent disponibles (machines, personnel, etc),
- *les ressources consommables,* qui, après avoir été allouées à une tâche, ne sont plus disponibles (argent, matières premières, etc).

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Par ailleurs, dans le cas des ressources renouvelables, on distingue principalement, les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité .

I.3.3 - Les contraintes

Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distinguées [Carlier et al, 88] : contraintes de ressources et contraintes temporelles.

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

  Les contraintes relatives aux ressources peuvent être disjonctives, induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource, ou cumulatives impliquant la limitation du nombre de tâches à réaliser en parallèle.
- *les contraintes temporelles* : elles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. Ces contraintes peuvent être :
  - des contraintes de dates butoirs, certaines tâches doivent être achevées avant une date préalablement fixée,
  - des contraintes de précédence, une tâche *i* doit précéder la tâche *j*,

- des contraintes de dates au plus tôt, liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

I.3.4 - Les critères

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi.
Les critères dépendant d'une application donnée sont très nombreux; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la plus satisfaisante dépend du ou des critères préalablement définis, pouvant être classés suivant deux types, réguliers et irréguliers.
Les différents critères ne sont pas indépendants; certains même sont équivalents. Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement [Carlier et al, 88]

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

La satisfaction de tous les critères à la fois est souvent délicate, car elle conduit souvent à des situations contradictoires [Roy et al, 93] et à la recherche de solutions à des problèmes complexes d'optimisation.

I.4 - Les ateliers

Une classification des problèmes d'ordonnancement dans un atelier peut s'opérer selon le nombre de machines et leur ordre d'utilisation pour fabriquer un produit, qui dépend de la nature de l'atelier considéré. Un atelier est caractérisé par le nombre de machines qu'il contient et par son type.
Comme le montre la figure1.1, On distingue les trois types d'ateliers suivants : flow-shop, job-shop et open-shop, avec des extensions possibles pour chacun d'eux.

I.4.1 - Les ateliers de type flow-shop

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

I.4.2 - Les ateliers de type job-shop

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

I.4.3 - Les ateliers de type open-shop

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

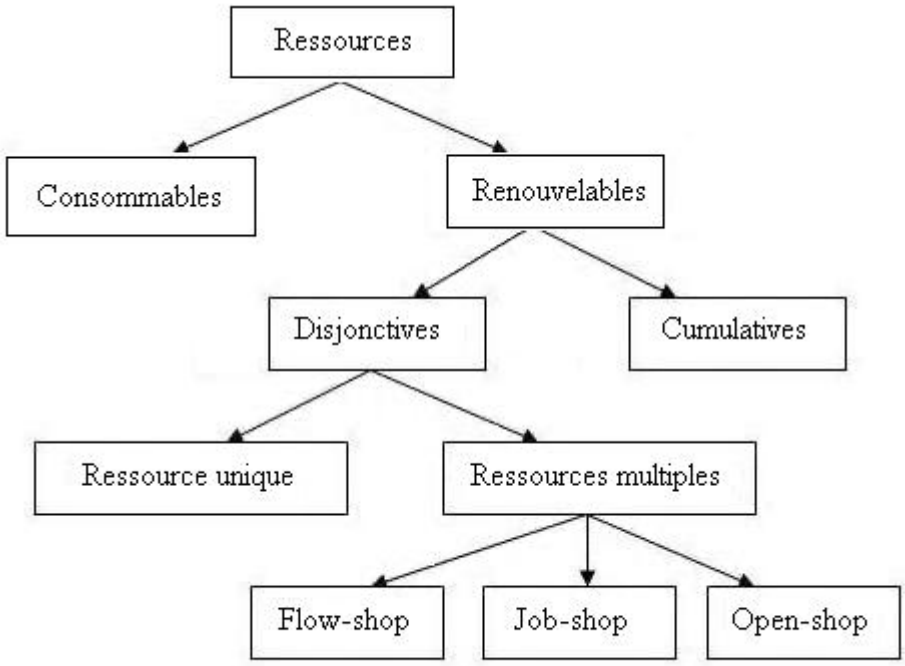

Figure 1. 1 - Classification des types d'ateliers

I.5 - Représentation des problèmes d'ordonnancement

Il existe trois sortes de représentations possibles d'un problème d'ordonnancement: le diagramme de Gantt, le graphe Potentiel-Tâches et la méthode PERT.

I.5.1 - Le diagramme de Gantt

Le diagramme de Gantt est un outil permettant de modéliser la planification des tâches nécessaires à la réalisation d'un projet. Il s'agit d'un outil élaboré en 1917 par Henry L. Gantt. Etant donné la facilité relative de lecture des diagrammes de Gantt, cet outil est utilisé par la quasi-totalité des chefs de projet dans tous les secteurs. Il permet de représenter graphiquement l'avancement du projet et constitue également un bon moyen de communication entre les différents acteurs d'un projet. Le diagramme de Gantt présente en ordonnée la liste des tâches, notées T_i à exécuter par les machines notées M_j et en abscisse l'échelle du temps, comme le montre la figure 1.2 dans le cas où i = 1,2,…,5 et j = 1,2.

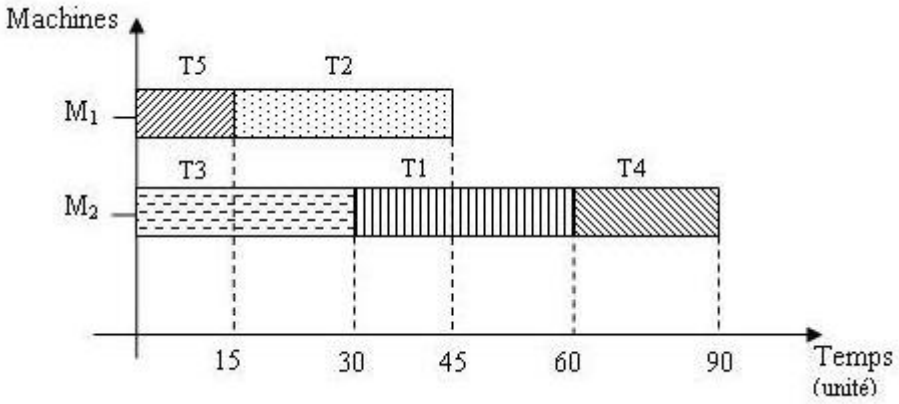

Figure 1. 2 - Diagramme de Gantt d'un ordonnancement

I.5.2 - Graphe Potentiel-Tâches

Cette outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à évènements discrets [Carlier et al, 84].

Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les contraintes par des arcs [Roy 70], comme le montre la figure 1.3.

Ainsi, les arcs peuvent être de deux types :

- les arcs conjonctifs illustrant les contraintes de précédence et indiquant les durées des tâches,
- les arcs disjonctifs indiquant les contraintes de ressources [Gotha, 93], [Jain et al, 99].

Exemple de graphe potentiel-tâches

Tableau 1. 1 - Données utilisées pour la réalisation d'un graphe potentiel-tâches

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

Pour qu'une tâche puisse commencer, il est nécessaire que toutes les tâches qui la relient à la tâche du début **S** du projet, soient réalisées. On définit donc :

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

  Exemple : la tâche f ne peut s'exécuter que si a et d ont été réalisées. Donc, pour exécuter a il faut 6 mois et pour exécuter d il faut 2+3 mois. La tâche f ne pourra commencer au plus tôt que 6 mois après le début du projet: c'est donc le plus long chemin entre a et f.
- **la durée du projet**, qui correspond au plus long chemin entre S (tâche de début du projet) et S' (tâche de fin du projet).

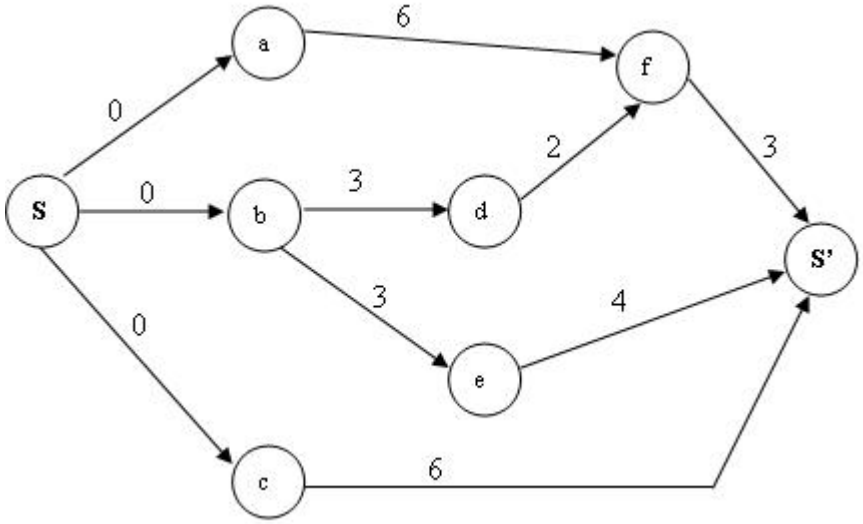

Figure 1. 3 - Graphe Potentiel-Tâches d'un ordonnancement

I.5.3 - Méthode PERT (Program Evaluation and Research Task)

Cette représentation, semblable à la précédente, permet de représenter une tâche par un arc, auquel est associé un chiffre qui représente la durée de la tâche. Entre les arcs, figurent des cercles, appelés sommets ou événements, qui marquent l'aboutissement d'une ou de plusieurs tâches. Ces cercles sont numérotés afin de suivre l'ordre de succession des divers évènements.

Les méthodes graphiques ont connu une très importante évolution surtout avec l'apparition des Réseaux de Pétri (RdP) [Chrétienne, 83], [Carlier et al, 84], qui permettent de traduire plusieurs notions fondamentales ayant un lien avec les problèmes d'ordonnancement, telles que :

- les conflits sur les ressources,
- les durées opératoires certaines,
- les durées opératoires aléatoires,
- les gammes,
- les disponibilités, les multiplicités et les capacités de ressources
- la répétitivité, etc …

I.6 - Complexité des problèmes d'ordonnancement

D'une manière générale, les problèmes d'ordonnancement d'ateliers étant des problèmes combinatoires difficiles, il n'existe pas de méthodes universelles permettant de résoudre tous les cas.

Plusieurs algorithmes peuvent être utilisés pour résoudre un problème d'ordonnancement mais tous ne sont pas équivalents.

On peut différencier les divers algorithmes de résolution par le moyen des critères suivants :

- l'efficacité de l'algorithme en terme de durée d'exécution; un algorithme est dit plus efficace qu'un autre si pour les mêmes données, il s'exécute en un laps de temps plus court;
- l'efficacité de l'algorithme en espace mémoire de stockage; un algorithme est dit plus efficace qu'un autre si pour résoudre le même problème, il utilise moins d'espace mémoire;
- la fiabilité de l'algorithme; plus un programme est complexe, plus il y a des risques d'existence de bugs, les bugs étant des erreurs plus ou moins évidentes qui se manifestent lors de la mise en exploitation d'un programme. Un programme est jugé plus fiable ou plus stable qu'un autre s'il présente moins de bugs;
- la robustesse de l'algorithme; elle mesure son degré de tolérance aux erreurs des utilisateurs et sa résistance aux attaques des pirates; un programme est plus robuste qu'un autre s'il résiste mieux aux erreurs de manipulations des utilisateurs plus ou moins bien attentionnés.

Il est à noter qu'il n'y a pas de méthode ou d'échelle de mesure permettant d'évaluer la fiabilité ou la robustesse d'un algorithme. C'est à l'usage que ces qualités sont mesurées. Par contre, il existe des méthodes rationnelles et rigoureuses pour évaluer l'efficacité en temps ou en espace

d'un algorithme. Ces méthodes d'évaluation portent le nom d'analyse de complexité des algorithmes.

Deux types de complexité peuvent être cités:

- *la complexité méthodologique*, qui exprime une fonction du nombre d'opérations élémentaires de calcul effectuées par la méthode ou par l'algorithme de résolution en fonction du nombre des données du problème traité,
- *la complexité problématique*, liée à la difficulté du problème à résoudre et au nombre des opérations élémentaires qu'un algorithme déterministe peut effectuer pour trouver l'optimum en fonction de la taille du problème.

Selon son degré de complexité, un problème peut appartenir à l'une des quatre classes suivantes [Sakarovitch 84] :

- *les problèmes les plus difficiles*, qui sont des problèmes pour lesquels il n'existe aucune méthode de résolution; ils sont dits indécidables,
- *les problèmes de la classe P*, dits polynomiaux, s'il existe un algorithme de complexité polynomiale pour leur résolution,
- *les problèmes de la classe NP*, dits problèmes NP-difficiles, qui ne peuvent à priori être résolus en un temps polynomial que par des méthodes approchées (heuristiques); au cours de leur exécution, ces algorithmes font des choix dont l'optimalité n'est pas démontrable,
- *les problèmes NP-Complets*, qui répondent à la définition suivante : un problème de décision A est dit NP-Complet s'il appartient à la classe NP et si pour tout A′ de NP :
  - il existe une application polynomiale qui transforme toute instance I′ de A′ en une instance I de A,
  - A′ admet une réponse "oui" pour l'instance I′, si et seulement si A admet une réponse "oui" pour l'instance I.

  Autrement dit, s'il existe un algorithme polynomial pour résoudre A, alors, pour tout le reste des problèmes de la classe, il existe des algorithmes polynomiaux pour les résoudre.

I.7 - Méthodes d'optimisation

Etant donnés un ensemble de tâches et un ensemble de ressources, il est nécessaire de programmer les tâches et d'affecter les ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), en respectant un ensemble de contraintes. La principale difficulté à laquelle est confronté un décideur, en présence d'un problème d'optimisation est celui du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable.
Les différentes méthodes de résolution développées peuvent être classées en deux catégories : les méthodes exactes qui garantissent la complétude de la résolution et les méthodes approchées qui perdent la complétude pour gagner en efficacité.

I.7.1 - Les méthodes exactes

On peut définir une méthode exacte comme étant une méthode qui fournit une solution optimale pour un problème d'optimisation.
L'utilisation de ce type de méthodes s'avère particulièrement intéressante dans les cas des problèmes de petites tailles. La méthode par séparation et évaluation (branch and bound) [Le Pape, 95], [Baptiste, 96] constituent certainement celles qui sont les plus utilisées pour résoudre les problèmes d'optimisation multi-objectifs [Sakarovitch, 84]. D'autres méthodes telles que la programmation linéaire ou la programmation dynamique, sont aussi utilisées couramment.
Toutes ces méthodes examinent d'une manière implicite, la totalité de l'espace de recherche pour produire la solution optimale.

a - La méthode Branch and Bound

L'algorithme Branch and Bound consiste à placer progressivement les tâches sur les ressources en explorant un arbre de recherche décrivant toutes les combinaisons possibles.
Il s'agit de trouver la meilleure configuration donnée de manière à élaguer les branches de l'arbre qui conduisent à de mauvaises solutions.
L'algorithme branch and bound effectue une recherche complète de l'espace des solutions d'un problème donné, pour trouver la meilleure solution.
La démarche de l'algorithme Branch and Bound consiste à [Collette et al, 02] :

- diviser l'espace de recherche en sous espaces,
- chercher une borne minimale en terme de fonction objectif associée à chaque sous espace de recherche,
- éliminer les mauvais sous-espaces,
- reproduire les étapes précédentes jusqu'à l'obtention de l'optimum global.

b - La programmation dynamique

Elle se base sur le principe de Bellman [Bellman, 86] : « Si C est un point qui appartient au chemin optimal entre A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C ». C'est une méthode qui consiste donc à construire d'abord les sous-chemins optimaux et ensuite par récurrence le chemin optimal pour le problème entier. Cette méthode est destinée à résoudre des problèmes d'optimisation à vocation plus générale que la méthode de séparation et d'évaluation (branch and bound) sans permettre pour autant d'aborder des problèmes de tailles importantes.

c - La programmation linéaire

C'est l'une des techniques classiques de recherche opérationnelle. Elle repose sur la méthode du simplexe et les algorithmes de points intérieurs de Karmarkar [Sakarovitch, 84].
Elle consiste à minimiser une fonction coût en respectant des contraintes, le critère et les contraintes étant des fonctions linéaires des variables du problème [Mellouli et al, 04].

d - Les heuristiques

Les heuristiques sont des méthodes empiriques qui donnent généralement de bons résultats sans pour autant être démontrables. Elles se basent sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable [Bel, 01]. Parmi ces stratégies, nous distinguons :

- FIFO (First In First Out) où la première tâche arrivée est la première à être ordonnancée,
- SPT (Shortest Processing Time) où la tâche ayant le temps opératoire le plus court est traitée en premier,

- LPT (Longest Processing Time) où la tâche ayant le temps opératoire le plus important est traitée en premier,
- EDD (Earliest Due Date) où la tâche ayant la date due la plus petite est la plus prioritaire, …

I.7.2 - Les méthodes approchées ou métaheuristiques

Malgré l'évolution permanente de l'informatique, il existe toujours, pour un problème polynomial, une taille critique au-dessus de laquelle une énumération, même partielle, des solutions admissibles devient prohibitive. Compte tenu de ces difficultés, la plupart des spécialistes de l'optimisation combinatoire ont orienté leurs recherches vers le développement des métaheuristiques [Widmer, 01]. Une métaheuristique est souvent définie comme une procédure exploitant au mieux la structure du problème considéré, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [Nicholson, 71].

Les métaheuristiques sont ainsi des méthodes de recherche générales, dédiées aux problèmes d'optimisation difficile. Elles sont, en général, présentées sous forme de concepts.

Les principales métaheuristiques sont celles basées sur la recherche locale, telles que le recuit simulé et la recherche Tabou, et celles basées sur les algorithmes évolutionnistes telles que les algorithmes génétiques ainsi que les algorithmes basés sur la recherche globale tels que les algorithmes de colonies de fourmis et les algorithmes reposant sur la méthode d'optimisation par essaim particulaire.

a - Les méthodes basées sur la recherche locale

La recherche locale peut être résumée comme étant une procédure de recherche itérative qui, à partir d'une première solution réalisable, l'améliore progressivement en appliquant une série de modifications (ou mouvements) locales, comme montré dans la figure 1.4. Il faut, pour cela introduire une structure de voisinage qui consiste à spécifier un voisinage pour chaque solution. Ainsi, à chaque itération, la recherche s'oriente vers une nouvelle solution réalisable qui diffère légèrement de la solution courante en remplaçant celle-ci par une meilleure située dans son voisinage. La recherche se termine si un optimum local est rencontré. L'inconvénient important de cette méthode est qu'à moins d'être extrêmement chanceux, cet optimum local est souvent une solution assez médiocre. Dans la recherche locale, la qualité des solutions

obtenues dépend fortement de la richesse de l'ensemble des transformations (mouvements) considérées à chaque itération.

Pour faire face à cette limitation, des méthodes de recherche locale plus sophistiquées ont été développées au cours de ces vingt dernières années. Ces méthodes acceptent des solutions voisines moins bonnes que la solution courante afin d'échapper aux minima locaux. En règle générale, seule une portion du voisinage courant est explorée à chaque étape [Widmer, 01]. Les méthodes les plus connues sont le recuit simulé [Kirkpatrick et al, 83] et la recherche tabou [Glover, 89], [Glover, 90].

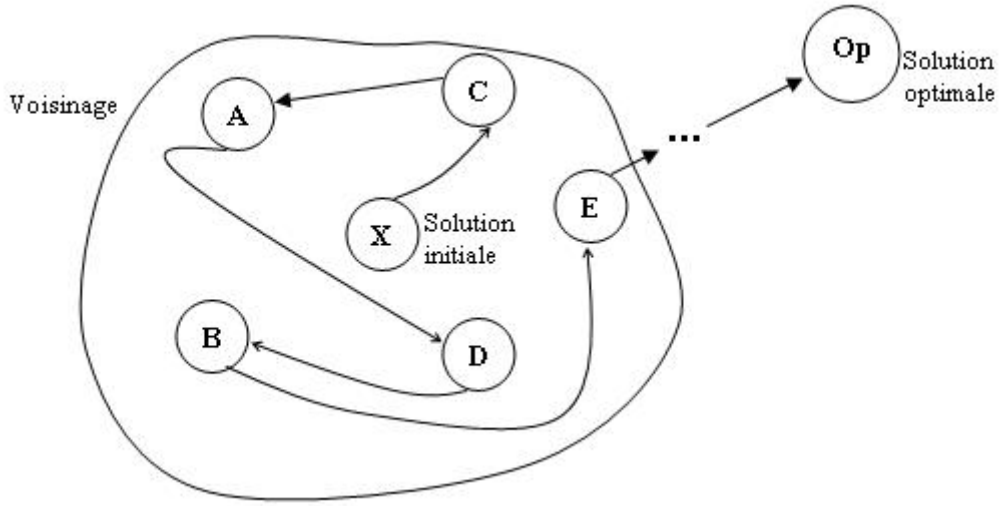

Figure 1. 4 - Exploration de l'espace de recherche dans la méthode de recherche locale

- **Le recuit simulé**

Inspiré du recuit physique, ce processus est utilisé en métallurgie pour améliorer la qualité d'un solide et cherche un état d'énergie minimale qui correspond à une structure stable du solide. Ainsi, pour qu'un métal retrouve une structure proche du cristal parfait, on porte celui-ci à une température élevée, puis on le laisse refroidir lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement.

L'algorithme du recuit simulé [Kirkpatrick et al, 83], permet de résoudre les problèmes de minima locaux. En effet, une nouvelle solution de coût supérieur à celui de la solution courante ne sera pas forcément rejetée, son acceptation sera déterminée aléatoirement en tenant compte de la différence entre les coûts ainsi que du facteur température *T*. Ce paramètre, sert à prendre en compte le fait que plus le processus d'optimisation est avancé, moins on est près à accepter une solution plus coûteuse. Par contre, l'acceptation de solutions

fortement coûteuses permet, au début, de mieux explorer l'espace des solutions possibles et ainsi, d'accroître les chances d'approcher le minimum global.

Kirkpatrick [Kirkpatrick et al, 83] et Cerny [Cerny, 85] se sont inspirés d'une telle technique pour résoudre des problèmes d'optimisation combinatoire. Le voisinage *N(s)* d'une solution, s'apparente à l'ensemble des états atteignables depuis l'état courant, en faisant subir des déplacements aux atomes du système physique.

A chaque itération, une seule solution voisine est générée. Celle-ci est acceptée si elle est meilleure que la solution courante. Dans le cas contraire, la nouvelle solution est acceptée avec une certaine probabilité qui dépend de l'importance de la détérioration et du paramètre *T* correspondant à la température. En règle générale, la température est diminuée par paliers, à chaque fois qu'un certain nombre d'itérations est effectué. La meilleure solution trouvée est mémorisée. L'algorithme est interrompu lorsqu'aucune solution voisine n'a été acceptée pendant un cycle complet d'itérations à température constante [Widmer, 01].

De nombreuses études ont été effectuées sur la méthode du recuit simulé [Collins et al, 88], [Osman et Christofides, 1994]. En optimisation, le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coûts plus faibles tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût [Collette, 02], [Hao, 99]. Ainsi, si une amélioration du critère est constatée, le nouvel état est retenu, sinon une diminution ΔE du critère E est calculée et le nouvel état est retenu si p étant un nombre tiré de façon aléatoire entre 0 et 1, on a $\exp(-\Delta E/T) > p$.

Un algorithme d'optimisation par recuit simulé se décompose selon les étapes suivantes [Laquerbe et al, 98] :

- choix d'une fonction à optimiser,
- adoption d'un schéma de recuit dans lequel sont précisés la température initiale, le nombre de configurations générées à chaque température et le schéma de décroissance du critère,
- génération stochastique de configurations voisines, correspondant aux transitions,
- choix d'un critère d'acceptation.

Cet algorithme est présenté dans la figure 1.5.

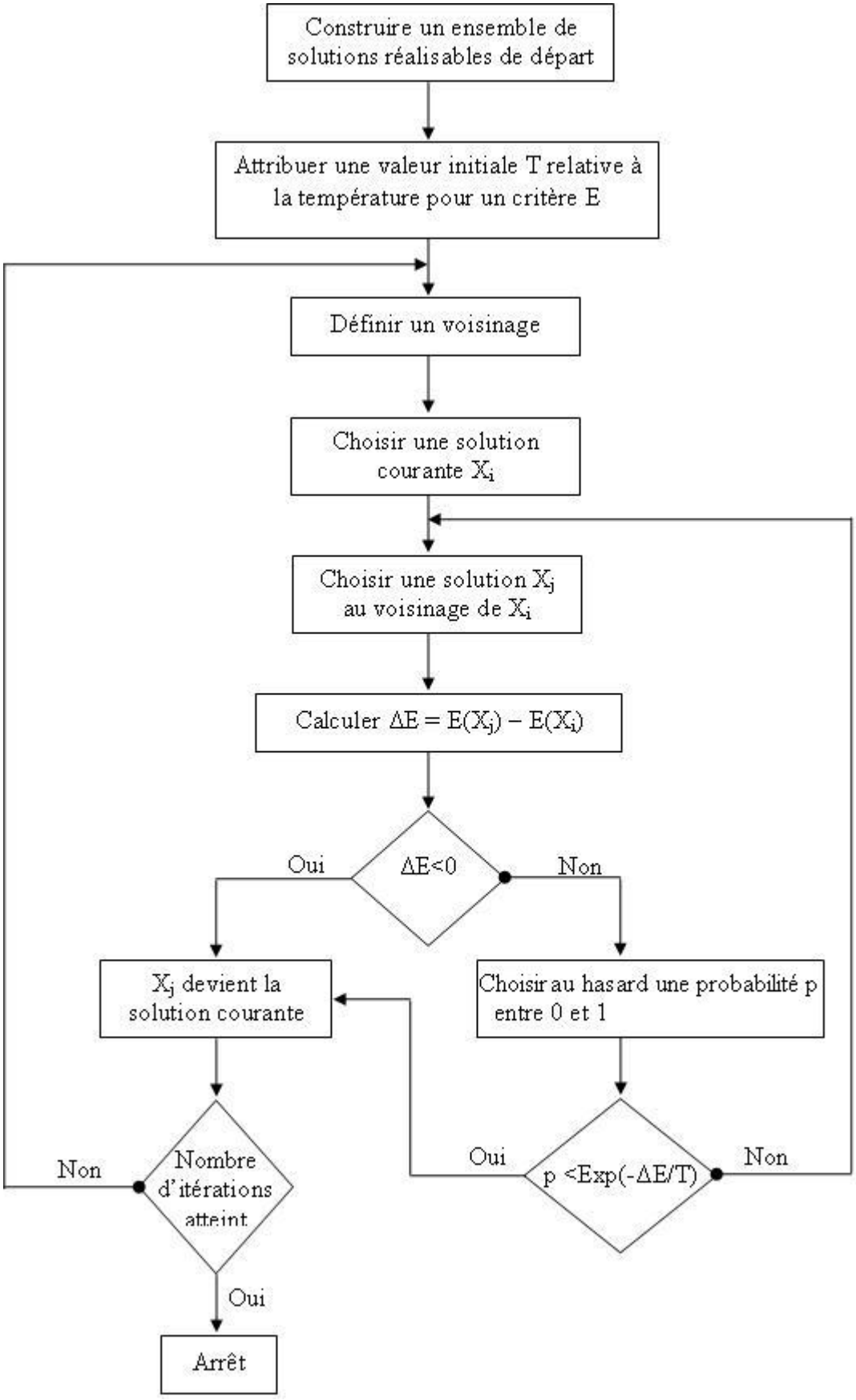

Figure 1. 5 - Algorithme relatif au fonctionnement général du recuit simulé

- **La recherche tabou**

Bien que son origine remonte à 1977, la recherche tabou n'est proposée qu'au milieu des années 80 par Fred Glover [Glover, 89], [Glover, 90]. Cette méthode, développée pour résoudre des problèmes combinatoires, la plupart NP-difficiles, propose de surmonter le problème des optima locaux par l'utilisation d'une mémoire.

La méthode tabou est une procédure itérative qui, partant d'une solution initiale, tente de converger vers la solution optimale en exécutant, à chaque pas, un mouvement dans l'espace de recherche. Chaque pas consiste d'abord à engendrer un ensemble de solutions voisines de la solution courante pour ensuite en choisir la meilleure, même si ce choix entraîne une augmentation de la fonction objectif à minimiser.

En acceptant de détériorer la valeur de la solution courante, le minimum local peut être évité mais, en contre partie, des parcours répétitifs sont déplorés.

Aussi, pour palier à l'inconvénient majeur des méthodes de recherche locale, la recherche tabou a pour but d'améliorer à chaque étape, la valeur de la fonction objectif, en utilisant une mémoire afin de conserver les informations sur les solutions déjà visitées.

Cette mémoire constitue la *liste Tabou* qui va servir à interdire l'accès aux dernières solutions visitées. Lorsqu'un optimum local est atteint, il y a interdiction de revenir sur le même chemin.

Un *critère d'aspiration,* est également utilisé pour lever l'interdiction d'utilisation d'un mouvement si ce dernier conduit à une meilleure solution.

Plusieurs stratégies ont été proposées récemment afin d'améliorer l'efficacité de la méthode tabou. L'intensification et la diversification de la recherche constituent deux d'entre elles [Widmer, 01].

- *L'intensification* consiste à explorer en détails une région de l'espace de recherche jugée prometteuse. Sa mise en œuvre consiste, le plus souvent, en un élargissement temporaire du voisinage de la solution courante dans le but de visiter un ensemble de solutions partageant certaines propriétés.
- *La diversification* a pour objectif de diriger la procédure de recherche vers des régions inexplorées de l'espace de recherche. La stratégie de diversification la plus simple consiste à redémarrer périodiquement le processus de recherche à partir d'une solution, générée aléatoirement ou choisie judicieusement, dans une région non encore visitée de l'ensemble des solutions admissibles.

Les domaines d'application de la recherche tabou sont vastes et variés, ils passent de l'ordonnancement à la robotique, au problème du voyageur de commerce, à l'électronique voire même aux applications médicales, … En tenant compte des notations suivantes, les étapes d'évolution de l'algorithme sont présentées dans la figure 1.6.

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

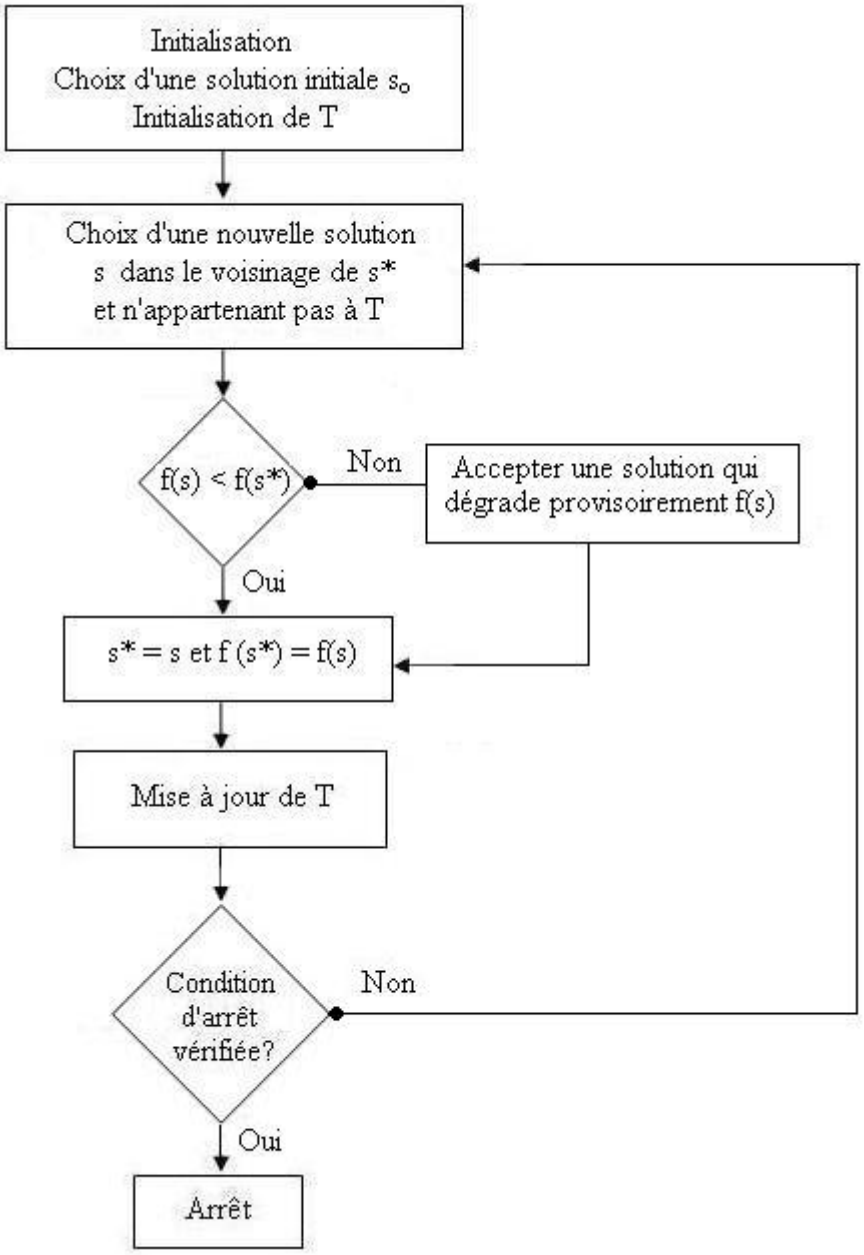

Figure 1. 6 - Algorithme relatif au fonctionnement général de la méthode de recherche tabou

b - Les algorithmes évolutionnistes : algorithmes génétiques

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

En règle générale, la taille de la population reste constante tout au long du processus. Après avoir généré une population initiale de solutions, une méthode évolutive tente d'améliorer la qualité moyenne de la population courante en ayant recours à des principes d'évolution naturelle [Widmer, 01].

Parmi les méthodes évolutionnistes, nous citons les algorithmes génétiques.

Les Algorithmes Génétiques (A.G.) [Holland, 75] sont des algorithmes itératifs dont le but est d'optimiser une fonction prédéfinie, appelée fitness.

Pour réaliser cet objectif, l'algorithme travaille sur un ensemble de points, appelés population d'individus. Chaque individu ou chromosome (chaîne binaire de longueur finie dans les premières définitions) représente une solution possible du problème donné. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles.

L'utilisation d'un algorithme génétique nécessite la définition, au préalable, d'un espace de recherche dont les éléments de base sont les chromosomes et d'une fonction définie sur cet espace (fonction fitness) dont la valeur optimale est évaluée en rapport avec les opérateurs de croisement et de mutation choisis [Iyer, 04].

Cinq éléments de base sont nécessaires pour l'utilisation des algorithmes génétiques [Durand et al, 94] :

- *un principe de codage* des éléments de la population, qui consiste à associer à chacun des points de l'espace d'état une structure de données, la qualité de ce codage des données conditionnant le succès des algorithmes génétiques; bien que le codage binaire ait été très utilisé à l'origine, les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles,
- *un mécanisme de génération* de la population initiale qui doit être capable de produire une population d'individus non homogène servant de base pour les générations futures;

le choix de la population initiale est important car il influence la rapidité de la convergence vers l'optimum global; dans le cas où l'on ne dispose que de peu d'informations sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche,

- *une fonction à optimiser,* appelée *fitness* ou fonction d'évaluation de l'individu,
- *des opérateurs* permettant de diversifier la population au cours des générations et d'explorer l'espace d'état; l'opérateur de croisement recompose les gènes d'individus existant dans la population alors que l'opérateur de mutation garantit l'exploration de l'espace d'état,
- *des paramètres de dimensionnement*, représentés par la taille de la population, le nombre total de générations, ou le critère d'arrêt, ainsi que les probabilités d'application des opérateurs de croisement et de mutation.

L'enchaînement de ces différents éléments est représenté dans la figure 1.6. Cette méthode sera présentée plus en détail dans le second chapitre de ce mémoire.

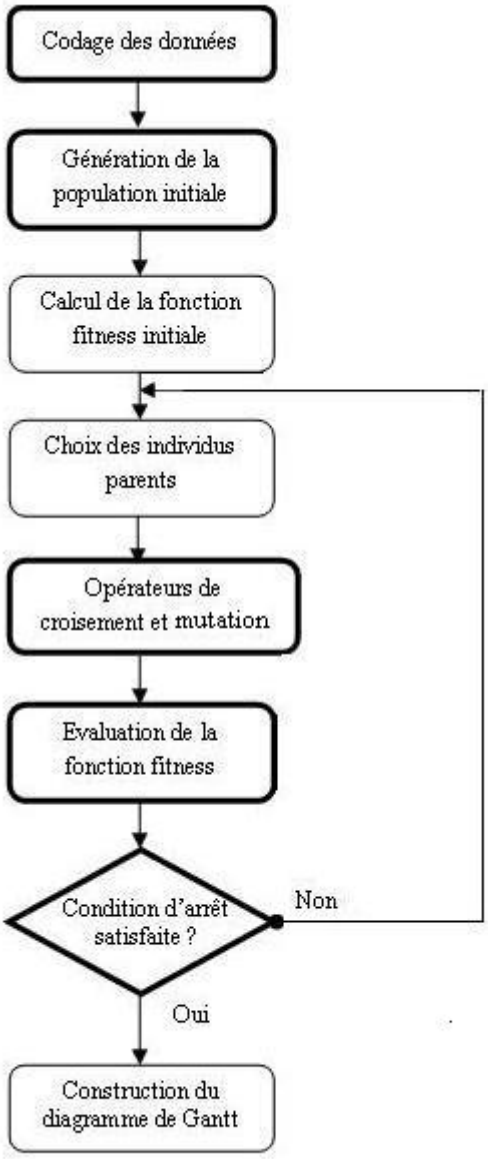

Figure 1. 7 - Fonctionnement général d'un algorithme génétique

c - Les algorithmes de recherche globale

- **Les algorithmes de colonies de fourmis**

Ces algorithmes sont nés à la suite de constatations faites sur le comportement des fourmis qui sont capables de trouver le chemin le plus court, du nid à une source de nourriture, et de s'adapter aux changements de l'environnement.

Les biologistes ont, ainsi, étudié comment les fourmis arrivent à résoudre collectivement des problèmes trop complexes pour un seul individu, notamment les problèmes de choix lors de

l'exploitation des sources de nourriture et ceci grâce à la phéromone (substance leur permettant de laisser une trace sur leur chemin), [Dorigo et al, 97].

Ces algorithmes ont donc pour but de reproduire le comportement naturel des fourmis pour retrouver le meilleur chemin possible vers un objectif donné.

L'exemple présenté dans les figures 1.8, 1.9 et 1.10, montre la capacité de cet algorithme à se rapprocher de la solution optimale.

- Au départ, les fourmis disposent d'un chemin pour aller de leur nid, la fourmilière, à la source de nourriture, figure 1.8.

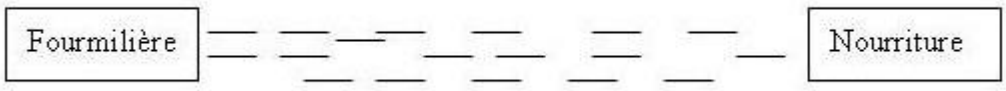

Figure 1. 8 - Déplacement des fourmis vers une source de nourriture

- Un obstacle est placé entre le nid et la source de nourriture, les fourmis vont alors commencer par se séparer de part et d'autre de l'obstacle en déposant de la phéromone sur leur passage, figure 1.9. Les fourmis les plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui ont emprunté les deux branches les plus courtes à l'aller et au retour. Ainsi la quantité de phéromone présente sur le plus court trajet est plus importante que celle présente sur le chemin le plus long.

  Or, une piste présentant une plus grande concentration en phéromone, étant plus attirante pour les fourmis, a donc une probabilité plus grande d'être empruntée [Tangour et al, 09].

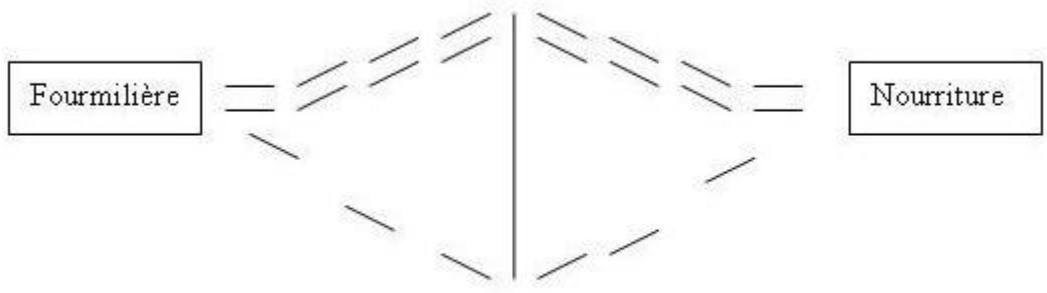

Figure 1. 9 - Déplacement des fourmis après placement d'un obstacle sur leur chemin

- Finalement la quantité de phéromone sur le chemin le plus court fait que ce chemin finit par être emprunté par la plupart les fourmis, figure 1.10.

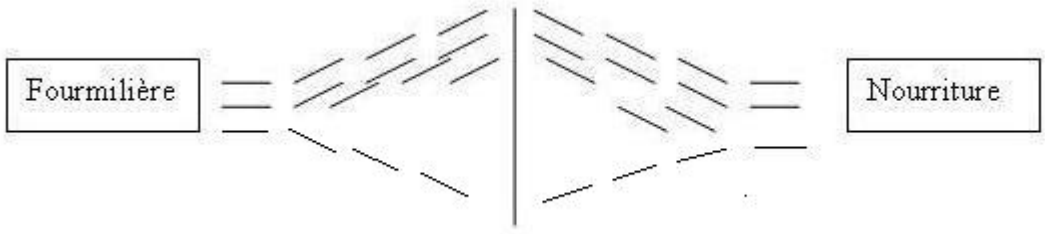

Figure 1. 10 - Choix du chemin le plus court par la plupart des fourmis

- **Les algorithmes d'optimisation par essaim particulaire**

La méthode d'Optimisation par Essaim Particulaire (OEP) a été proposée en 1995 par James Kennedy et Russel Eberhart [Kennedy et al, 95] qui cherchaient à simuler la capacité des oiseaux à voler de façon synchrone et leur aptitude à changer brusquement de direction, tout en restant en formation optimale.

Le fonctionnement de l'OEP fait qu'elle peut être classée parmi les méthodes itératives (approche progressive de la solution) et stochastiques (faisant appel au hasard) dans le but d'améliorer la situation existante en se déplaçant partiellement au hasard et partiellement selon des règles prédéfinies, en vue d'atteindre la solution globale souhaitée [Clerc et al, 08].

La méthode d'optimisation par essaim particulaire, est une procédure de recherche basée sur une population d'individus, appelés particules, qui changent leur position (état) avec le temps. Dans un système d'OEP, les particules se déplacent à l'intérieur d'un espace de recherche. Pendant le déplacement, chaque particule ajuste sa position selon sa propre expérience, et selon l'expérience des particules voisines, se servant de sa meilleure position produite et de celle de ses voisines. Ce comportement est semblable à celui du comportement humain consistant à prendre des décisions où les individus considèrent leur expérience antérieure et celle des personnes qui les entourent [Kennedy et al, 95]. L'OEP peut ainsi combiner des méthodes de recherche locale avec des méthodes de recherche globale (métaheuristiques).

Même si beaucoup de similitudes existent entre les méthodes évolutionnaires et l'OEP, cette dernière se distingue par le fait qu'elle n'utilise pas l'opérateur de sélection qui choisit les individus gardés dans la prochaine génération.

En effet, tous les membres de la population sont maintenus par le procédé de recherche de sorte que l'information soit toujours mise en commun entre les individus pour diriger la recherche vers les meilleures positions trouvées dans l'espace de recherche [Deroussi, 06].
Cette méthode est exposée plus en détails dans le troisième chapitre de ce mémoire.

I.8 - Position du problème

La résolution des problèmes rencontrés pour la gestion de leur poste de production, constitue une des principales préoccupations des industriels. En effet, comment réussir à produire le plus possible, le rapidement possible avec le moins de coût possible, le moins d'arrêts possible, le moins de personnel possible tout en respectant toutes les contraintes liées au produit (précédence, achèvement, périssabilité, …)?
Notre travail rentre dans ce cadre et consiste donc à développer des outils d'aide à l'ordonnancement permettant la gestion du poste de production et en particulier celui du conditionnement au niveau des industries pharmaceutiques. Il s'agit donc, de prendre en considération les différentes contraintes relatives aux ressources et aux produits nécessaires ainsi qu'au temps et ceci pour réaliser une optimisation multi-objectifs reposant sur deux objectifs principaux relatifs d'une part à la minimisation des coûts de fabrication et aux coûts des temps engendrés par les opérations de nettoyage ainsi que les temps d'arrêt et aux coûts de non utilisation des ressources, d'autre part.
Pour cela, deux méthodes sont utilisées et comparées: les algorithmes génétiques et les algorithmes d'optimisation par essaim particulaire.

I.9 - Conclusion

Après la présentation des problèmes d'ordonnancement et de leurs principales caractérisations, différentes méthodes, exactes et approchées, pouvant être utilisées pour la résolution de ces problèmes sont introduites dans ce chapitre.
Parmi les méthodes exactes, nous avons distingué la programmations linéaire, la programmation dynamique et la méthode branch and bound.
Parmi les méthodes approchées ou métaheuristiques, nous avons présenté les méthodes de recherche locale telles que le recuit simulé ou la méthode de recherche tabou, et les méthodes évolutionnistes, telles que les algorithmes génétiques, les algorithmes de colonies de fourmis ou les algorithmes d'optimisation par essaim particulaire, récemment proposés.

La résolution d'un problème d'ordonnancement en industries pharmaceutiques par l'utilisation de la méthode des algorithmes génétiques est proposée dans le prochain chapitre et les résultats obtenus seront comparés avec ceux fournis par la méthode d'optimisation par essaim particulaire présentée dans le troisième chapitre.

Chapitre II

Algorithmes génétiques pour la résolution de problèmes d'ordonnancement en industries pharmaceutiques

II.1 - Introduction

La résolution de problèmes d'optimisation consiste, en règle générale, à définir une ou plusieurs fonctions objectifs (fonctions coûts) à minimiser ou maximiser selon le type de problème traité.

La définition du problème d'optimisation est souvent complétée par la donnée de contraintes et toutes les solutions retenues doivent respecter ces contraintes, qui peuvent aussi être vues comme définissant l'espace de recherche [Clerc et al, 04].

Les métaheuristiques permettent de trouver une ou plusieurs solutions proches de l'optimum pour des problèmes d'optimisations, en tenant compte des fonctions objectifs définies. Le principe d'une métaheuristique est de minimiser ou de maximiser ces fonctions, de trouver un minimum global à un problème de minimisation et de ne pas rester bloqué sur un minimum local.

Ainsi, les métaheuristiques ont pour ambition commune de résoudre au mieux les problèmes d'optimisation qualifiés de difficiles.

Quelque soit la méthode utilisée, le choix des critères à optimiser continue à poser de sérieux problèmes. Dans une grande partie de la littérature, la résolution des problèmes

d'ordonnancement traite surtout le cas mono-objectif. Seulement, de nos jours, différents objectifs peuvent être pris en considération et gérés simultanément pour une meilleure optimisation [Chang et al, 07], [Saad et al, 06], [Tangour et al, 06].

Dans ce chapitre, un problème d'ordonnancement multi-objectifs d'un atelier de conditionnement de type flow-shop en industries pharmaceutiques est présenté. Les différentes caractéristiques et opérations le concernant sont introduites puis sa formulation et sa résolution par les algorithmes génétiques proposées.

II.2 - Ordonnancement en industries pharmaceutiques

Pour pouvoir faire face à la concurrence, les industries pharmaceutiques doivent maintenir un taux de productivité en constante évolution; des faiblesses peuvent toutefois apparaître au niveau du système de production et plus particulièrement au niveau du poste de conditionnement et engendrer des coûts de production ainsi que des coûts de non utilisation relativement élevés. Pour y pallier et résoudre le problème d'ordonnancement au niveau de ce poste, plusieurs méthodes d'optimisation peuvent être envisagées, parmi lesquelles, on trouve les algorithmes génétiques.

II.2.1 - Types de produits utilisés dans les industries pharmaceutiques

Pour obtenir la boîte de médicaments parvenant au consommateur, le produit passe par les formes suivantes :

- les matières premières composées du principe actif (mélange ou composé extrait d'une substance végétale, animale ou de synthèse qui confère au médicament son action pharmacologique), de l'excipient (substance neutre dans laquelle est incorporé un médicament pour permettre son absorption) et parfois d'un colorant,
- les produits semi-finis qui résultent de la transformation des matières premières en comprimés ou en gélules,
- les articles de conditionnement primaires, en contact direct avec le semi-fini, tels que les bobines d'aluminium ou de PVC, et les articles de conditionnement secondaires tels que les notices, les vignettes et les étuis en carton,
- les produits finis qui sont mis à disposition du consommateur.

II.2.2 - Cheminement des produits au niveau des industries pharmaceutiques

Les matières premières et les articles de conditionnement primaires et secondaires arrivent au magasin et sont déposés dans une zone de réception, étiquetés puis acheminés vers une zone de quarantaine en l'attente d'être contrôlés par le service contrôle qualité. S'ils sont acceptés, les produits sont alors stockés dans une zone portant la mention «accepté» avec une étiquette avec la même mention ; sinon, ils sont déposés dans une zone dite «refusé».

Suivant les plannings établis, les matières premières et les articles de conditionnement sont livrés par un magasinier au service production en vue d'être traités.

- Les matières premières, portant l'étiquette «accepté», sont acheminées vers le service des formes solides.
- Les articles de conditionnement, portant l'étiquette «accepté», sont, au besoin, directement envoyés au service conditionnement.
- Les produits semi-finis, résultant de la transformation des matières premières au niveau de l'atelier des formes solides, sont envoyés, en fonction du planning, au service conditionnement.

Après avoir été traités au service conditionnement, les produits finis résultant de la transformation des semi-finis et des articles de conditionnement, sont envoyés au magasin, en attendant d'être transmis aux particuliers (pharmaciens, délégués médicaux) ou aux hôpitaux.

II.2.3 - Spécificités d'un atelier de conditionnement

Un atelier de conditionnement de produits pharmaceutiques peut contenir une ou plusieurs lignes de conditionnement automatique, qui sont exploitées pour transformer les produits semi-finis, en utilisant les articles de conditionnement, en produits finis.

L'atelier étudié comporte :

- deux chaînes de conditionnement automatique servant pour l'emballage des comprimés et gélules,
- un tapis de conditionnement de produits,
- une encartonneuse semi-automatique,
- une ligne de vignettage à plat des étuis,
- un local de préparation, dans lequel on trouve une plieuse de notices et un local d'outillage,

- une laverie, où s'effectuent le nettoyage du matériel et les tests d'étanchéité des blisters,
- une zone de pré-stockage de produits semi-finis et semi-conditionnés.

II.2.4 - Lignes de conditionnement

Au niveau de l'atelier considéré, le conditionnement des comprimés et des gélules est effectué essentiellement sur deux lignes de conditionnement.

Une ligne de conditionnement de comprimés et de gélules est constituée de quatre machines, comme le montre la figure 2.1 : une blistéreuse, une encartonneuse, une vignetteuse et une fardeleuse [Boukef et al, 06].

- *La blistéreuse* traîte le film PVC en le passant sous une plaque chauffante, permettant de former de petites alvéoles (thermoformage) qui sont alimentées par un chargeur contenant les produits semi-finis. Une autre plaque chauffante a pour rôle de sceller la feuille d'aluminium au-dessus ; les blisters sont ensuite coupés et envoyés vers l'encartonneuse.
- *L'encartonneuse* reçoit les quantités de blisters nécessaires selon le format et les introduit dans l'emballage en même temps que la notice.
- *La vignetteuse* reçoit les blisters dans leurs boîtes et colle dessus une vignette imprimée.
- *La fardeleuse* regroupe les produits finis en lots ; un opérateur humain se charge de remplir les caisses avec le nombre de produits finis, fixé à l'avance.

Pour chacune des lignes de conditionnement, les produits passent par les quatre machines (en série) dans le même ordre. Il s'agit donc d'un atelier de type flow-shop.

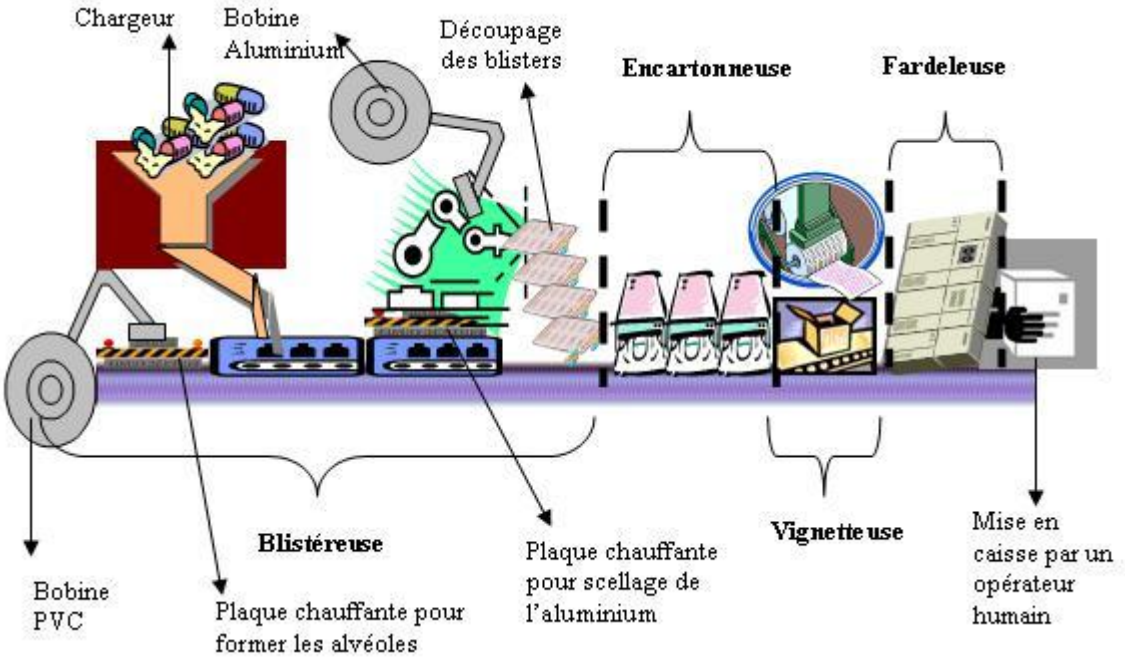

Figure 2. 1 - Machines composant une ligne de conditionnement

II.2.5 - Problèmes survenant dans un atelier de conditionnement

Plusieurs opérations de nettoyage et de changement de format sont à gérer sur le poste de conditionnement, conjointement aux opérations de production.

Les temps improductifs générés par ces opérations sont assez importants, compte tenu du fait que le temps de lancement de la fabrication d'un produit dépend de celui qui l'a précédé.

D'un produit à un autre, le changement entraîne, ainsi, certaines modifications au niveau de chacune des machines [Boukef et al, 06].

- *Au niveau de la blistéreuse*, il peut y avoir des changements de chargeur, de brosse, de format, de bobines PVC et Aluminium, etc.
- *Au niveau de l'encartonneuse*, les changements consistent en des réglages assez fins qui permettent de passer d'un format d'étui à un autre.
- *Au niveau de la vignetteuse*, il s'agit de modifier l'édition de la vignette concernant le nom du produit, le prix, le numéro de lot, la date limite de sa consommation, etc.
- *Au niveau de la fardeleuse***,** et suivant les changements effectués au niveau de l'encartonneuse, des réglages sont envisageables pour agrandir ou rétrécir la taille du fardeau.

Tous les arrêts et les changements survenant au cours d'une opération doivent être pris en compte pour permettre un bon ordonnancement du poste de conditionnement.

II.3 - Problèmes d'ordonnancement de type flow-shop

Les problèmes d'ordonnancement d'ateliers de type flow-shop, ou ateliers à cheminements uniques, sont parmi les problèmes les plus connus dans le domaine de l'ordonnancement. Ils ont fait l'objet de nombreuses études dont l'objectif principal est la minimisation de la date de fin d'exécution [Negenman, 01], [Breit, 06], [Ruiz et al, 06].
Le problème d'atelier flow-shop auquel nous nous intéressons est traité sous une forme multi-critères où les objectifs sont la minimisation des coûts de fabrication ainsi que la minimisation des différents coûts d'arrêt et de non utilisation des lignes de conditionnement.

II.3.1 - Présentation des ateliers de type flow-shop

Un problème d'ordonnancement flow-shop met en œuvre m machines, notées $M1, M2,..., Mm$, et n produits (tâches), notées $p1, p2,..., pn$. Chaque produit (tâche) est exécuté(e) sur la machine $M1$, puis sur $M2$, et ainsi de suite, jusqu'à ce qu'à la dernière machine. A chaque instant, une machine traite au plus un produit à la fois et chaque produit est exécuté par au plus une machine.
La figure 2.2 montre l'ordre de cheminement des produits dans un atelier classique de type flow-shop.

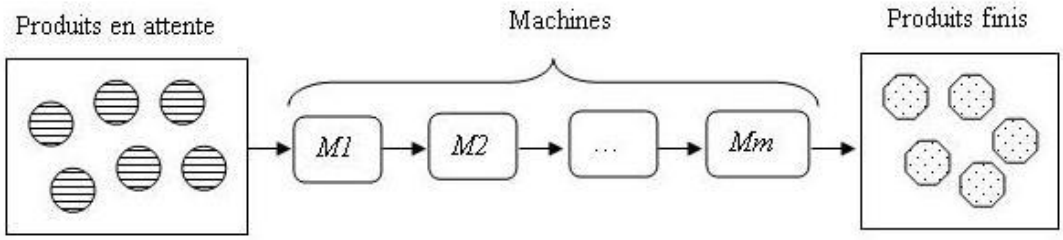

Figure 2. 2 - Cheminement des produits dans un atelier de type flow-shop

II.3.2 - Ordonnancement d'ateliers de type flow-shop

La résolution des problèmes d'ordonnancement de type flow-shop a connu plusieurs étapes. Depuis les travaux de Johnson [Johnson, 54], l'objectif le plus visé, dans la résolution de ces problèmes d'ordonnancement est de réduire au maximum le Makespan ou Cmax, temps de fin de fabrication du dernier produit.

Diverses approches heuristiques [Dannenbring, 77], [Nawaz, 83], [Osman, 89] et [Widmer et al, 90] ont été proposées pour réduire le Makespan [Murata, 96].

Alors que ces études ont eu pour rôle de traiter un seul objectif, beaucoup de problèmes réels impliquant des objectifs multiples ont été mis de côté.

Récemment, plusieurs travaux de recherche ont abordé les problèmes d'ordonnancement multi-objectifs de type flow-shop [Ho et al, 91], [Gangadhran, et al, 94], [Mabed et al, 01], [Bertel et al, 01], [Neumann et al, 05], [Onwubolu et al, 06].

Nous nous proposons de résoudre le problème d'ordonnancement en industries pharmaceutiques d'un point de vue multi-objectifs tout en énonçant les différents critères et contraintes s'y rattachant et en utilisant les algorithmes génétiques comme méthode de résolution.

II.4 - Optimisation mono-objectif / Optimisation multi-objectifs

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

II.4.1 - Optimisation mono-objectif

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

$$\begin{cases} \text{minimiser } f(\vec{x}) & \text{(fonction à optimiser)} & \vec{x} \in \Re^n, f(\vec{x}) \in \Re \\ \text{avec } \vec{g}(\vec{x}) \le 0 & \text{(m contraintes inégalités)} & \vec{g}(\vec{x}) \in \Re^m \\ \text{et } \vec{h}(\vec{x}) = 0 & \text{(p contraintes égalités)} & \vec{h}(\vec{x}) \in \Re^p \end{cases}$$

Il existe deux types de minima : les minima locaux et les minima globaux.

- *Minimum global*

Un point $\vec{x}*$ est un minimum global de la fonction f, si on a :

$f(\vec{x}*) \prec f(x) \ \forall \vec{x}$, tel que $\vec{x}* \neq \vec{x}$

- *Minimum local*

Un point $\vec{x}*$ est un minimum local de la fonction f, si on a :

$f(x*) \leq f(x) \ \forall \vec{x} \in V(\vec{x}*)$ et $\vec{x}* \neq \vec{x}$

$V(\vec{x}*)$ définissant un voisinage de $\vec{x}*$

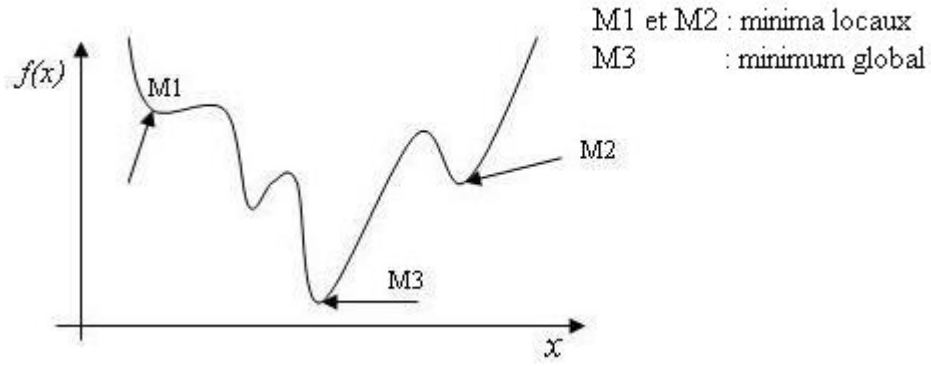

Figure 2. 3 - Types de minima

Les objectifs les plus régulièrement considérés sont la minimisation de la durée totale de l'ordonnancement, le respect des dates au plus tard, la minimisation des retards, la minimisation d'un coût, la minimisation de la date effective de fin de toutes les opérations, …

II.4.2 - Optimisation multi-objectifs

D'un point de vue mathématique, un problème d'optimisation multi-objectifs, se présente, dans le cas où le vecteur f regroupe k fonctions objectif, de la façon suivante [Collette, 02] :

$$\begin{cases} \text{minimiser } \vec{f}(\vec{x}) & \vec{x} \in \Re^m, \ \vec{f}(\vec{x}) \in \Re^k \\ \text{avec } \vec{g}(\vec{x}) \leq 0 & \vec{g}(\vec{x}) \in \Re^m \\ \text{et } \vec{h}(\vec{x}) = 0 & \vec{h}(\vec{x}) \in \Re^p \end{cases}$$

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

Parmi les différentes méthodes utilisées, la méthode de pondération des fonctions objectif qui se présente comme suit : A chacune des fonctions objectif, est appliqué un coefficient de pondération et la somme pondérée des fonctions objectifs est effectuée. Une nouvelle fonction objectif est ainsi obtenue [Collette, 02].

Formulation du problème initial

$$\begin{cases} \text{minimiser } \vec{f}(\vec{x}),\ \vec{f}(\vec{x}) \in \Re^k,\ \vec{x} \in \Re^n \\ \text{avec } \vec{g}(\vec{x}) \le 0 \qquad \vec{g}(\vec{x}) \in \Re^m \\ \text{et } \vec{h}(\vec{x}) = 0 \qquad \vec{h}(\vec{x}) \in \Re^p \end{cases}$$

Formulation du problème avec la méthode de pondération des fonctions objectif

$$\Rightarrow \begin{cases} \text{minimiser } f_{eq}(\vec{x}) = \sum_{i=1}^{k} w_i f_i(\vec{x}) \qquad \vec{x} \in \Re^n \\ w_i \ge 0 \ \ \forall i \in \{1,...,k\} \text{ et } \sum_{i=1}^{k} w_i = 1 \\ \text{avec } \vec{g}(\vec{x}) \le 0 \qquad \vec{g}(\vec{x}) \in \Re^m \\ \text{et } \vec{h}(\vec{x}) = 0 \qquad \vec{h}(\vec{x}) \in \Re^p \end{cases}$$

L'avantage de cette approche, qualifiée de naïve, de l'optimisation multi-objectifs, est sa simplicité. Elle est également très efficace algorithmiquement compte tenu du fait qu'elle permet de retrouver la surface de compromis, en faisant varier les coefficients de pondération.

II.5 - Résolution d'un problème d'ordonnancement en industries pharmaceutiques par les algorithmes génétiques

Dans cette partie, une étude d'un problème d'ordonnancement multi-objectifs d'un atelier de type flow-shop en industries pharmaceutiques est proposé. Le problème traîté est présenté et résolu par les algorithmes génétiques.

II.5.1 - Présentation du problème

Avant sa mise sur le marché, un produit pharmaceutique passe par plusieurs ateliers : le magasin, l'atelier des formes solides et l'atelier de conditionnement dans lequel s'effectue la mise en boîtes des comprimés et gélules.

L'étape de conditionnement, consistant à mettre le produit fini en boîtes est une étape importante, mais qui entraîne des retards pouvant engendrer des coûts élevés.

En effet, des temps d'arrêts dus à des pannes machines ou à des arrêts obligatoires peuvent survenir lors de l'opération de conditionnement, sans compter les temps de nettoyage et de changement de format, devant être effectués sur les lignes entre le conditionnement de deux produits ainsi que les coûts engendrés par la non utilisation de ces lignes.

L'atelier de conditionnement étudié possède deux lignes de conditionnement L_1 et L_2 fonctionnant en flow-shop puisque chaque produit conditionné passe par l'ensemble des machines constituant chaque ligne dans un ordre unique.

L'optimisation du passage des produits dans les lignes de conditionnement consiste à minimiser les temps de production, et donc les coûts qui en découlent.

Nous nous proposons de mettre en œuvre une méthode d'optimisation multi-objectifs basée sur les algorithmes génétiques pour la résolution du problème d'ordonnancement des produits pharmaceutiques sur les deux lignes de conditionnement citées ci dessus.

II.5.2 - Formulation du problème

Il s'agit, dans un premier temps, d'ordonnancer seize produits p_i, $i = 1,...,16$, sur deux lignes de conditionnement L_j, $j = 1,2$. Par la suite, un problème plus complexe traite l'ordonnancement de 30 produits sur les deux lignes de conditionnement.

Il est à noter qu'une ligne de conditionnement peut fabriquer plusieurs produits. Un produit ne peut être fabriqué que sur une seule ligne et par respect d'un temps de nettoyage entre le conditionnement de deux produits. Des temps d'arrêts peuvent également être constatés durant le conditionnement d'un produit. Pour cela, il faut trouver le meilleur ordonnancement de ces produits sur les deux lignes pour minimiser les coûts de production en minimisant les temps d'arrêt, de nettoyage et de non utilisation des lignes.

a - Notations

O_{ik} : opération de conditionnement du produit i sur la ligne L_k

P_i : produit fini après l'opération O_{ik}

p_{ik} : temps de conditionnement de l'opération O_{ik}

$C_{P_{ik}}$: temps de fin d'exécution de P_i sur la ligne L_k

$C_{P_i}^{stk}$: coût de stockage par unité de temps du produit P_i

tp_{ik} : temps de préparation de la ligne L_k avant l'opération O_{ik}

t_{ik}^{arr} : temps d'arrêt durant l'opération O_{ik} sur la ligne L_k

t_{ik}^{nu} : temps de non utilisation de la ligne L_k avant l'opération O_{ik}

C_{prod}^{tot} : coût total de production

C_k^{ui} : coût unitaire de production du produit *i* sur la ligne L_k

DO_{ik}^{nett} : durée des opérations de nettoyage sur la ligne L_k

DO_{ik}^{chf} : durée des changements de format sur la ligne L_k

$C_{arr\,k}$: coûts d'arrêt et de non utilisation de la ligne L_k par unité de temps

C_{arr}^{tot} : coût total d'arrêt et de non utilisation des lignes par unité de temps

b - Critères à minimiser

Dans ce problème, les critères à minimiser sont ceux relatifs aux coûts de fabrication et aux coûts des temps engendrés par les opérations de nettoyage ainsi que des temps d'arrêt et de non utilisation des lignes de conditionnement.

Ils s'expriment respectivement par les deux fonctions objectifs suivantes : F_1 *et* F_2, correspondant respectivement au coût total de production et au coût d'arrêt et de non utilisation des lignes de conditionnement.

$$F_1 = C_{prod}^{tot} = \sum_k \sum_i C_k^{ui} w_{ik} p_{ik} \qquad \text{(II.1)}$$

$$F_2 = C_{arr}^{tot} = \sum_k C_{arr_k} \sum_i w_{ik} (tp_{ik}^{arr} + t_{ik}^{nu}) \qquad \text{(II.2)}$$

avec :

$$tp_{ik}^{arr} = DO_{ik}^{nett} + DO_{ik}^{chf}$$

$$w_{ik} = \begin{cases} 1 : \text{si le produit est fabriqué sur la ligne } L_k \\ 0 : \text{sinon} \end{cases}$$

c - Fonction fitness à optimiser

Etant donné que la minimisation du coût de production est plus importante que la minimisation des coûts de non utilisation des lignes de conditionnement, la fonction fitness

F à minimiser correspond à la somme pondérée des deux fonctions objectifs F_1 *et* F_2, avec des poids α_1 *et* α_2 définis en fonction de l'importance de ces deux critères :

$$F = \alpha_1 F_1 + \alpha_2 F_2 \quad \text{(II.3)}$$

$$F = \alpha_1 C_{prod}^{tot} + \alpha_2 C_{arr}^{tot} \quad \text{(II.4)}$$

$$F = \alpha_1 \left(\sum_k C_k^u \sum_i w_{ik} p_{ik} \right) + \alpha_2 \left(\sum_k C_{arr_k} \sum_i w_{ik} (tp_{ik}^{arr} + t_{ik}^{nu}) \right) \quad \text{(II.5)}$$

avec :

$\alpha_i \succ 0$, $i = 1, 2,$ et $\alpha_1 + \alpha_2 = 1$

où les α_i sont les coefficients qui, peuvent, de plus, par un choix adéquat de leurs valeurs, privilégier un critère par rapport à un autre.

II.5.3 - Algorithmes génétiques

Les algorithmes génétiques font partie des algorithmes évolutionnistes qui doivent leur nom à l'analogie avec les mécanismes d'évolution des espèces vivantes. Un algorithme évolutionniste typique est composé de trois éléments essentiels : une population constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné, un mécanisme d'évaluation de l'adaptation de chaque individu de la population à l'égard de son environnement extérieur et un mécanisme d'évolution composé d'opérateurs permettant d'éliminer certains individus et de produire de nouveaux individus à partir des individus sélectionnés [Hao, 99].

a - Présentation des algorithmes génétiques

Depuis l'idée originale proposée par Holland, le développement des algorithmes génétiques dans les trois dernières décennies a connu un essor considérable [Davis, 91], [Goldberg, 94], [Ross, 97], [Knosala et al, 01], [Iyer et al, 04], [Watanabe et al, 05], [Chang et al, 07], [Ruiz et al, 07].

Les algorithmes génétiques sont des techniques de recherche basées sur les mécanismes de la sélection naturelle et de la génétique. Ils se sont avérés très efficaces dans la résolution des problèmes complexes d'optimisation, tels que le problème du voyageur de commerce [Jog, 89], [Starkweather, 91] ou les problèmes d'ordonnancement [Ischibuchi, 94], [Kacem, 03],

[Mesghouni, 99], là ou les méthodes classiques, telles que l'algorithme branch and bound, ont montré leurs limites.

b - Fonctionnement d'un algorithme génétique

Dans la quête de la robustesse et de la solution optimale, ce qui distingue les algorithmes génétiques des méthodes classiques, peut être formulé selon quatre axes principaux :

- les algorithmes génétiques utilisent un codage des paramètres et non les paramètres eux-mêmes,
- ils travaillent sur une population de points et non sur un point unique,
- ils n'utilisent que les valeurs de la fonction étudiée, pas de sa dérivée ou d'une autre connaissance auxiliaire,
- ils utilisent des règles de transition probabilistes et non déterministes [Goldberg, 94].

Ainsi, la première étape dans l'application d'un algorithme génétique à un problème particulier consiste à convertir les solutions réalisables de ce problème dans une structure, appelée chromosome.

Afin de trouver la solution optimale d'un problème donné, un algorithme génétique standard commence par la production aléatoire d'un ensemble de solutions (chromosomes), appelé population initiale, et évolue vers des solutions différentes, mais meilleures, au fil des générations. Dans chaque génération, la fonction objectif, ou fitness, détermine les chromosomes à choisir pour la reproduction.

L'algorithme choisit, ainsi, les meilleurs chromosomes (ayant les meilleures fonctions fitness). Les opérateurs génétiques, tels que le croisement et la mutation, sont appliqués à ces chromosomes. Les nouveaux chromosomes (descendants) ainsi produits, constituent la génération suivante. Ces itérations continuent jusqu' à ce que le critère d'arrêt soit satisfait.

c - Codage des algorithmes génétiques

Le codage des solutions du problème d'optimisation, obtenues en utilisant les algorithmes génétiques, consiste à modéliser chaque solution par un chromosome, c'est-à-dire par une séquence de gènes. Initialement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace d'état.

Ce type de codage a pour intérêt de permettre de créer des opérateurs de croisement et de mutation simples. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus [Durand, 94]. Plus généralement, les gènes peuvent être des entiers, des réels, des caractères ou toute autre extension ou ensemble de ces entités élémentaires [Kacem, 03].

Le choix adéquat du codage, ou de la description de la solution, est une tâche importante pouvant garantir la réussite de l'application des algorithmes génétiques [Gargouri, 03].

d - Opérateurs des algorithmes génétiques

Deux opérateurs permettant la diversification de la population et son évolution sont utilisés par les algorithmes génétiques : l'opérateur de croisement et l'opérateur de mutation.

L'opérateur de croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants, comme le montre la figure 2.4. Cet opérateur assure donc le brassage et la recombinaison des gènes parentaux, permettant, ainsi, de former des descendants aux potentialités nouvelles.

Etant aléatoire, cet opérateur agit selon une probabilité p fixée par l'utilisateur, en fonction du problème à optimiser [Mesghouni, 99].

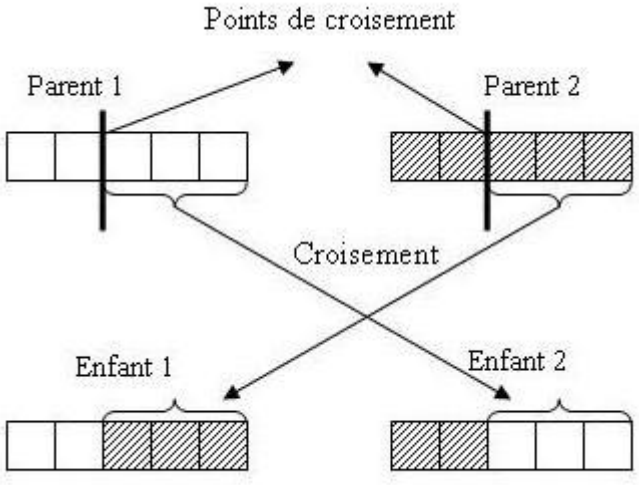

Figure 2. 4 - Fonctionnement de l'opérateur de croisement

L'opérateur de mutation consiste à changer, aléatoirement, la valeur de certains gènes dans un chromosome, comme le montre la figure 2.5. Sans elle, la population reste uniforme et risque de devenir incapable d'évoluer.

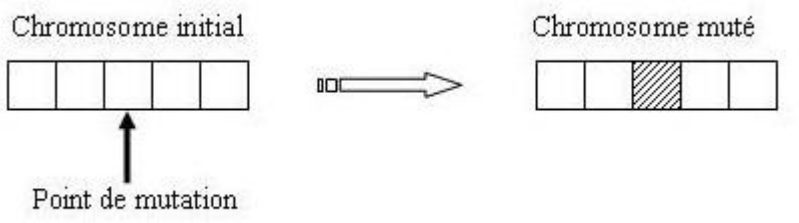

Figure 2. 5 - Fonctionnement de l'opérateur de mutation

Un codage et des opérateurs de croisement et de mutation sont proposés et utilisés, dans ce qui suit, pour la mise en œuvre et la résolution du problème d'ordonnancement flow-shop en industries pharmaceutiques.

II.5.4 - Codage CLOS proposé

Le codage des éléments de la population initiale est une étape importante dans la mise en œuvre des algorithmes génétiques. Le codage pris en compte dans le problème d'ordonnancement en industries pharmaceutiques, considéré dans cette étude, est le Codage en Liste des Opérations Structuré (CLOS) [Boukef et al, 06], [Boukef et al, 07].

Dans le cas de plusieurs machines, ce codage consiste à proposer, pour chaque individu, une structure de données contenant le numéro de la machine sur laquelle le produit est traité ainsi que ses dates de début et de fin de fabrication (tableau 2.1).

La formulation de ce codage utilise les notations suivantes :

nL_i : le numéro de la ligne de conditionnement sur laquelle est fabriqué le produit P_i,

t_i : le temps de début de fabrication du produit P_i,

C_i : le temps de fin de fabrication du produit P_i,

F : la fonction fitness à optimiser.

Tableau 2. 1 - Codage CLOS pour n lignes et m produits pour un individu *i* donné

$Produit_1$	nL_1	t_1	C_1	F
$Produit_2$	nL_2	t_2	C_2	
…				
$Produit_j$	nL_j	t_j	Cj	
…				
$Produit_m$	nL_n	t_n	C_n	

II.5.5 - Opérateurs proposés

Les opérateurs utilisés par les algorithmes génétiques permettent de diversifier la population au cours des générations et d'explorer l'espace d'état, représenté par l'espace des solutions. L'opérateur de croisement recompose les gènes d'individus existant dans la population, quant à l'opérateur de mutation, il a pour but de garantir l'exploration de l'espace d'état.

Ces différents opérateurs proposés pour la résolution du problème traité, sont présentés dans ce qui suit.

a - Opérateur de sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer partiellement les mauvais. Néanmoins, ce n'est pas parce qu'un individu est bon qu'il survit nécessairement et ce n'est pas parce qu'il est mauvais qu'il disparaît.

En effet, bien souvent, une espèce « bien adaptée » peut descendre d'un individu décrété « mauvais ». Il existe différents principes de sélection, dont le principe de la roulette pondérée [Goldberg, 94]. Il s'agit d'une roulette sur laquelle sont placés tous les chromosomes de la population, la place accordée à chacun des chromosomes étant en relation avec sa valeur d'adaptation (sa fonction fitness). Ensuite, une bille est lancée et s'arrête sur un chromosome. Les meilleurs chromosomes peuvent ainsi être tirés plusieurs fois et les plus mauvais ne jamais être sélectionnés.

Le principe de la roulette pondérée est expliqué dans les figures 2.6, 2.7 et 2.8 suivantes.

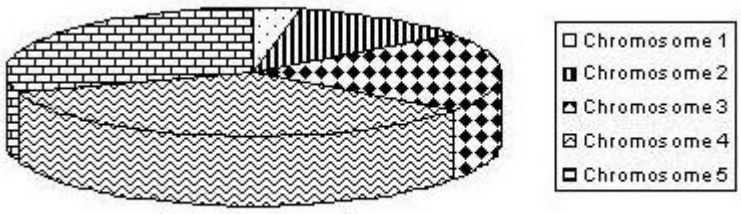

Figure 2. 6 - Placement des chromosomes sur la roulette la roulette

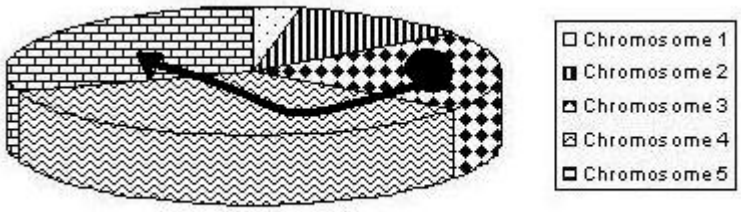

Figure 2. 7 - Lancement d'une bille sur la roulette

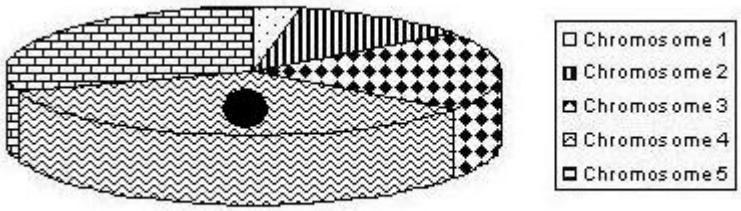

Figure 2. 8 - Arrêt de la bille sur un chromosome, ici, sur celui ayant la meilleure fitness

b - Opérateur de croisement

Le croisement a pour but d'enrichir et de diversifier la population en manipulant la structure des chromosomes.

- L'opérateur de croisement à un point proposé consiste à choisir deux parents et un point de croisement. Le parent 1 (respectivement le parent 2) reçoit les chromosomes du parent 2 (respectivement du parent 1) qui suivent le point de croisement, comme le montre la figure 2.9. Une mise à jour est ensuite effectuée pour le calcul de la fonction fitness.

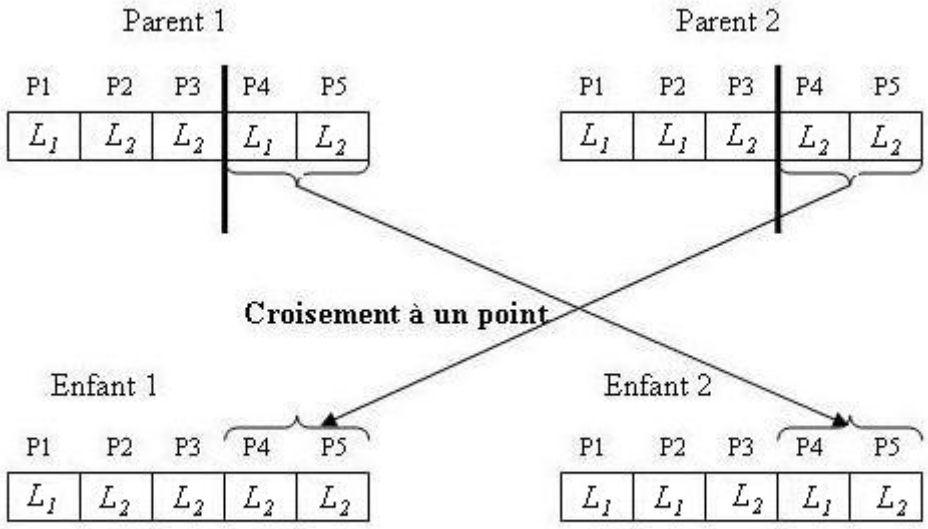

Figure 2. 9 - Fonctionnement de l'opérateur de croisement à un point

- L'opérateur de croisement à deux points proposé, consiste à choisir deux parents et deux points de croisement. Le parent 1 (respectivement le parent 2) reçoit les chromosomes du parent 2 (respectivement du parent 1) qui précèdent le premier point de croisement et qui suivent le deuxième, comme le montre la figure 2.10. Une mise à jour est ensuite effectuée pour le calcul de la fonction fitness.

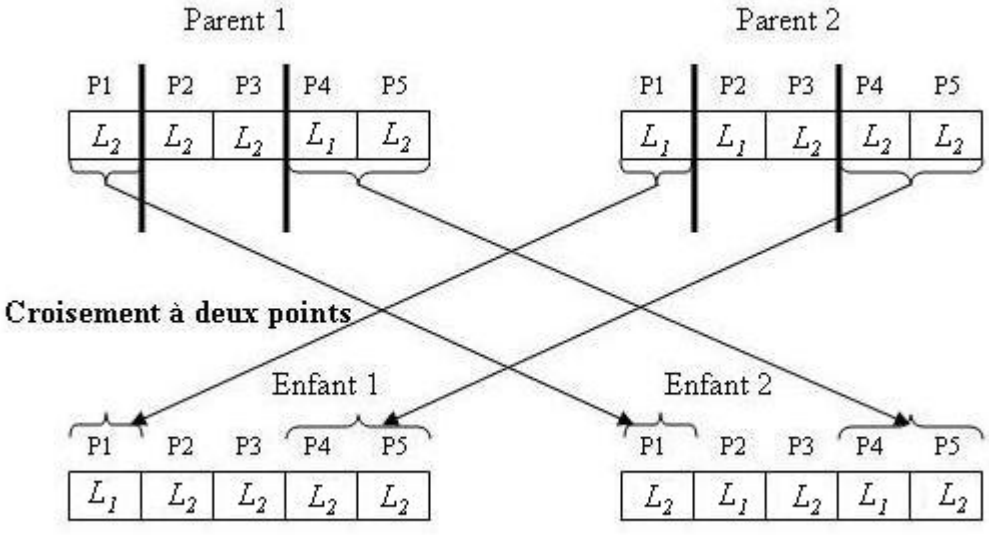

Figure 2. 10 - Fonctionnement de l'opérateur de croisement à deux points

c - Opérateur de mutation

Les propriétés de convergence des algorithmes génétiques sont fortement dépendantes de l'opérateur de mutation. En effet, cet opérateur permet à l'algorithme génétique d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous, dans le processus de résolution [Durand, 94].

- L'opérateur de mutation à un point proposé, consiste à choisir un produit fabriqué par une ligne de conditionnement et le transférer sur l'autre ligne, comme le montre la figure 2.11. Une mise à jour est ensuite effectuée conduisant à une modification de la fonction fitness.

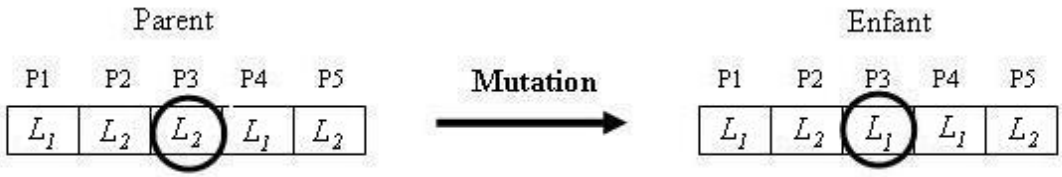

Figure 2. 11 - Fonctionnement de l'opérateur de mutation à un point

- L'opérateur de mutation à deux points proposé, consiste à choisir deux produits fabriqués sur deux lignes de conditionnement différentes et à inverser les numéros des lignes, tel que le montre la figure 2.12.

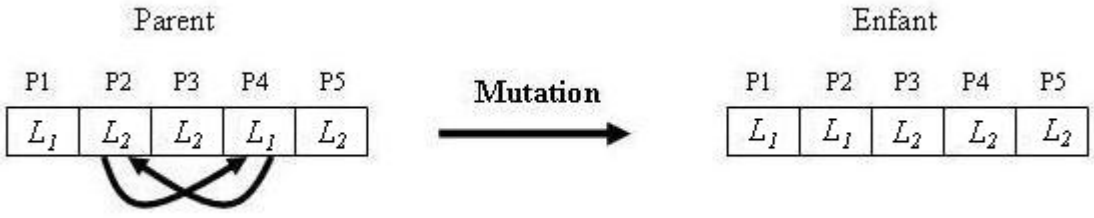

Figure 2. 12 - Fonctionnement de l'opérateur de mutation à deux points proposé

II.5.6 - Algorithme proposé

Les étapes décrites ci-après, expliquent en détail le fonctionnement des différents opérateurs de l'algorithme génétique appliqué aux problèmes d'ordonnancement et plus particulièrement à ceux des industries pharmaceutiques.

Début
Pour un nombre d'itérations choisis
    Créer (population initiale)
    Choisir une probabilité p
    Pour *i* = 0 ***à*** nbre_individus – 1 Sélectionner individu1
        Si individu1(i).prob < p ***Alors***
            Appliquer opérateur de mutation à individu1
        S***inon***
            Pour *j =0* ***à*** nbre_individus - 1
                Selectionner individu2 ***tq*** individu1 ≠ individu2
                Si individu2(j).prob < p ***Alors***
                    Appliquer opérateur de mutation à individu1
                Sinon
                    Appliquer les opérateurs de croisement à un point et à deux points aux individus 1 et 2
                Fin Si
            Fin Pour
        Fin Si
    Mettre à jour la population d'individus
    Fin Pour
Trouver la meilleure fonction fitness
Enregistrer le meilleur individu
Fin Pour
Fin

La démarche de cet algorithme est proposée dans la figure 2.13 ci-dessous.

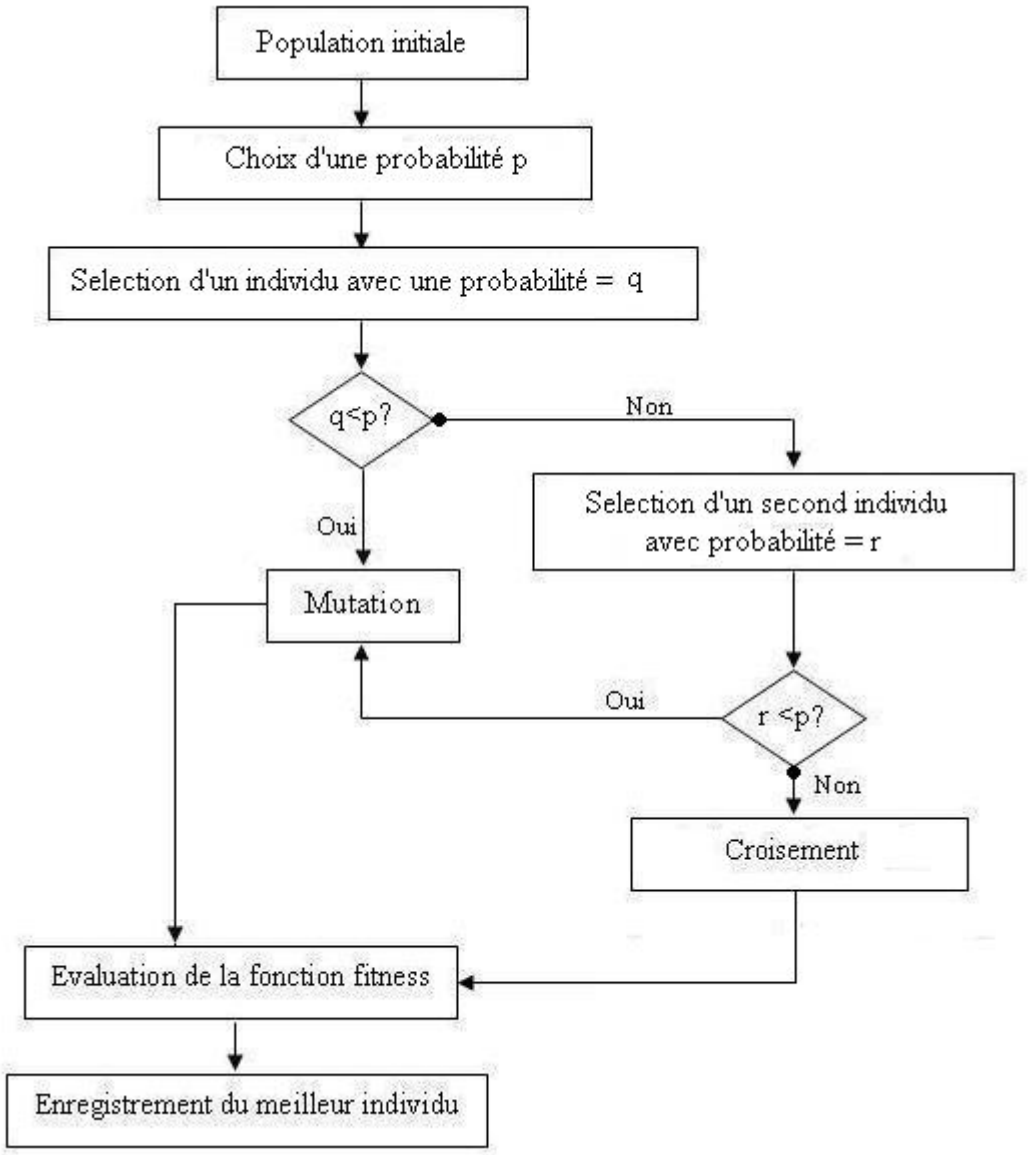

Figure 2. 13 - Etapes de mise en œuvre de l'algorithme génétique proposé

II.6 - Simulation et résultats

Deux exemples, traitant de problèmes réels rencontrés dans une industrie pharmaceutique comportant deux lignes de conditionnement (L1 et L2) sont présentés dans cette partie.

Ces problèmes consistent à ordonnancer respectivement 16 et 30 produits sur les deux lignes L1 et L2 pour montrer le comportement des algorithmes génétiques face à deux problèmes de tailles et de complexités différentes.

Les données de ces problèmes d'ordonnancement de type flow-shop en industries pharmaceutiques sont présentées dans les tableaux 2.2 et 2.3.

Elles sont relatives aux temps de nettoyage (*tn_Li)*, temps d'arrêt (*ta_Li)*, temps de production (*tp_Li),* coûts de production (*Cp_Li)* et aux coûts de non utilisation (*Cnu_Li)* des produits *(Nom_pr)* de chacune des deux lignes de conditionnement *L1* et *L2*.

II.6.1 - Exemple de 16 produits traités sur 2 lignes de conditionnement

Les temps de production (*tp_Li*), d'arrêt (*ta_Li*) et de nettoyage (*tn_Li*), indiqués dans le tableau 2.2, sont en heures et les différentes dates sont calculées à partir d'un temps initial t_0.
Prenant en compte les données présentées, les coûts de production (*Cp_Li*) et de non utilisation des lignes (*Cnu_Li*) sont calculés en fonction des formules II.1 et II.2 et la fonction fitness déduite à partir de la formule II.5 du paragraphe II.5.2.

Tableau 2. 2 - Données relatives à un problème d'ordonnancement 16x2 en industries pharmaceutiques

Nom_pr	*tn_Li*	*ta_Li*	*tp_Li*	*Cp_Li*	*Cnu_Li*
pr1	3/2,5	3/2	42/30	5/5,5	3,5/2
pr2	2,5/3	4/4	30/30	5/5,5	3,5/2
pr3	4/3	2/3	40/40	5/5,5	3,5/2
pr4	3/3	4/4	18/22	5/5,5	3,5/2
pr5	2/2,5	3/2	15/25	5/5,5	3,5/2
pr6	4/2,5	3/2	42/30	5/5,5	3,5/2
pr7	2,5/3	4/4	30/30	5/5,5	3,5/2
pr8	2/3	2/3	40/40	5/5,5	3,5/2
pr9	3,5/3	4/4	17/25	5/5,5	3,5/2
pr10	2/2,5	3/2	15/20	5/5,5	3,5/2
pr11	3/3	4/4	30/30	5/5,5	3,5/2
pr12	4/3	2/3	40/40	5/5,5	3,5/2
pr13	3,5/3	4/4	18/20	5/5,5	3,5/2
pr14	2/2,5	3/2	15/12	5/5,5	3,5/2
pr15	3/2,5	3/2	40/30	5/5,5	3,5/2
pr16	2,5/3	4/4	30/32	5/5,5	3,5/2

Les algorithmes génétiques, utilisant le codage CLOS sont appliqués en vue d'obtenir le meilleur ordre de passage des produits pour minimiser la fonction objectif *F*.
Ainsi, pour 500 générations et une population de 20 individus par génération, les opérateurs de croisement et de mutation sont appliqués, générant de nouveaux individus dont les nouveaux coûts sont calculés jusqu'à l'obtention du meilleur individu avec le coût le plus bas.

Le meilleur individu pour lequel le minimum global, 1260, est obtenu est présenté dans la figure 2.14. Les différents types de minima sont ainsi illustrés avec pour chacun la fonction fitness correspondante et une convergence à la 445ème génération.

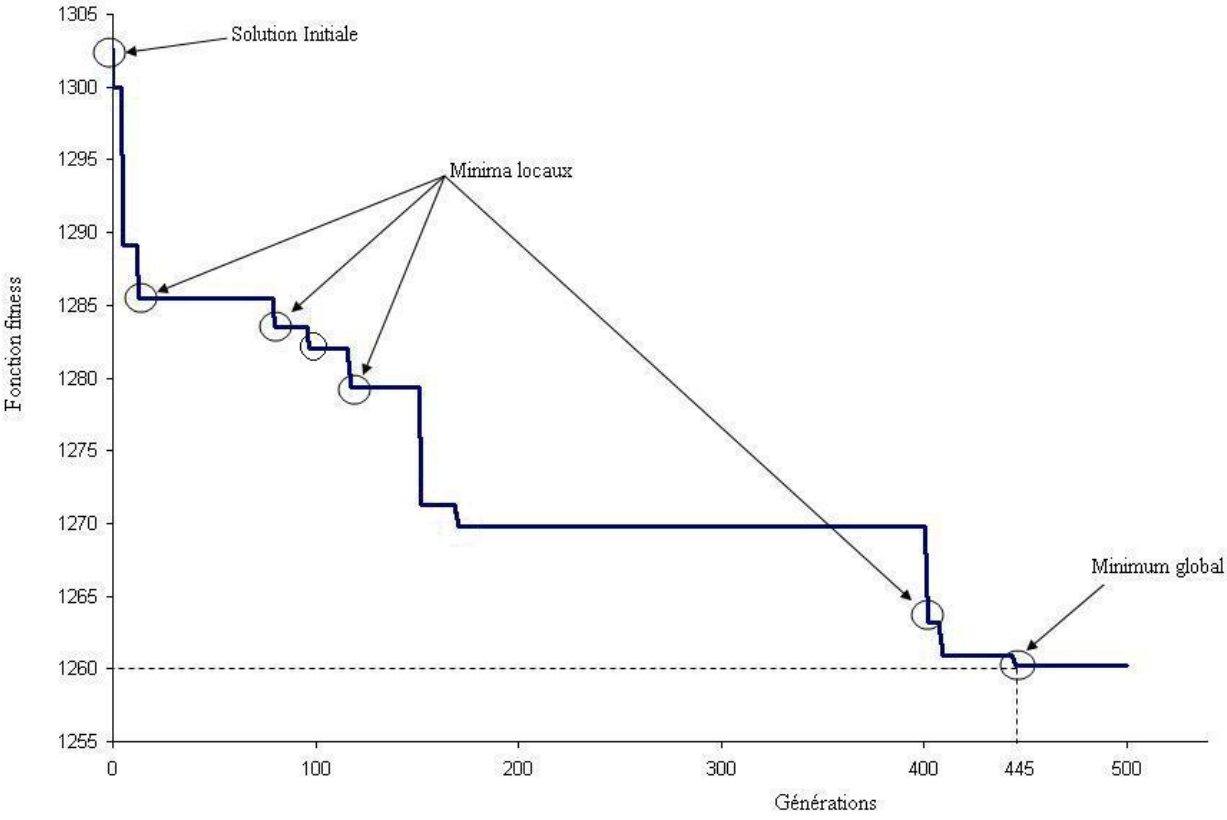

Figure 2. 14 - Evolution des coûts à travers les générations pour le problème d'ordonnancement 16x2 en industries pharmaceutiques

Le diagramme de Gantt présenté dans la figure suivante correspond au meilleur minimum obtenu après application des algorithmes génétiques sur le problème 16 x 2.

Nous obtenons, ainsi, une répartition assez équilibrée des seize produits sur les deux lignes de conditionnement avec un Cmax de 230.

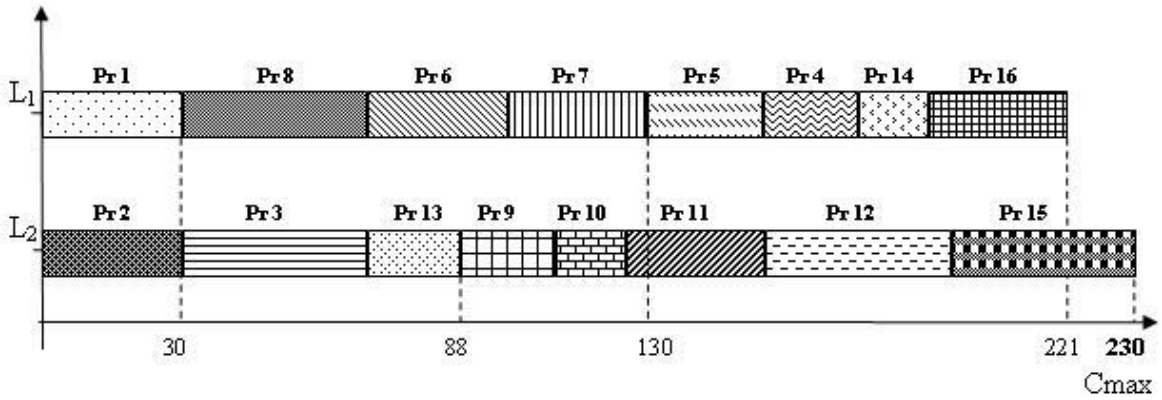

Figure 2. 15 – Diagramme de Gantt relatif au meilleur individu pour le problème 16 x 2 utilisant les algorithmes génétiques

II.6.2 - Exemple de 30 produits traités sur 2 lignes de conditionnement

Les temps de production (*tp_Li*), d'arrêt (*ta_Li*) et de nettoyage (*tn_Li*) indiqués dans le tableau 2.3 sont en heures et les différentes dates sont calculées à partir d'un temps initial t_0. Prenant en compte les données présentées, les coûts de production (*Cp_Li*) et de non utilisation des lignes (*Cnu_Li*) sont calculés en fonction des formules 1 et 2 et la fonction fitness déduite à partir de la relation II.5.

Tableau 2. 3 - Données relatives à un problème d'ordonnancement 30x2 en industries pharmaceutiques

Nom_pr	*tn_Li*	*ta_Li*	*tp_Li*	*Cp_Li*	*Cnu_Li*
pr1	3/2,5	3/2	42/30	5/5,5	3,5/2
pr2	2,5/3	4/4	30/30	5/5,5	3,5/2
pr3	4/3	2/3	40/40	5/5,5	3,5/2
pr4	3/3	4/4	18/22	5/5,5	3,5/2
pr5	2/2,5	3/2	15/25	5/5,5	3,5/2
pr6	4/2,5	3/2	42/30	5/5,5	3,5/2
pr7	2,5/3	4/4	30/30	5/5,5	3,5/2
pr8	2/3	2/3	40/40	5/5,5	3,5/2
pr9	3,5/3	4/4	17/25	5/5,5	3,5/2
pr10	2/2,5	3/2	15/20	5/5,5	3,5/2
pr11	3/3	4/4	30/30	5/5,5	3,5/2
pr12	4/3	2/3	40/40	5/5,5	3,5/2
pr13	3,5/3	4/4	18/20	5/5,5	3,5/2
pr14	2/2,5	3/2	15/12	5/5,5	3,5/2
pr15	3/2,5	3/2	40/30	5/5,5	3,5/2
pr16	2,5/3	4/4	30/32	5/5,5	3,5/2
pr17	3/2	3/4	60/65	5/5,5	3,5/2
pr18	2,5/4	3/2	35/30	5/5,5	3,5/2
pr19	4/4	3/3	40/40	5/5,5	3,5/2
pr20	3/4	4/3	45/50	5/5,5	3,5/2
pr21	4/3,5	3/4	30/32	5/5,5	3,5/2
pr22	3/3,5	2/3	44/37	5/5,5	3,5/2
pr23	4/2	4/3	37/40	5/5,5	3,5/2
pr24	3,5/3	3/4	45/45	5/5,5	3,5/2
pr25	3/3	3/2	13/12	5/5,5	3,5/2
pr26	2/2	4/4	27/25	5/5,5	3,5/2
pr27	2/2	4/3	30/33	5/5,5	3,5/2
pr28	2,5/3	2/3	10/13	5/5,5	3,5/2
pr29	4/3,5	2/2	31/30	5/5,5	3,5/2
pr30	3,5/3	2/2	29/25	5/5,5	3,5/2

Les algorithmes génétiques, utilisant le codage CLOS sont appliqués en vue d'obtenir le meilleur ordre de passage des produits pour minimiser la fonction objectif *F*.

Ainsi, pour 500 générations et une population de 20 individus par génération, les opérateurs de croisement et de mutation sont appliqués, générant de nouveaux individus dont les nouveaux coûts sont calculés jusqu'à l'obtention du meilleur individu avec le coût le plus bas.

Le meilleur individu pour lequel le minimum global est obtenu est présenté dans la figure 2.15. Les différents types de minima sont ainsi illustrés avec pour chacun la fonction fitness correspondante et une convergence à la 161ème génération.

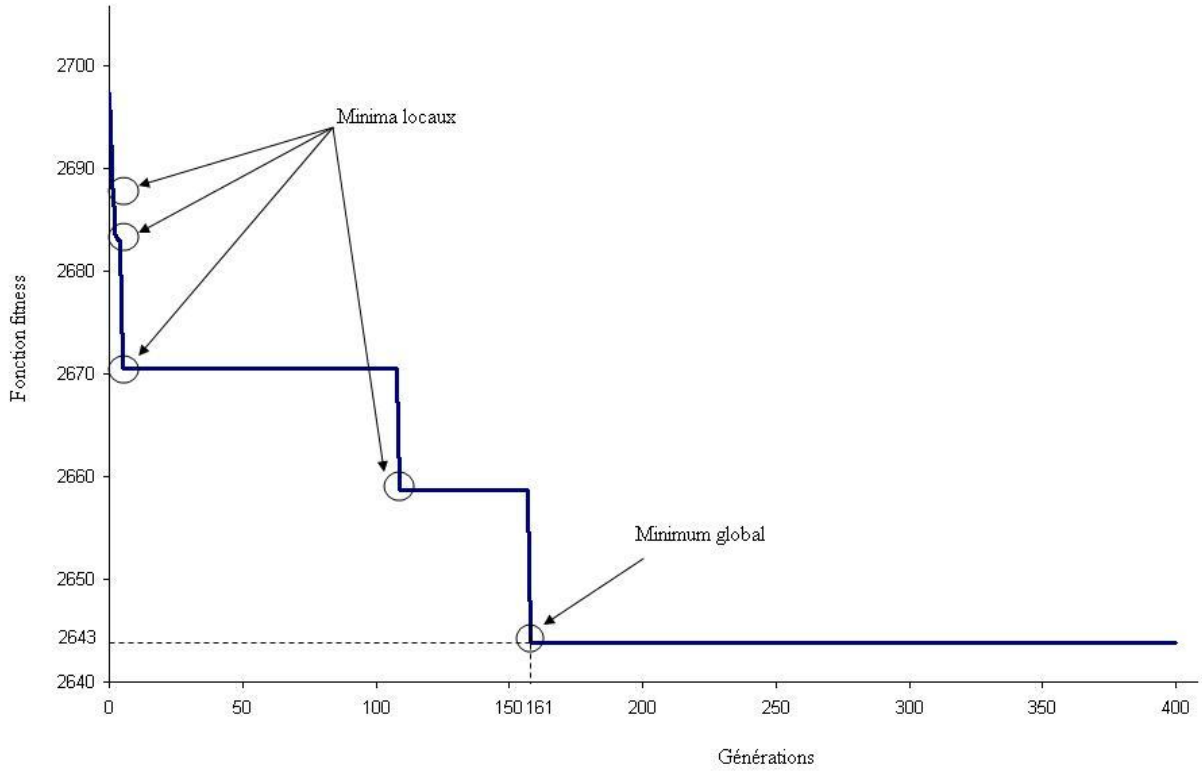

Figure 2. 16 - Evolution des coûts à travers les générations pour le problème d'ordonnancement 30x2 en industries pharmaceutiques

Le diagramme de Gantt présenté dans la figure suivante représente le meilleur minimum obtenu après application des algorithmes génétiques sur le problème 30 x 2.

Nous obtenons, ainsi, une répartition des trente produits sur les deux lignes de conditionnement avec un Cmax de 502.

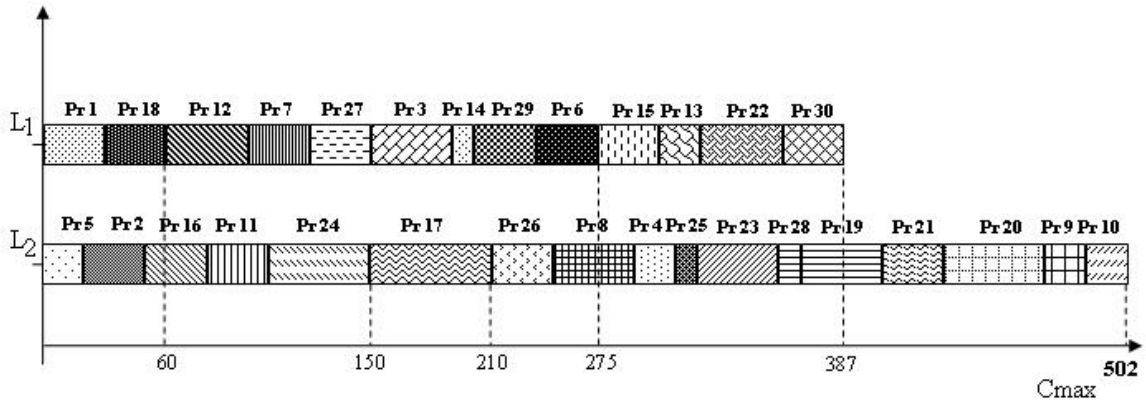

Figure 2. 17 – Diagramme de Gantt relatif au meilleur individu pour le problème 30 x 2 utilisant les algorithmes génétiques

L'utilisation des algorithmes génétiques pour la résolution de ces deux problèmes d'ordonnancement en industries pharmaceutiques a confirmé la capacité de cette méthode à atteindre un optimum global respectant les différents critères et contraintes établies.

Cette méthode est comparée avec la méthode d'optimisation par essaim particulaire dans le chapitre suivant nous permettant ainsi de conclure sur l'utilisation de l'une ou l'autre des deux méthodes dans le cas de problèmes d'optimisation dans le cas discret et plus particulièrement dans les cas des problèmes job-shop flexible et des problèmes flow-shop en industries pharmaceutiques

II.7 - Conclusion

La résolution des problèmes d'ordonnancement dans le cas des industries pharmaceutiques par les algorithmes génétiques dans le cas multi-objectifs, constitue la contribution principale de ce chapitre. Après avoir introduit les ateliers de type flow-shop ainsi que les algorithmes génétiques, une résolution de deux problèmes d'ordonnancement réels en industries pharmaceutiques est proposée. Les différents critères, leur combinaison formant la fonction fitness ainsi que les contraintes relatives au problème de l'ordonnancement, des ateliers de type flow-shop en industries pharmaceutiques, sont ensuite présentés et le codage CLOS ainsi que les différents opérateurs à utiliser et leur fonctionnement proposés.

La résolution de ces deux problèmes, ainsi présentés, a abouti à la conclusion que les algorithmes génétiques ont fait leurs preuves dans la recherche du minimum global et permettent d'atteindre une bonne solution souvent proche de l'optimum.

Chapitre III

Résolution de problèmes d'ordonnancement job-shop flexible par la méthode basée sur l'optimisation par essaim particulaire

III.1 - Introduction

Dans le contexte industriel actuel, l'entreprise doit veiller à garantir un système de production fiable pour ne pas s'écarter des objectifs qu'elle se fixe et des budgets qu'elle s'alloue. L'ordonnancement représente l'une des fonctions les plus importantes pour assurer la performance globale de ce système.

A l'heure actuelle, certains problèmes d'ordonnancement ne peuvent être résolus par des méthodes exactes, l'intérêt des industries se trouve donc focalisé sur le développement de méthodes approchées. Naturellement, ces méthodes doivent avoir des bases théoriques solides permettant d'approcher ces problèmes complexes avec une certaine confiance dans la qualité du résultat.

Parmi les problèmes d'ordonnancement les plus difficiles, on cite ceux relatifs aux ateliers de type job-shop classique. Leur résolution de manière optimale s'avère, dans la plupart des cas, très difficile à cause de leur caractère fortement combinatoire.

L'utilisation des méthodes exactes requiert un ensemble de calculs, dont le nombre évolue de façon exponentielle avec la taille du problème considéré. Il s'avère donc préférable d'utiliser les méthodes approchées, telles que celles basées sur le principe de la recherche locale ou sur les méthodes évolutives.

Les problèmes d'ordonnancement d'ateliers de type job-shop flexible constituent une extension des problèmes job-shop et sont, de ce fait, encore plus difficiles à résoudre [Garey et al, 79].

La méthode d'optimisation par essaim particulaire a été proposée, au départ, comme solution aux problèmes d'optimisation continus non linéaires tels que le contrôle des puissances [Abido, 02], l'apprentissage des réseaux de neurones [Van den Bergh et al, 00] et la recherche du chemin optimal [Onwubolu et al, 04]. Les domaines d'application de l'optimisation par essaim particulaire sur les problèmes d'optimisation combinatoire restent, néanmoins, assez limités [Lian et Jiao, 06], [Lian et al, 06 b], [Pan et al, 08],…

Dans ce troisième chapitre, après l'introduction des Problèmes d'ordonnancement de type Job-Shop Flexible (FJSP en anglais) ainsi que leur formulation, sont présentées la méthode d'optimisation par essaim particulaire et ses spécificités dans le cas continu.

L'utilisation de cette méthode dans le cas discret pour la résolution de certains problèmes de ce type est par la suite proposée.

Des résultats relatifs à l'application de cette méthode sont finalement présentés et une comparaison avec ceux obtenus ultérieurement par application des algorithmes génétiques, est considérée pour mettre en exergue les différences et les similitudes entre ces deux méthodes.

III.2 - Problèmes d'ordonnancement de type job-shop flexible (FJSP)

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

III.2.1 - Présentation des problèmes FJSP

Les problèmes d'ordonnancement d'ateliers de type job-shop flexible sont connus dans la littérature comme étant les problèmes les plus difficiles à résoudre.
La difficulté réside dans le choix de la meilleure métaheuristique pour leur résolution ainsi que pour la détermination des meilleurs ordonnancements en des temps raisonnables, les plus proches possibles de la solution optimale. Plusieurs auteurs se sont penchés sur la résolution des problèmes d'ordonnancement job-shop flexible.
Bruker et Schlie [Bruker et al, 90] ont développé un algorithme polynomial pour la résolution d'un problème FJSP à deux jobs. Yang [Yang, 01] a présenté un nouvel algorithme génétique basé sur l'approche de la programmation linéaire. Chambers [Chambers, 96] a développé un algorithme de recherche tabou pour ce type de problème. Wu and Weng [Wu et al, 05] ont proposé une méthode basée sur les systèmes multi-agents. Xia et Wu [Xia et al, 05] ont traité ce problème avec une approche hybride alliant l'algorithme d'optimisation par essaim particulaire et le recuit simulé. Gao, Gen, Sun et Zhao [Gao et al, 07] ont quand à eux, traité ce problème en utilisant un algorithme génétique hybride. Dans ce rapport, le problème FJSP est traité en totalité par l'algorithme d'optimisation par essaim particulaire discrétisé.

III.2.2 - Formulation des problèmes FJSP

Les problèmes d'ateliers job-shop flexibles peuvent être formulés comme suit [Saad et al, 07]:

- soit un ensemble de n produits indépendants à réaliser sur m machines M_k, $k = 1, 2, \ldots, m$,
- 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 ressource sélectionnée à partir d'un ensemble de machines disponibles,
- chaque machine ne peut réaliser qu'une seule opération à la fois,
- l'assignation d'une opération $O_{i,j}$ à une machine M_k entraîne l'occupation de cette machine durant tout le temps d'exécution de l'opération, noté $p_{i,j,k}$,
- la préemption n'est pas autorisée.

Le FJSP présente deux difficultés principales :

- la première est relative à l'assignation de chaque opération $O_{i,j}$ à une machine M_k,
- la seconde correspond au calcul des temps de début $t_{i,j}$ et des temps de fin $tf_{i,j}$ de l'opération $O_{i,j}$.

III.3 - Optimisation par essaim particulaire (OEP)

La méthode d'optimisation par essaim particulaire est semblable à la méthode des algorithmes génétiques dans le sens où sa population initiale est générée d'une manière aléatoire, et différente de celle-ci par le fait qu'elle n'utilise pas les opérateurs de croisement et de mutation à la base de la mise en œuvre des algorithmes génétiques.

III.3.1 - Présentation de la méthode OEP

La méthode d'optimisation par essaim particulaire fait partie des méthodes de recherche globale dont le système est initialisé avec une population, nommée « essaim », composée de solutions aléatoirement générées, et pour laquelle la recherche de la solution optimale se fait par une mise à jour de cette population.
Chaque individu ou solution potentielle, appelé également particule, possède une vitesse ajustée dynamiquement, relativement à l'expérience propre de la particule ainsi qu'à celle des particules voisines.
L'OEP a été initialement introduite pour traiter des problèmes d'optimisation dans le domaine continu alors que beaucoup de problèmes d'optimisation se situent dans l'espace discret [Lian et al, 06 a], [Lian et al, 06 b]. Il s'avère donc intéressant de recourir à des modifications sur l'algorithme initial pour pouvoir répondre aux besoins spécifiques aux problèmes considérés pour différents types d'ateliers [Lian et al, 06 c], [Sha et al, 06], [11], [Xia et al, 05].

III.3.2 - Optimisation par essaim particulaire dans le cas continu

Les principes de base de l'OEP classique sont très simples. Un essaim, qui contient un ensemble de particules, entre au commencement dans un espace de recherche. Chaque particule de l'essaim possède les cinq caractéristiques suivantes:

- sa position,
- sa vitesse,

- sa position actuelle et la valeur de la fonction objectif pour cette position,
- la valeur de la meilleure position de ses voisins et la fonction objectif correspondante,
- sa meilleure position précédente.

Les notations relatives à ces caractéristiques sont les suivantes:

$V_i(t)$: vitesse de la particule i à l'itération t,

$X_i(t)$: position courante de la particule i à l'itération t,

$P_i(t)$: meilleure ancienne position de la particule i à l'itération t,

$G_i(t)$: meilleure position du voisinage de la particule i à l'itération t.

A un moment donné, une particule doit effectuer un choix entre :

- garder sa position actuelle et continuer selon sa vitesse actuelle,
- prendre en compte son ancienne position,
- prendre en compte la meilleure position de son voisinage [Clerc et al, 09].

Ceci peut être formulé analytiquement par les relations suivantes:

$$V_i(t+1) = c_1 V_i(t) + c_2\left(P_i(t) - X_i(t)\right) + c_3\left(G_i(t) - X_i(t)\right) \quad \text{(III.1)}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad \text{(III.2)}$$

c_1, c_2, c_3 étant des coefficients de confiance.

III.3.3 - Optimisation par essaim particulaire dans le cas discret

L'algorithme d'optimisation par essaim particulaire étant défini initialement dans le domaine continu, il s'avère évidemment nécessaire d'effectuer une conversion du domaine continu vers le domaine discret pour la résolution des problèmes d'ordonnancement. La première étape de cette conversion consiste à créer une structure de particules puis à proposer un algorithme présentant la marche à suivre pour une meilleure résolution des problèmes FJSP.

a - Formulation générale des problèmes d'ordonnancement FJSP par essaim particulaire

Pour passer du cas continu vers le cas discret, considérons les nouvelles notations suivantes :

O_{ij}^f : opération i d'un job j pour une particule f,

$P^f(t)$: position actuelle de la particule f à une itération t,

$m_{ij}^f(t)$: machine sélectionnée pour l'exécution de l'opération O_{ij}^f à une itération t pour la position actuelle de la particule f,

$P_B^f(t)$: meilleure position du voisinage de la particule f à une itération t,

$P_b^f(t)$: meilleure position connue de la particule f à une itération t,

$[m_{ij}^f(t)]_b$: machine sélectionnée pour l'exécution de l'opération O_{ij}^f pour la meilleure position connue de la particule f à une itération t,

$[m_{ij}^f(t)]_B$: machine sélectionnée pour l'exécution de l'opération O_{ij}^f pour la meilleure position du voisinage de la particule f à une itération t,

Δm_{ij}^f : modification à appliquer à la particule f pour l'affectation des machines,

μm_{ij}^f : vecteur de mutation à appliquer à la particule f pour permettre le changement de position,

α, β, γ : coefficients de confiance.

Pour chaque individu, les changements à effectuer pour passer d'une position $P^f(t)$ à une autre $P^f(t+1)$ doivent respecter la relation III.3 suivante :

$$P^f(t+1) = P^f(t) + \Delta m_{ij}^f \quad \text{(III.3)}$$

avec:

$$\Delta m_{ij}^f = f(\mu m_{ij}^f, m_{ij}^f) \quad \text{(III.4)}$$

et :

$$\mu m_{ij}^f = \alpha[m_{ij}^f(t)] + \beta[(m_{ij}^f(t))_b - m_{ij}^f(t)] + \gamma[(m_{ij}^f(t))_B) - m_{ij}^f(t)] \quad \text{(III.5)}$$

b - Présentation de la structure d'une particule

Présenter une structure de particule revient à présenter un exemple d'affectation d'un ensemble d'opérations O_{ij} considérées, à un ensemble de machines M_k en indiquant les temps de début t_{ij} correspondants [Boukef et al, 08].

Tableau 3. 1- Exemple de structure d'une particule

Opération	Numéro de la machine	Temps de début d'exécution de l'opération
O_{11}	M_1	t_{11}
O_{21}	M_3	t_{21}
O_{31}	M_3	t_{31}
O_{12}	M_3	t_{12}
O_{22}	M_2	t_{22}
O_{32}	M_1	t_{32}
O_{13}	M_2	t_{13}
O_{23}	M_1	t_{23}

III.3.4 - Algorithme Basé sur la méthode d'Optimisation par Essaim Particulaire pour le cas discret (BOEP)

a - Etapes de l'algorithme BOEP proposé

Une population initiale, appelée essaim, est générée aléatoirement dans une première étape, en affectant chaque opération à une machine, tout en respectant un ordre de priorité. En deuxième étape, une particule est sélectionnée parmi l'essaim et un voisinage contenant la particule désignée est choisi. Des heuristiques sont ensuite utilisées pour améliorer certaines affectations de machines aux opérations. Des changements à effectuer sur la position de la particule pour la rapprocher de la meilleure solution de son voisinage ou de sa propre meilleure performance ultérieure sont, par la suite, déterminés par l'élaboration d'un vecteur appelé « vecteur mutation » qui permet, en effet, de modifier les affectations des machines aux opérations en vue d'améliorer la fonction à optimiser.

Ces étapes relatives à l'évolution de l'algorithme d'optimisation par essaim particulaire dans le cas discret, représentées dans la figure 3.1, sont reprises jusqu'à ce qu'un certain nombre d'itérations soit atteint [Boukef et al, 09].

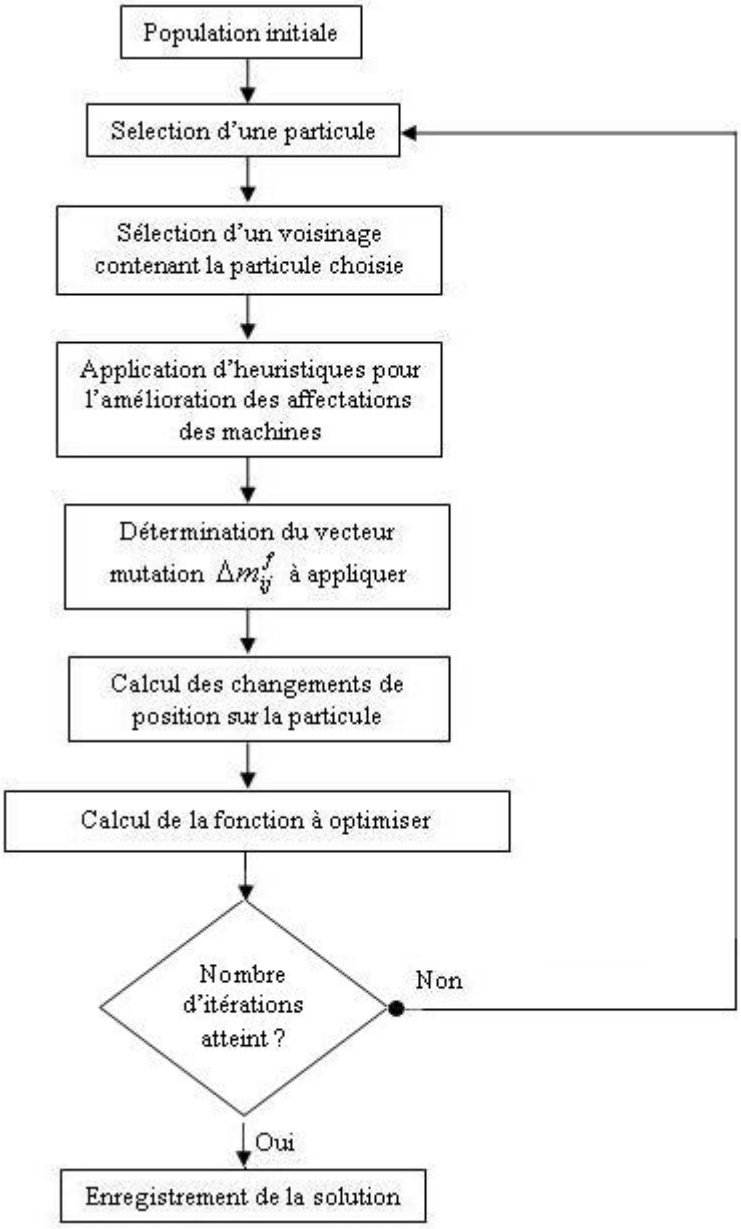

Figure 3. 1- Etapes relatives à l'évolution de l'algorithme BOEP

b - Algorithme BOEP proposé

Les différentes étapes schématisées ci-dessus dans la figure 3.1 sont expliquées en détail dans l'algorithme BOEP suivant. Pour chaque particule (individu) de l'essaim, ces étapes sont suivies et la particule présentant la meilleure fonction objectif est considérée comme la solution optimale obtenue.

$\boldsymbol{P}$: particule choisie,

$\boldsymbol{P_m}$: meilleure particule connue,

$\boldsymbol{P_M}$: meilleure particule du voisinage.

Début

Pour un nombre d'itérations choisi

Créer (population initiale)

Pour *i* = 0 ***à*** n – 1 // n correspondant au nombre de particules dans l'essaim

Sélectionner *une particule* ***P***

Sélectionner *un voisinage* contenant ***P***

// pour une population de 50 particules, un voisinage de 5 particules a été choisi

Appliquer **heuristique1** et **heuristique 2** pour améliorer l'affectation des machines*****

Pour *j* = 1 à nbre_machines

Vitesse_locale(j) = $\boldsymbol{P_b}$*.machine(j) -* ***P****.machine(j)*

//La vitesse locale correspondant à l'expression $(m_{ij}^f(t))_b - m_{ij}^f(t)$ dans la formule III.5.

Vitesse_globale(j) = $\boldsymbol{P_B}$*.machine(j) -* ***P****.machine(j)*

//La vitesse globale correspondant à l'expression $(m_{ij}^f(t))_B) - m_{ij}^f(t)$ de la relation III.5.

$Vitesse(j) = \alpha\ Vitesse\_actuelle(j) + \beta\ Vitesse\_locale(j) + \gamma\ Vitesse\_globale(j)$******

// La vitesse actuelle relative à l'affectation initiale des machines pour cette particule, correspond à l'expression $m_{ij}^f(t)$ dans la relation III.5. La vitesse à appliquer correspond donc, au vecteur de mutation μm_{ij}^f .

Fin pour

Pour *j* = 1 à m // m correspondant au nombre de machines

Si P. vitesse(*j*) ≥ -0,5 ***et P***. vitesse(*j*) < 0 ***Alors***

Garder le même numéro de machine

Sinon Si P. vitesse(*j*) ≥ 0 ***et P***. vitesse(*j*) < 1 ***Alors***

Incrémenter le numéro de machine*******

Sinon Si P. vitesse(*j*) < - 0,5 ***Alors***

Décrémenter le numéro de machine*******

Sinon Prendre la machine qui nécessite le moins de temps d'exécution pour traiter l'opération en question

Fin Si

Fin pour

Mettre à jour l'individu *i*

Fin Pour

Trouver la meilleure fonction objectif obtenue

Enregistrer le meilleur individu

Fin Pour

Fin

***** L'**heuristique 1** consiste à comparer les affectations des machines de l'individu sélectionné à celles du meilleur voisin connu et de modifier la plus mauvaise affectation, correspondant au numéro de machine qui donne le temps d'exécution le plus long.

L'**heuristique 2** consiste à vérifier pour chaque opération exécutée, si le fait d'attendre la libération d'une autre machine, pourrait ou non améliorer le temps d'exécution de l'opération en cours.

****** Le calcul de la vitesse dans cet algorithme correspond au calcul de la formule III.5 concernant le vecteur de mutation μm_{ij}^{f}. Ici la notion de vitesse nous sert pour nous rapprocher de l'algorithme initial de l'OEP qui se base sur des changements de positions en ajustant la vitesse de la particule.

******* Pour l'incrémentation des machines, si le numéro à incrémenter est le dernier, choisir aléatoirement un autre numéro de machine.

Pour la décrémentation des machines, si le numéro à décrémenter est le plus bas, choisir aléatoirement un autre numéro de machine.

III.4 - Elaboration d'un ordonnancement d'ateliers de type job-shop flexible par la méthode basée sur l'essaim particulaire minimisant le Makespan

III.4.1 - Présentation des cas d'ateliers étudiés

Trois problèmes d'ordonnancement d'ateliers de types job-shop flexible sont traités dans cette partie. Deux de ces problèmes sont de type mono-opération.

Le premier, consiste à ordonnancer 20 produits sur 5 machines, toutes les machines étant utilisables à un moment donné, comme le montre le tableau 3.2. Chaque produit contient une seule opération et son exécution nécessite une des 5 machines disponibles.

Le deuxième problème de type mono-opération, consiste à ordonnancer 10 produits sur 6 machines. Le symbole « -- » indique que la machine choisie ne traite pas le produit en question, comme le montre le tableau 3.3.

Le dernier problème traité est un problème de type multi-opérations. Il concerne l'ordonnancement de 3 produits sur 5 machines en respectant l'ordre de priorité des opérations, comme le montre le tableau 3.4.

Tableau 3. 2 – Benchmark 20x5 relatif à un problème d'ordonnancement de type job-shop flexible mono-opération toutes les machines étant utilisables

	M_1	M_2	M_3	M_4	M_5
J_1	54	79	16	66	58
J_2	83	3	89	58	56
J_3	15	11	49	31	20
J_4	71	99	15	68	85
J_5	77	56	89	78	53
J_6	36	70	45	91	35
J_7	53	99	60	13	53
J_8	38	60	23	59	41
J_9	27	5	57	49	69
J_{10}	87	56	64	85	13
J_{11}	76	3	7	85	86
J_{12}	91	61	1	9	72
J_{13}	14	73	63	39	8
J_{14}	29	75	41	41	49
J_{15}	12	47	63	56	47
J_{16}	77	12	47	40	87
J_{17}	32	21	26	54	58
J_{18}	87	86	75	77	18
J_{19}	68	5	77	51	68
J_{20}	94	77	40	31	28

Tableau 3. 3 - Benchmark 10x6 relatif à un problème d'ordonnancement de type job-shop flexible mono-opération certaines machines n'étant pas utilisables

	M_1	M_2	M_3	M_4	M_5	M_6
J_1	10	07	06	13	--	--
J_2	02	05	08	12	07	11
J_3	09	05	06	12	06	17
J_4	--	08	--	10	15	--
J_5	15	12	08	06	10	09
J_6	09	05	07	13	--	--
J_7	14	13	08	20	14	17
J_8	07	16	05	11	17	09
J_9	09	16	08	11	--	--
J_{10}	08	14	06	18	21	14

Tableau 3. 4 - Benchmark 3x5 relatif à un problème d'ordonnancement de type job-shop flexible multi-opérations

		M_1	M_2	M_3	M_4	M_5
J_1	O_{11}	1.50	9.50	3.12	4.91	4.50
	O_{21}	3.00	4.50	1.75	4.70	4.50
	O_{31}	4.50	7.00	1.75	4.50	3.75
J_2	O_{12}	1.50	4.50	4.50	3.25	6.37
	O_{22}	1.50	8.25	4.91	4.50	3.75
	O_{32}	4.50	4.50	1.75	2.00	4.50
J_3	O_{13}	1.50	4.50	4.91	3.25	3.00
	O_{23}	4.50	9.50	1.75	4.50	3.75

Ces trois problèmes différents, de part le nombre de jobs à exécuter et du fait qu'ils sont mono ou multi-opérations, sont résolus par l'algorithme BOEP dans le but de minimiser la date de fin effective de la dernière opération, appelée également Cmax ou Makespan.

III.4.2 - Résultats de mise en œuvre de la méthode BOEP

Pour les trois exemples traités, des essaims de 50 particules sont générés et des voisinages de 5 particules sont sélectionnés. L'algorithme est exécuté pour chacun des problèmes sur 1000 générations d'essaims, les coefficients de confiance, α, β et γ, choisis pour ces problèmes ont respectivement pour valeurs 0.2, 0.3 et 0.5.

D'autres valeurs de α, β et γ seront choisies ultérieurement et leur impact sur la convergence de ces problèmes étudié.

Pour le premier exemple 20x5, la convergence a eu lieu à la $232^{ème}$ génération comme le montre la figure 3.2. et la meilleure solution obtenue est représentée par le diagramme de Gantt de la figure 3.3.

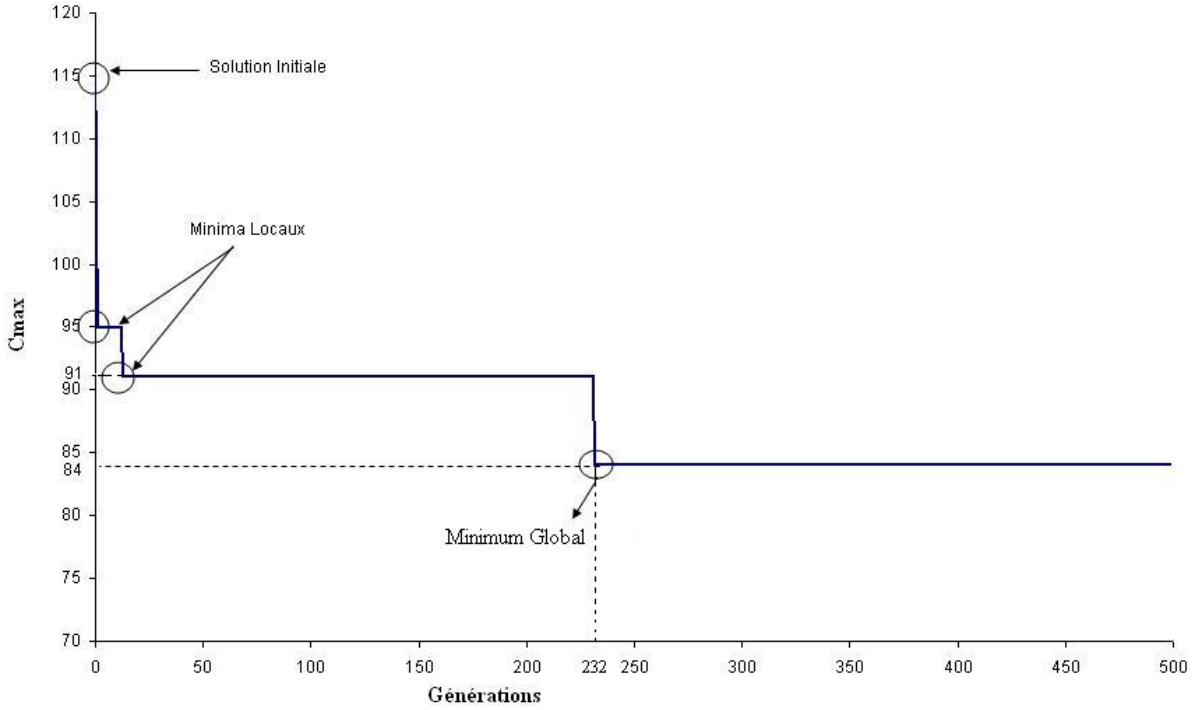

Figure 3. 2 - Evolution du Cmax à travers les générations pour un problème FJSP 20x5

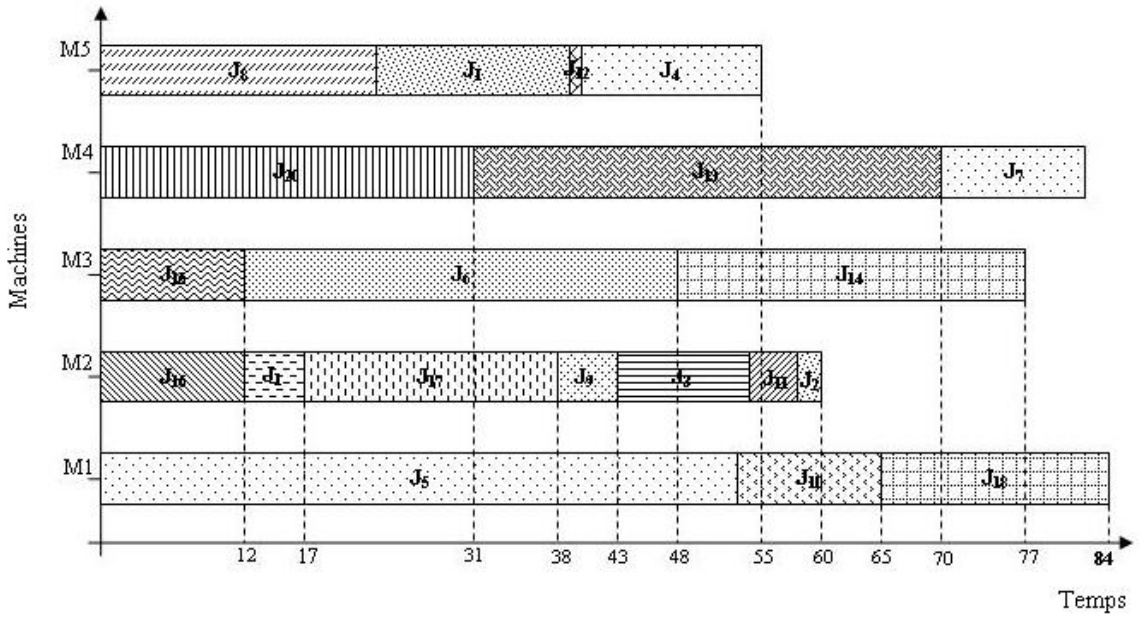

Figure 3. 3 - Diagramme de Gantt de la meilleure solution pour le problème 20x5

Pour le deuxième exemple 10x6, la convergence a eu lieu à la 63ème génération et est illustré dans la figure 3.4.

Le diagramme de Gantt correspondant est présenté dans la figure 3.5.

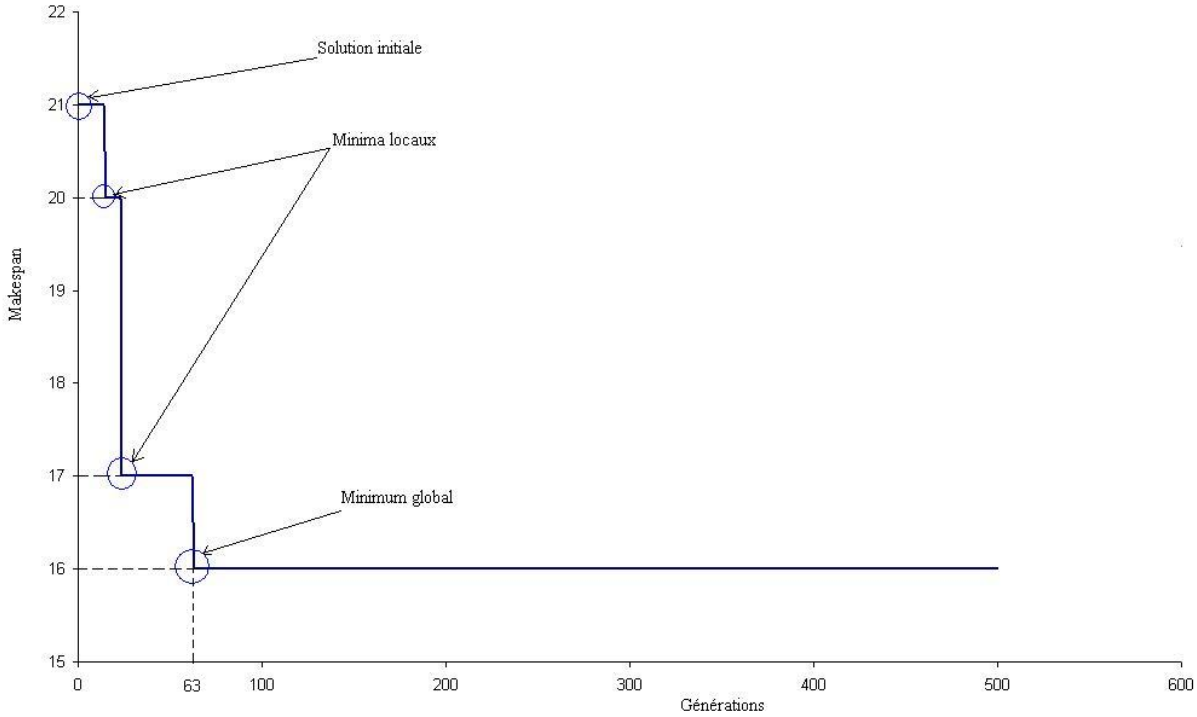

Figure 3. 4 - Evolution du Cmax à travers les générations pour un problème FJSP 10x6

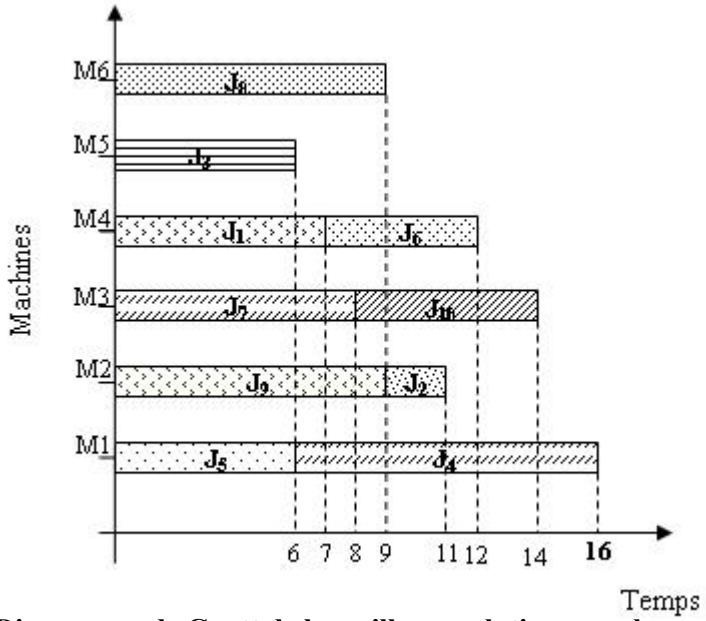

Figure 3. 5 - Diagramme de Gantt de la meilleure solution pour le problème 10x6

Quant au troisième exemple 3x5, traitant du problème multi-opérations, la convergence a eu lieu à la 12ème génération, comme montré dans la figure 3.6.

Le diagramme de Gantt, relatif à la meilleure solution, est présenté dans la figure 3.7.

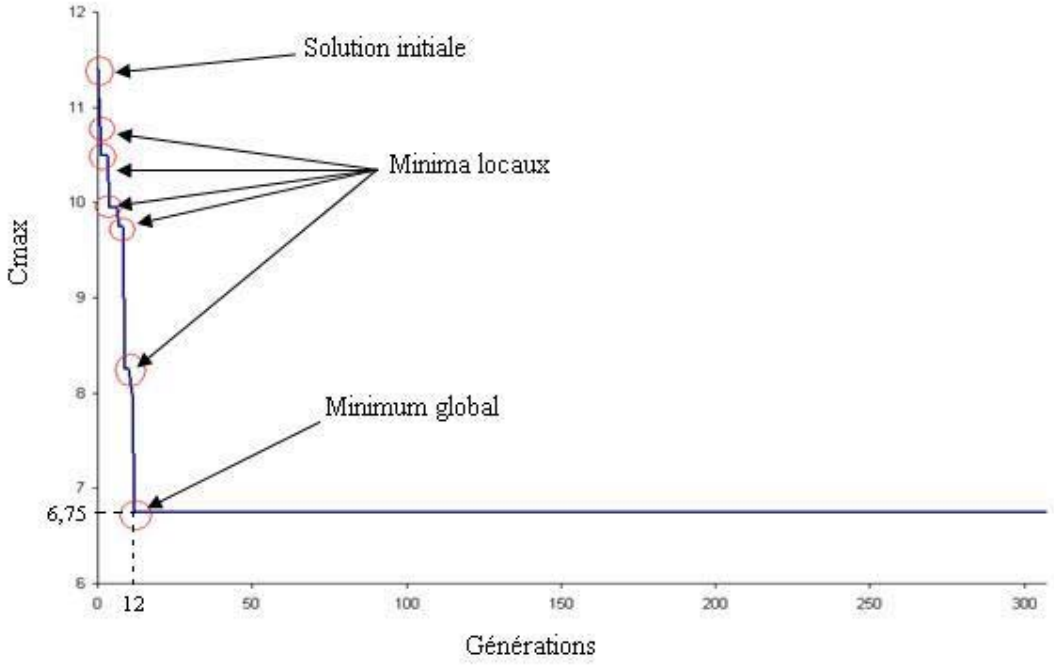

Figure 3. 6 - Evolution du Cmax à travers les générations pour un problème FJSP 3x5

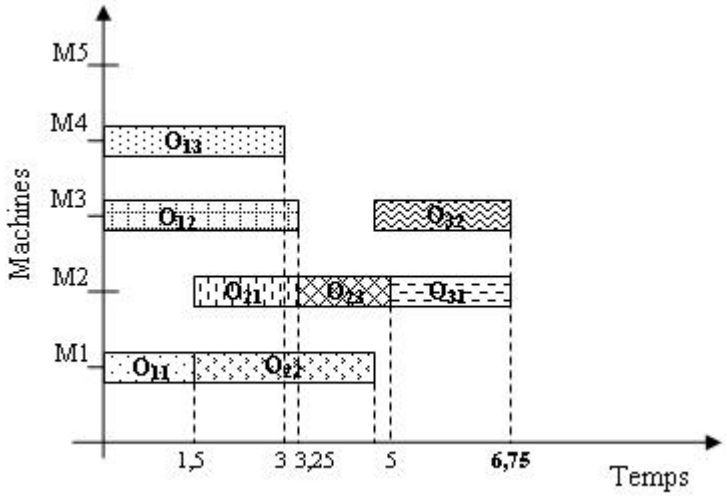

Figure 3. 7 - Diagramme de Gantt de la meilleure solution pour le problème 3x5

Les temps d'arrêts représentés, indiquent l'attente pour chaque opération, de la fin de celle qui la précède. Ainsi, L'opération O32 sur la machine 3 ne peut commencer que si l'opération O22 sur la machine 1 est terminée. Des temps d'attente sont ainsi générés.

III.4.3 - Influence du choix des coefficients α, β et γ sur les résultats obtenus

Le choix des coefficients α, β et γ fixés à 0.2, 0.3 et 0.5, accorde plus d'importance au 3ème coefficient, qui permet la comparaison de la position actuelle de la particule à son voisinage, qu'aux deux autres, qui permettent de prendre en compte la position actuelle de la particule et la comparaison de cette position avec la meilleure position connue.

Dans ce qui suit, nous reprenons les trois exemples traités précédemment en effectuant un choix aléatoire de ces coefficients [Clerc et al, 09]. Les résultats correspondant sont présentés dans les figures 3.8, 3.9 et 3.10.

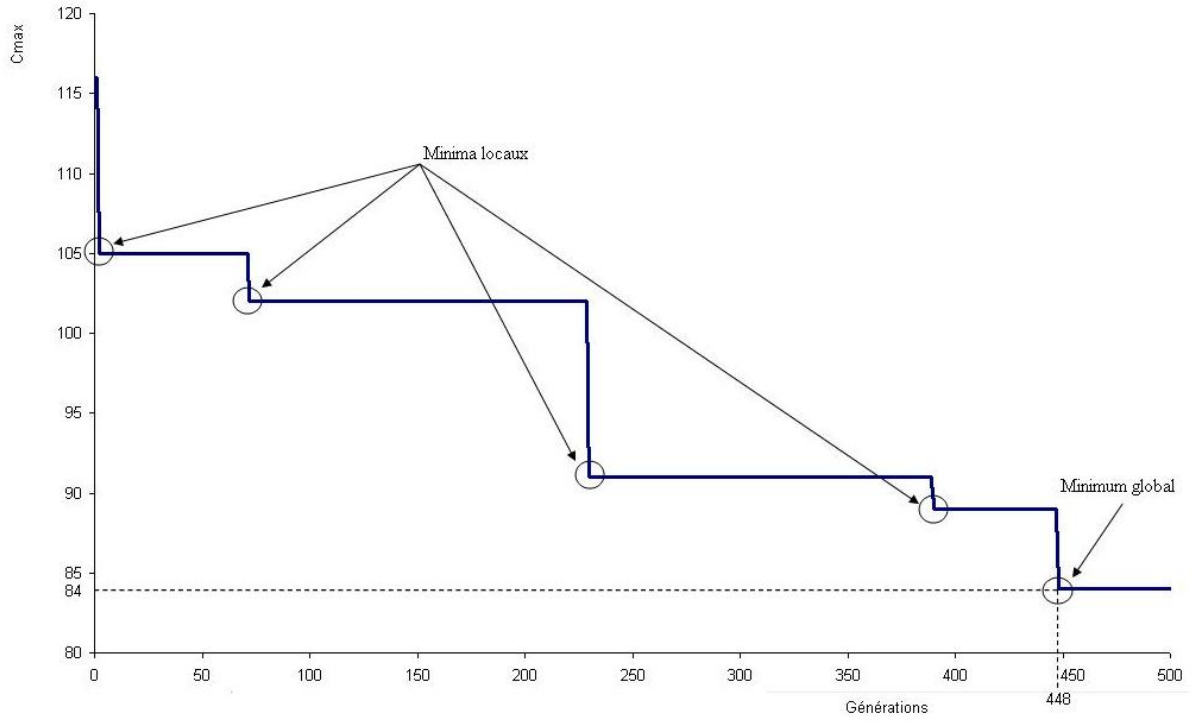

Figure 3. 8 - Evolution du Cmax à travers les générations pour un problème FJSP 20x5 pour un choix aléatoire des coefficients α, β et γ

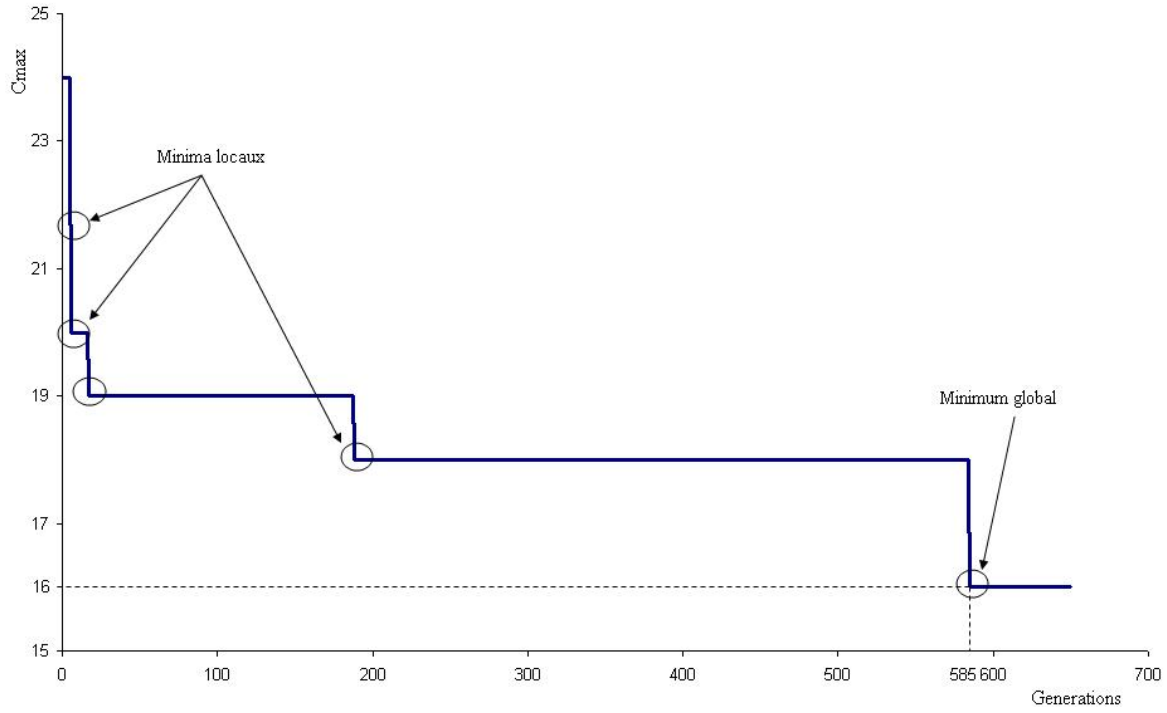

Figure 3. 9 - Evolution du Cmax à travers les générations pour un problème FJSP 10x6 pour un choix aléatoire des coefficients α, β et γ

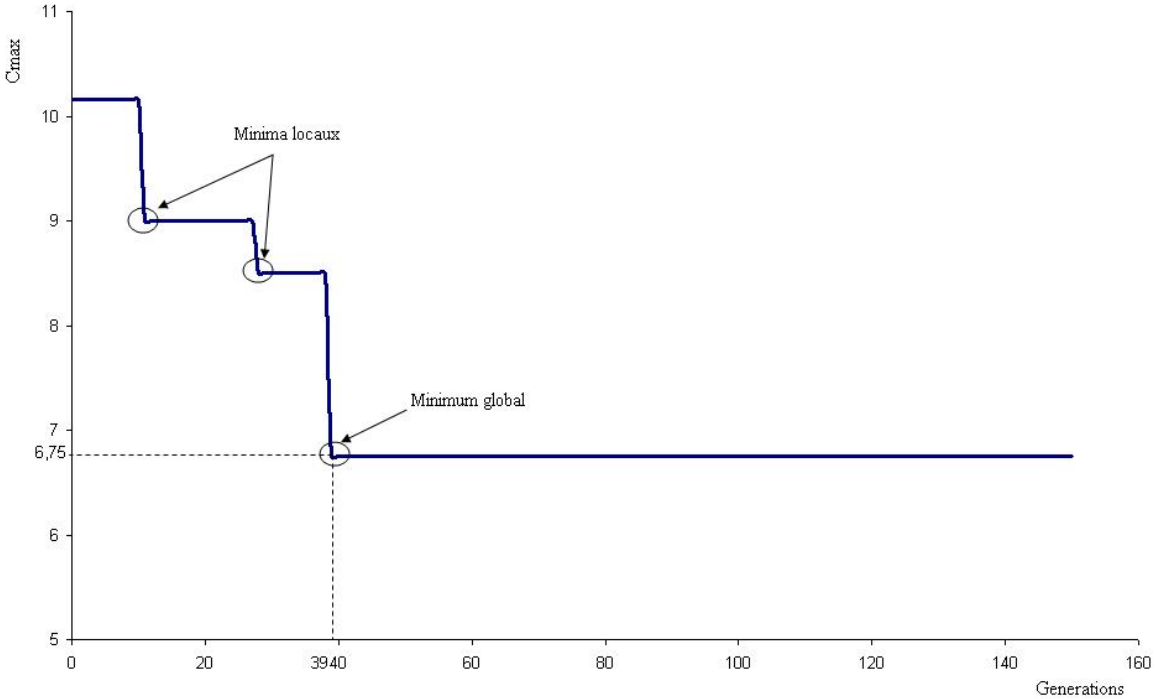

Figure 3. 10 - Evolution du Cmax à travers les générations pour un problème FJSP 3x5 pour un choix aléatoire des coefficients α, β et γ

Nous remarquons pour ces trois exemples, que les résultats obtenus après modification des coefficients de confiance sont les mêmes mais avec une convergence plus lente vers la solution optimale.

Ainsi, dans le premier exemple traitant de l'ordonnancement de 20 produits mono-opération sur 5 machines, la convergence a lieu à la 448ème génération dans le cas où les coefficients α, β et γ sont générés aléatoirement alors qu'elle a lieu à la 232ème génération dans le cas où ces coefficients sont préalablement fixés.

Dans le deuxième exemple, traitant de l'ordonnancement de 10 produits mono-opération sur 6 machines, la convergence à lieu à la 585ème génération pour α, β et γ générés aléatoirement alors qu'elle a lieu à la 63ème génération quand ces coefficients sont fixés au départ.

Le troisième exemple, traitant de l'ordonnancement de 3 produits multi-opérations sur 5 machines, la convergence a lieu à la 39ème génération pour α, β et γ générés aléatoirement alors qu'elle a lieu à la 12ème génération quand ces coefficients sont fixés au départ.

III.4.4 - Influence de la modification du voisinage sur les résultats obtenus

Dans une première étape de résolution, un voisinage de 5 particules est choisi pour un essaim de 50 particules. Dans ce qui suit, nous nous proposons d'étudier l'impact de la modification du voisinage sur les résultats obtenus.

Un exemple de voisinage de 10 particules est pris en compte et les résultats relatifs à cette modification sont présentés dans les figures 3.11, 3.12 et 3.13.

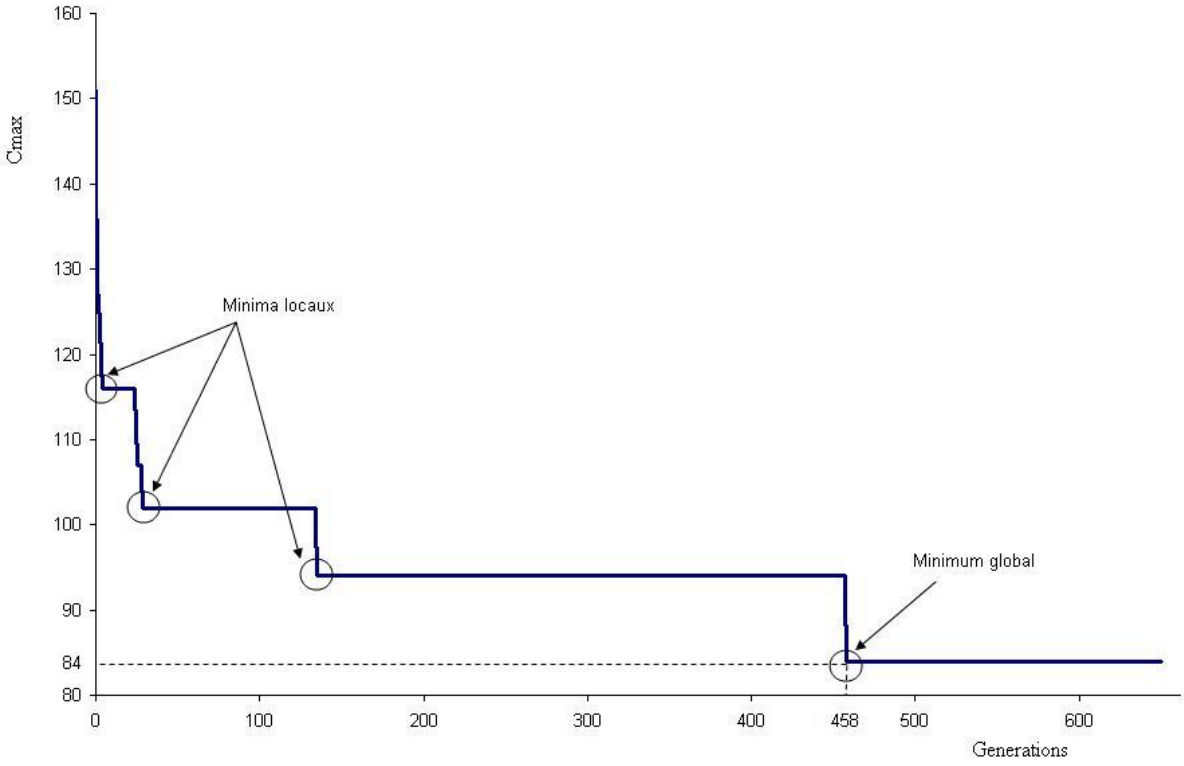

Figure 3. 11 - Evolution du Cmax à travers les générations pour un problème FJSP 20x5 pour un voisinage de 10 particules

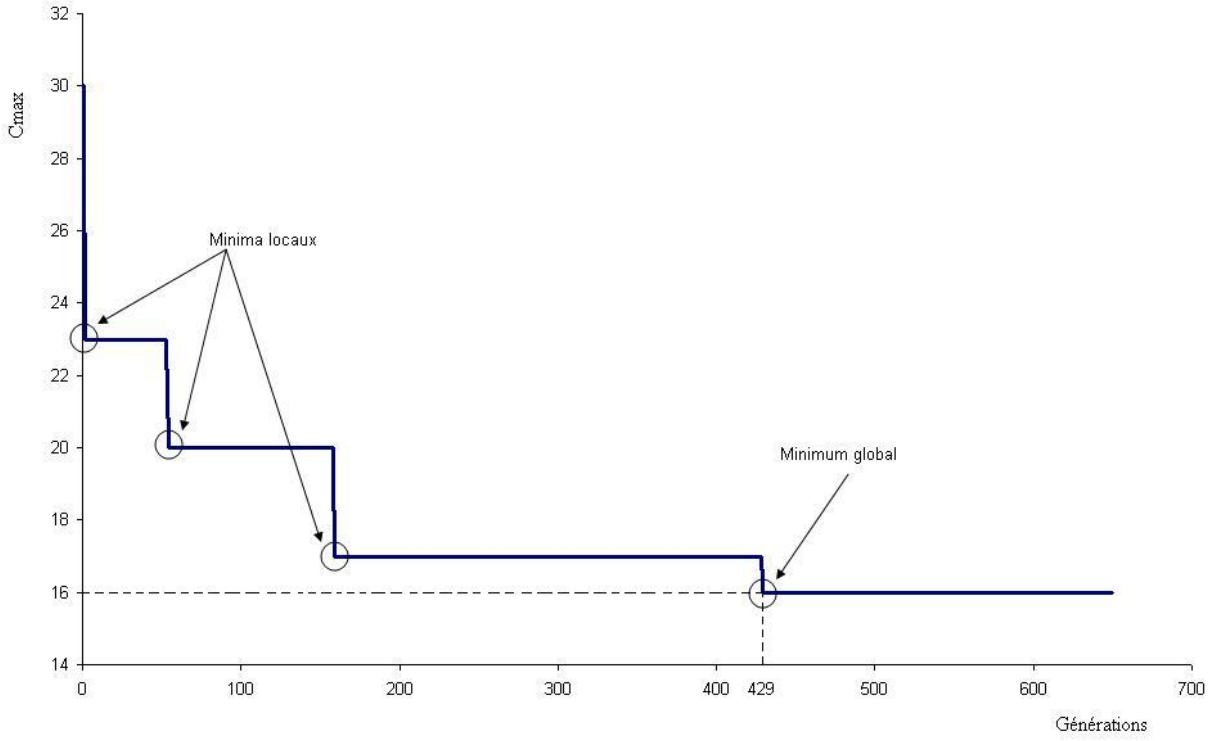

Figure 3. 12 - Evolution du Cmax à travers les générations pour un problème FJSP 10x6 pour un voisinage de 10 particules

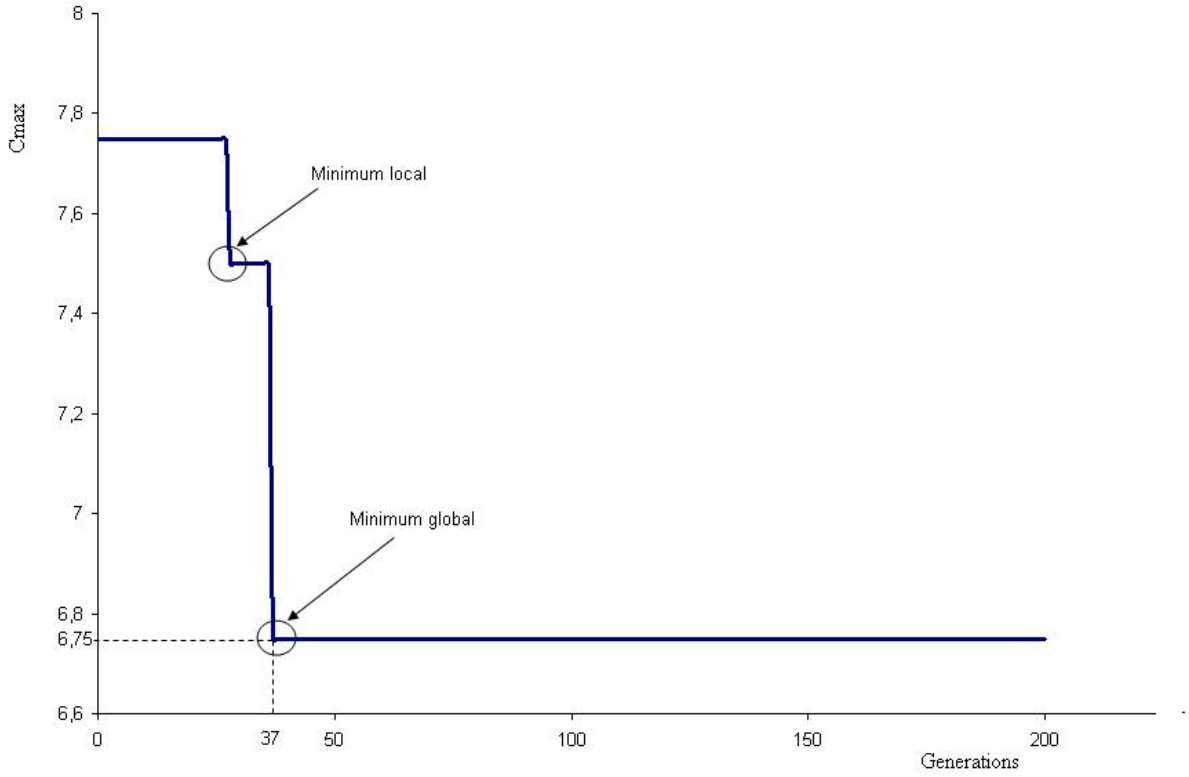

Figure 3. 13 - Evolution du Cmax à travers les générations pour un problème FJSP 3x5 pour un voisinage de 10 particules

Pour les trois exemples traités, en prenant en considération un voisinage de 10 particules, le Cmax obtenu est le même que pour un voisinage de 5 particules. La différence réside dans la rapidité de convergence. En effet, la méthode utilisant un voisinage de 10 particules est plus lente que celle utilisant un voisinage de 5.

Ainsi, dans le premier exemple traitant de l'ordonnancement de 20 produits mono-opération sur 5 machines, la convergence a lieu à la 232ème génération dans le cas où le voisinage est de 5 particules alors qu'elle a lieu à la 458ème génération dans le cas où le voisinage est de 10.

Dans le deuxième exemple, traitant de l'ordonnancement de 10 produits mono-opération sur 6 machines, la convergence à lieu à la 63ème génération dans le cas où le voisinage est de 5 particules alors qu'elle a lieu à la 429ème génération dans le cas où le voisinage est de 10.

Le troisième exemple, traitant de l'ordonnancement de 3 produits multi-opérations sur 5 machines, la convergence a lieu à la 12ème génération dans le cas où le voisinage est de 5 particules alors qu'elle a lieu à la 37ème génération dans le cas où le voisinage est de 10.

Les résultats des trois méthodes étudiées sont consignés dans le tableau III.5 suivant.

Tableau 3. 5 – Résultats comparatifs des différentes variantes de la méthode BOEP

	Problème 20x5		Problème 10x6		Problème 3x5	
	Cmax	Convergence	Cmax	Convergence	Cmax	Convergence
Méthode BOEP voisinage de 5 particules et α, β, γ fixes	84	232	16	63	6,75	12
Méthode BOEP voisinage de 5 particules et α, β, γ variables	84	448	16	585	6,75	39
Méthode BOEP voisinage de 10 particules	84	458	16	429	6,75	37

III.4.5 - Comparaison des résultats avec ceux obtenus par les algorithmes génétiques

Les résultats du tableau III.6 suivant, obtenus par application de l'algorithme d'optimisation par essaim particulaire, ont été comparés avec ceux obtenus par application des algorithmes génétiques [Saad, 07] dans le but de minimiser le Cmax.

Tableau 3. 6 -Tableau comparatif des résultats relatifs aux mises en œuvre de la méthode basée sur l'optimisation par essaim particulaire (BOEP) et des algorithmes génétiques (AG) pour les problèmes FJSP

	Problème 20 x 5		Problème 10 x 6		Problème 3x5	
Méthodes d'optimisation	AG	BOEP	AG	BOEP	AG	BOEP
Cmax	84	84	16	16	6,75	6,75
Convergence	905	232	20	63	14	12

Ce tableau montre que pour les trois problèmes considérés, la méthode d'optimisation par essaim particulaire et la méthode des algorithmes génétiques ont abouti aux mêmes résultats, Cmax = 84 pour le premier exemple, Cmax = 16 pour le deuxième et Cmax = 6,75 pour le troisième avec un nombre de générations identiques. La convergence de l'algorithme BOEP s'avère toutefois plus rapide pour le premier et le dernier problème mais l'est un peu moins pour le deuxième. Ceci s'explique certainement par le fait que, dans l'exemple 10x6, la non utilisation de certaines machines pour des produits spécifiques pose problème au niveau de la méthode d'optimisation par essaim particulaire.

III.5 - Comparaison de l'efficacité des AG et de la méthode BOEP pour la résolution de problèmes flow-shop en industries pharmaceutiques

III.5.1 - Efficacité de la méthode BOEP – Position du problème

Dans cette partie, nous reprenons l'exemple de l'atelier de conditionnement de type flow-shop en industries pharmaceutiques traité au premier chapitre pour lequel 16, puis 30 produits sont traités sur 2 lignes de conditionnement.

Après avoir résolu ces deux problèmes par utilisation des algorithmes génétiques, nous nous proposons de vérifier l'efficacité de la méthode d'optimisation par essaim particulaire pour ce type d'ateliers.

Considérons les notations suivantes relatives à ce problème :

$P_B^f(t)$: meilleure position connue du voisinage de la particule *f* à l'itération *t*

$P_b^f(t)$: meilleure position connue de la particule à l'itération t

$L_{ik}^f(t)$: ligne de conditionnement sélectionnée pour le traitement de l'opération O_{ik} à l'itération t pour la position actuelle de la particule f

$\left(L_{ik}^f(t)\right)_b$: ligne de conditionnement sélectionnée pour le traitement de l'opération O_{ik} à l'itération t pour la meilleure position connue de la particule f

$\left(L_{ik}^f(t)\right)_G$: ligne de conditionnement sélectionnée pour le traitement de l'opération O_{ik} à l'itération t pour la meilleure position du voisinage de la particule f

ΔL_f : changements appliqués à la particule f relatifs à l'affectation des lignes de conditionnement

μL_f : vecteur de mutation appliqué à la particule f permettant la mise à jour de la sa Position

Ainsi, pour chaque particule, les changements à effectuer pour passer d'une position $P^f(t)$ à une autre $P^f(t+1)$ doivent respecter la relation III.6 suivante, α, β et γ étant des coefficients de confiance :

$$P^f(t+1) = P^f(t) + \Delta L_f \qquad \text{(III.6)}$$

avec :

$$\Delta L_f = f(\mu L_f) \quad \text{(III.7)}$$

et:

$$\mu L_f = \alpha[L_{ik}^f(t)] + \beta[(L_{ik}^f(t))_b - L_{ik}^f(t) + \gamma[(L_{ik}^f(t))_G) - L_{ik}^f(t)] \quad \text{(III.8)}$$

Ainsi, après application du vecteur de mutation μL_f à la particule *f*, les critères F_1 et F_2, formulés dans le deuxième chapitre au paragraphe II.5.2 et correspondants respectivement au coût total de production et au coût d'arrêt et de non utilisation des lignes de conditionnement, sont repris et la fonction objectif *F* réévaluée, $F = \alpha_1 F_1 + \alpha_2 F_2$.

III.5.2 - Résultats de l'application de l'algorithme BOEP

L'algorithme BOEP, basé sur la méthode d'optimisation par essaim particulaire, est appliqué au problème d'ordonnancement flow-shop en industries pharmaceutiques. Ainsi, une population d'essaim est générée par affectation aléatoire de chaque produit à une ligne de conditionnement. Par la suite, une particule de l'essaim est sélectionnée et un voisinage de cette particule est choisi.

Une heuristique pour l'amélioration de l'affectation des lignes est ensuite utilisée. Elle consiste à choisir, pour le produit ayant le plus grand temps d'exécution, laquelle des deux lignes nécessite le moins de temps d'exécution et à faire le changement si nécessaire.

A cette heuristique locale est ajoutée une heuristique globale qui consiste à changer l'affectation des lignes de conditionnement en appliquant ΔL_f respectant les conditions suivantes :

Pour chaque élément de ΔL_f :

- si la valeur de μL_f est > 0 alors garder le même numéro de ligne,
- si la valeur de μL_f est < 0 alors changer aléatoirement le numéro de la ligne,
- sinon, prendre la machine qui nécessite le moins de temps pour le traitement du produit.

Ces étapes sont ainsi répétées jusqu'à ce qu'un nombre d'itérations fixé préalablement soit atteint.

Les résultats relatifs à l'application de cet algorithme sur les problèmes traitant de l'exécution de 16 puis de 30 produits sur deux lignes de conditionnement sont présentés dans ce qui suit.
Ainsi, pour le premier exemple, pour 500 générations et une population de 50 particules avec un voisinage de 5 particules, la méthode basée sur l'optimisation par essaim particulaire est appliquée jusqu'à l'obtention du meilleur individu avec le coût le plus bas.
La meilleure particule pour laquelle le minimum global, 1207, est obtenu est présentée dans la figure III.14. Les différents types de minima sont ainsi présentés avec convergence à la 49ème génération.

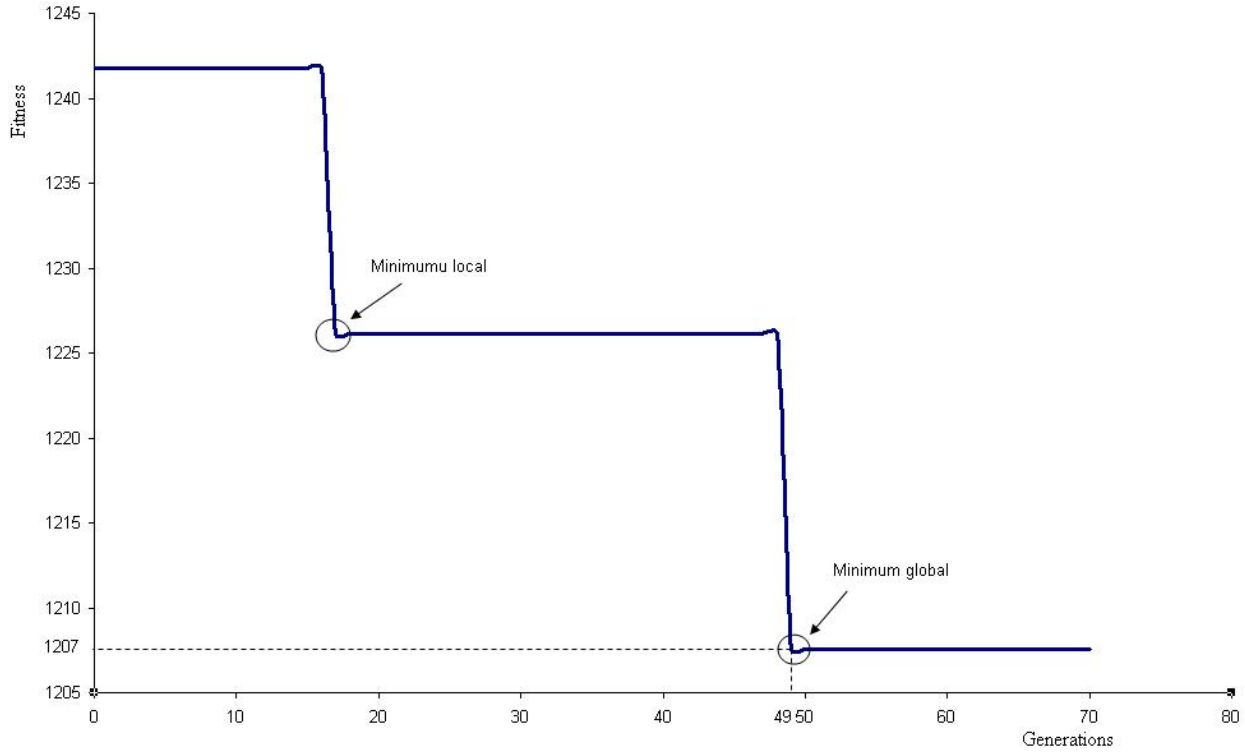

Figure 3. 14 - Evolution des coûts à travers les générations pour le problème d'ordonnancement 16x2 en industries pharmaceutiques

Le diagramme de Gantt, présenté dans la figure suivante, représente le meilleur minimum obtenu après application de la méthode basée sur l'optimisation par essaim particulaire sur le problème 16 x 2.
Nous obtenons, ainsi, une répartition presque parfaite des seize produits sur les deux lignes de conditionnement avec un Cmax de 213.

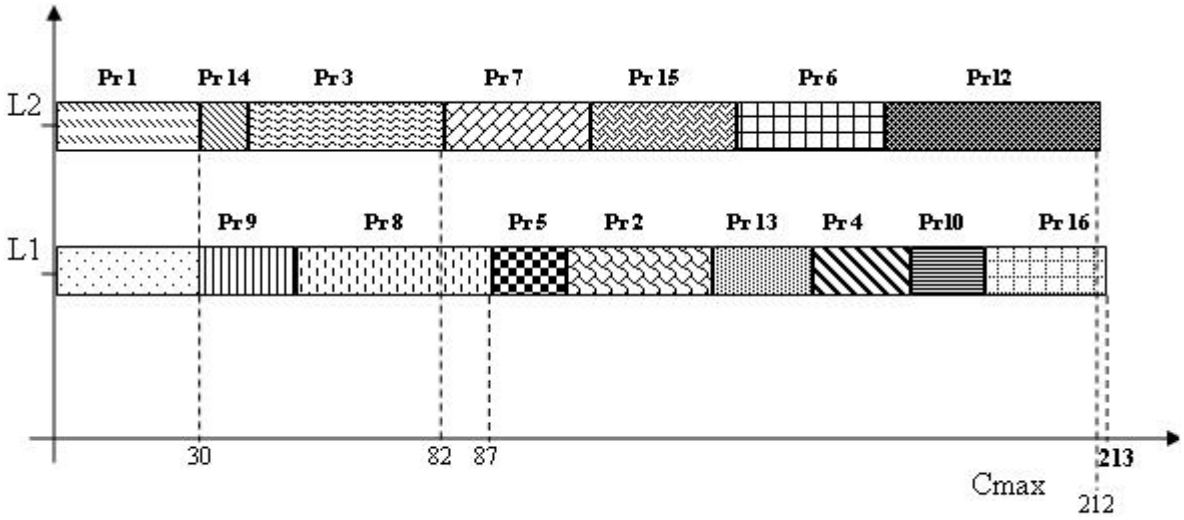

Figure 3. 15 - Diagramme de Gantt relatif au meilleur individu pour le problème 16 x 2 par application de la méthode BOEP

Pour le deuxième exemple, pour 500 générations et une population de 50 particules avec un voisinage de 5 particules, la méthode basée sur l'optimisation par essaim particulaire est appliquée jusqu'à l'obtention du meilleur individu avec le coût le plus bas.

La meilleure particule pour laquelle le minimum global, 2543, est obtenu, est présentée dans la figure III.16. Les différents types de minima sont ainsi présentés avec une convergence à la 128ème génération.

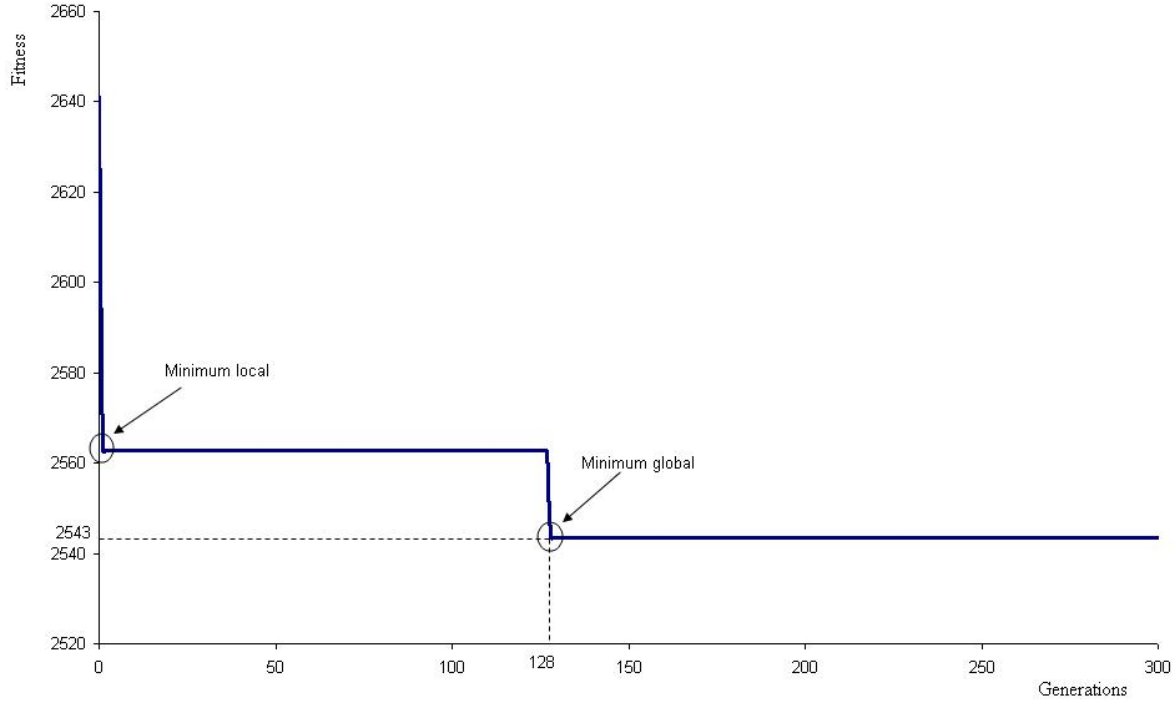

Figure 3. 16 - Evolution des coûts à travers les générations pour le problème d'ordonnancement 30x2 en industries pharmaceutiques

Le diagramme de Gantt, présenté dans la figure III.17, représente le meilleur minimum obtenu après application de la méthode basée sur l'optimisation par essaim particulaire sur le problème 30 x 2.

Nous obtenons, ainsi, une répartition des trente produits sur les deux lignes de conditionnement avec un Cmax de 498.

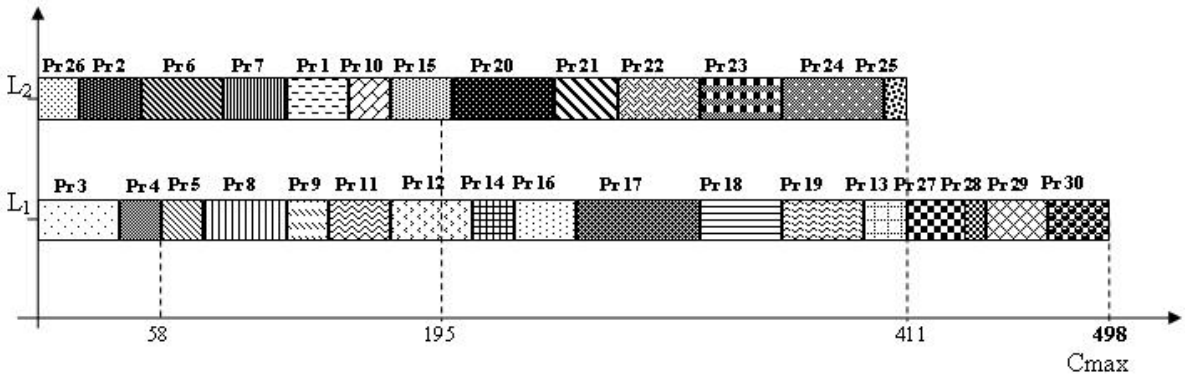

Figure 3. 17 - Diagramme de Gantt relatif au meilleur individu pour le problème 30 x 2 par application de la méthode BOEP

L'utilisation de la méthode d'optimisation par essaim particulaire pour la résolution de ces deux problèmes d'ordonnancement de type flow-shop en industries pharmaceutiques ont confirmé les résultats obtenus dans la première partie de ce chapitre pour des ateliers de type job-shop flexible.

La comparaison de ces résultats avec ceux obtenu dans le deuxième chapitre, par application des algorithmes génétiques sont présentés dans le tableau III.7 suivant.

Tableau 3. 7 - Tableau comparatif des résultats relatifs aux mises en œuvre de la méthode BOEP et des AG pour les problèmes flow-shop en industries pharmaceutiques

	Problème 16 x 2		**Problème 30 x 2**	
Méthodes d'optimisation	AG	BOEP	AG	BOEP
Fonction Fitness	1260	1207	2643	2543
Convergence	445	49	161	128
Cmax	230	213	502	498

Ce tableau montre que, pour les deux types de problèmes considérés, la méthode basée sur l'optimisation par essaim particulaire a abouti à des résultats plus intéressants que ceux obtenus par l'utilisation de la méthode des algorithmes génétiques.

Nous remarquons ainsi, pour ces deux exemples, qu'avec la méthode BOEP nous obtenons une convergence plus rapide (49 générations par rapport à 445 pour l'exemple 16x2 et 128 par rapport à 161 pour l'exemple 30x2) et un meilleur minimum global (1207 par rapport à 1260 et 2543 par rapport à 2643).
Ces résultats, associés à ceux obtenus dans la première partie de ce chapitre, traitant de l'ordonnancement d'exemples de type job-shop flexible, nous permettent de valider l'utilisation de la méthode basée sur l'optimisation par essaim particulaire dans le cas discret.
De plus, pour chacun des exemples, le Cmax obtenu par la méthode BOEP est toujours meilleur que celui obtenu par la méthode AG.

III.6 - Conclusion

La première partie de ce chapitre traite de la présentation de la méthode d'optimisation par essaim particulaire, son utilisation dans le cas continu et les différentes mises à jour et formulations nécessaires pour son adaptation au cas discret, donnant ainsi naissance à la méthode BOEP (Basée sur l'Optimisation par Essaim Particulaire).
Trois exemples portant sur l'ordonnancement d'ateliers de type job-shop flexible sont traités et deux variantes de la méthode BPSO par la suite essayées ; l'une utilise des coefficients de confiance aléatoires et l'autre une modification du voisinage. Ces deux variantes ont donné des résultats similaires à la méthode initiale mais avec des temps de convergence plus grands.
Dans la deuxième partie de ce chapitre, le problème d'ordonnancement d'atelier flow-shop en industries pharmaceutiques, étudié dans le deuxième chapitre par application des algorithmes génétiques, est repris avec utilisation de la méthode basée sur l'optimisation par essaim particulaire. Les résultats obtenus par cette dernière sont largement positifs et nous conduisent à conclure que la méthode BPSO est comparable à, sinon meilleure que, la méthode des AG en ce qui concerne les deux types de problèmes traités.

Conclusion générale

La résolution de problèmes d'ordonnancement d'ateliers de type flow-shop en industries pharmaceutiques et d'ateliers de type job-shop flexible constitue la principale contribution de nos travaux consignés dans ce mémoire.

Deux méthodes d'optimisation ont été considérées : la méthode des algorithmes génétiques, qui a déjà prouvé son efficacité dans différents domaines tels que celui de l'optimisation, de l'intelligence artificielle, de la robotique, du traitement du signal et de l'économie et la méthode d'optimisation par essaim particulaire, qui a fait ses preuves dans le cas de problèmes continus tels que les commandes des machines ou dans la robotique et qui devient intéressante à utiliser dans le domaine discret.

Dans ce mémoire, nous nous sommes proposés d'étudier de développer et d'exploiter ces deux méthodes et de comparer leurs résultats respectifs et leurs efficacités.

Nous nous sommes, tout d'abord, intéressés à l'étude de l'ordonnancement dans sa globalité, à savoir ses différentes composantes (tâches, ressources, contraintes, critères), ses différents types d'ateliers (job-shop, flow-shop et open-shop) et les méthodes d'optimisation le plus souvent utilisées pour sa résolution, allant des méthodes exactes (programmation linéaire, programmation dynamique) aux méthodes approchées (recuit simulé, recherche tabou, algorithmes génétiques et algorithmes de colonies de fourmis).

Par la suite, notre intérêt s'est porté sur l'étude de la méthode des algorithmes génétiques et son utilisation pour la résolution de problèmes multi-objectifs d'ordonnancement de type flow-shop en industries pharmaceutiques. Pour cela, les différentes caractéristiques relatives à ce type d'industries et plus particulièrement du poste de conditionnement sont présentées et les contraintes pouvant y survenir détaillées, le coût total de production et le coût d'arrêt et de non utilisation des lignes de conditionnement étant les deux critères à minimiser.

Notre contribution dans ce sens touche la formulation du problème, la proposition d'un codage (CLOS), d'opérateurs de croisement et de mutation ainsi que d'un algorithme spécifique à la résolution de problèmes d'ordonnancement en industries pharmaceutiques.

Les résultats obtenus, relatifs à deux problèmes étudiés, ont confirmé la capacité des algorithmes génétiques à s'approcher de la solution optimale.

Dans le dernier chapitre, nous nous sommes intéressés à l'étude de la méthode d'Optimisation par Essaim Particulaire (OEP). Ce chapitre est décomposé en deux parties. Après l'étude de problèmes d'ordonnancement de type job-shop flexible et de problèmes d'ordonnancement de type flow-shop en industries pharmaceutiques, sont présentées la méthode OEP ainsi que celle relative aux problèmes job-shop flexibles.

La formulation générale de la méthode OEP dans le cas continu est introduite avant de passer aux différentes conversions nécessaires pour le passage vers le domaine discret, tout en gardant l'esprit coopératif de cette méthode.

Notre contribution dans ce sens a été la proposition d'une nouvelle structure de particule pouvant être utilisable, d'une manière générale, pour la représentation des particules dans les problèmes d'ordonnancement. Nous en avons déduit un algorithme Basé sur l'Optimisation par Essaim Particulaire (BOEP) montrant l'intérêt de l'utilisation de cette méthode dans le cas discret.

Pour les trois exemples mono-objectif d'ordonnancement de problèmes job-shop flexibles, pour la minimisation du Cmax étudiés, les résultats obtenus comparés avec ceux issus de la méthode des Algorithmes Génétiques (AG) sont satisfaisants vu que le Cmax obtenu pour tous les cas est le même pour les deux méthodes, et que la méthode BOEP converge dans deux cas sur trois plus rapidement que la méthode des AG.

Les résultats relatifs à l'efficacité de cette nouvelle méthode comparés à ceux relatifs aux algorithmes génétiques pour deux problèmes d'ordonnancement en industries pharmaceutiques, ont montré que la méthode BOEP permet d'obtenir des solutions meilleures et une convergence plus rapide que celles obtenues par la méthode AG.

Toutes ces conclusions, confirment que la méthode d'optimisation par essaim particulaire a fait ses preuves, autant que les algorithmes génétiques ou les algorithmes de colonies de fourmis, pour la résolution de problèmes d'ordonnancement.

Les résultats obtenus étant encourageants, il serait intéressant de pouvoir développer certains autres aspects dans nos travaux envisagés en perspectives :

- tenir compte d'autres critères à résoudre au niveau des industries pharmaceutiques tels que les retards de livraison des produits ou les temps d'arrêts survenant lors des maintenances des lignes de conditionnement,

- comparer la méthode BOEP avec d'autres méthodes approchées telles que le recuit simulé, la recherche tabou ou les algorithmes de colonies de fourmis,
- étudier l'impact de l'hybridation de cette méthode avec d'autres méthodes approchées.

Bibliographie

[Abido, 02] M.A. Abido, « Optimal power flow using particle swarm optimization ». Electrical Power and Energy Systems, vol. 24, pp. 563-571, 2002.

[Baptiste et al, 96] P. Baptiste et C. Le Pape, « A constraint-Based Branch and Bound Algorithm for Preemptive Job-Shop Scheduling ». 5th IEEE, International Symposium on Assembly and Task Planning, Besançon, 1996.

[Bel et al, 2001] G. Bel et J-B. Cavaillé, « Ordonnancement de la production ». Editions Hermès, Paris, 2001.

[Bellman, 86] R.E. Bellman, « The Bellman continuum ». Editions Robert S. Roth, 1986.

[Bertel et al, 01] S. Bertel et J. C. Billaut, « Problème d'ordonnancement multicritère dans un flow-shop hybride avec recirculation ». 3ème Conférence Francophone de MOdélisation et SIMulation, MOSIM'01, 25-27 avril 2001, Troyes.

[Borne et al, 90] P. Borne, G.D. Tanguy, J.P. Richard, F. Rotella et I. Zambettakis, « Commande et optimisation des processus ». Edition Technip, Paris, 1990.

[Blazewicz et al, 96] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt et J. Weglarz, « Scheduling computer and manufacturing process ». Springer, Berlin, 1996.

[Boukef et al, 06] H. Boukef, F. Tangour et M. Benrejeb, « Sur la formulation d'un problème d'ordonnancement de type flow-shop d'ateliers de production en industries pharmaceutiques ». Journées Tunisiennes d'Electrotechnique et d'Automatique, JTEA'06, Hammamet, 2006.

[Boukef et al, 06] H. Boukef, F. Tangour, M. Benrejeb et P. Borne, « Nouveau codage pour la résolution de problèmes d'ordonnancement d'ateliers de type flow-shop par les algorithmes génétiques ». 7ème Conférence Internationale des Sciences et Techniques de l'Automatique, STA'06, Hammamet, pp. 1-14, 2006.

[Boukef et al, 07] H. Boukef, M. Benrejeb et P. Borne, « A proposed genetic algorithm coding for flow-shop scheduling problems ». International Journal of Computers, Communications and Control, vol. 2, n° 3, pp. 229-240, 2007.

[Boukef et al, 08] H. Boukef, M. Benrejeb et P. Borne, « Flexible Job-shop Scheduling Problems Resolution Inspired From Particle Swarm Optimization ». Studies in Informatics and Control, vol. 17, n° 3, pp. 241-252, 2008.

[Boukef et al, 09] H. Boukef, M. Benrejeb et P. Borne, « Genetic Algorithm and Based Particle Swarm Optimization Comparison for Solving a Flow-shop Multiobjective Scheduling Problem in Pharmaceutical Industries ». International Review of Automatic Control, vol. 2, n°2, pp. 223-228

[Breit, 06], J. Breit, « A polynomial-time approximation scheme for the two-machine flow-shop scheduling problem with an availability constraint ». Computers & Operations Research, vol. 33, pp. 2143–2153, 2006.

[Bruker et al, 90] P. Bruker et R. Schlie, « Job-shop scheduling with multi-purpose machines ». Computing, vol. 45, pp. 369–375, 1990.

[Carlier et al, 88] J. Carlier et P. Chrétienne, « Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes ». Edition Masson, Paris, 1988.

[Carlier et al, 84] J. Carlier, P. Chrétienne et C. Girault, « Modelling scheduling problems with Petri nets. Advanced studies in Petri nets ». Lecture notes in Computer Science, Springer Verlag, Paris, 1984.

[Chambers, 96] J.B. Chambers, « Classical and flexible job shop scheduling by tabu search ». Thèse de Doctorat, Université du Texas, Austin, USA, 1996.

[Chang et al, 07] P.C. Chang, S.H. Chen et C.H. Liu, « Sub-population genetic algorithm with mining gene structures for multi-objective flow-shop scheduling problems ». Expert Systems with Applications, vol. 33, pp. 762–771, 2007.

[Chrétienne, 83] P. Chrétienne, « Les réseaux de pétri temporises ». Thèse de Doctorat, Université de Paris VI, Paris, 1983.

[Clerc et al, 04] M. Clerc et P. Siarry, « Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires ». J3eA, vol. 3, n° 7, pp. 1-13, 2004.

[Clerc et al, 09] M. Clerc et P. Siarry, « Une méthode inspirée de comportements coopératifs observés dans la nature : l'optimisation par essaim particulaire ». Revue de l'Electricité et de l'Electronique, REE, n°4, Avril 2009.

[Collette et al, 02] Y. Collette et P. Siarry, « Optimisation Multiobjectif ». Editions Eyrolles, Paris, 2002.

[Collins et al, 88] N.E. Collins, R.W. Eglese et B.L. Golden, « Simulated annealing: an annotated bibliography ». American Journal of Mathematical and Management Sciences, vol. 8, pp. 209-307, 1988.

[Dannenbring, 77] D. G. Dannenbring, « An evaluation of flow-shop sequencing heuristics ». Management Science, vol. 23, pp. 1174-1182, 1977.

[Davis, 91] L. Davis, « Handbook of genetic algorithm ». New York: Van Nostrand Reinhold, 1991.

[Dorigo et al, 97] M. Dorigo et L.M. Gambardella, « Ant colony system: a cooperative learning approach to the travelling salesman problem ». IEEE Transactions on Evolutionary Computation, vol.1, pp. 53-66, 1997.

[Durand et al, 94] N. Durand, J.M. Alliot et J. Noailles, « Algorithmes génétiques : un croisement pour les problèmes partiellement séparables ». Journées Evolution Artificielle Francophones, JEAF, Toulouse, 1994.

[Gangadhran et al, 94] R. Gangadharan et C. Rajendran, « A simulated annealing heuristic for scheduling in a flow-shop with bi-criteria ». 16th International Conference on Computers Industrial Engineering, pp. 345-348, 1994.

[Gao et al, 07], J. Gao, M. Gen, L. Sun et X. Zhao, « A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems ». Computers and Industrial Engineering, vol. 53, pp. 149–162, 2007.

[Gargouri, 03] E. Gargouri, « Ordonnancement Coopératif en Industrie Agroalimentaire ». Thèse de Doctorat, Université des Sciences et Technologies de Lille.

[Garey et al, 79] R. Garey et D.S. Johnson, « Computers and Intractability: A guide to the theory of P-completeness ». Editions Freeman and Co, 1979.

[Glover, 89] F. Glover, « Tabu search, part I ». ORSA, Journal of Computing, vol.1, pp. 190-206, 1989.

[Glover, 90] F. Glover, « Tabu search, part II ». ORSA, Journal of Computing, vol.2, pp. 4-32, 1990.

[Goldberg, 94] G.E. Goldberg, « Algorithmes génétiques ». Editions Addison Wesley, France, 1994.

[Gotha, 93] Gotha, « Les problèmes d'ordonnancement ». RAIRO-Recherche Opérationnelle, vol. 27, n° 1, pp. 77-150, 1993.

[Hao et al, 99] J.K. Hao, P. Galinier et M. Habib, « Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes ». Revue d'Intelligence Artificielle, vol.1999, pp. 2-39, 1999.

[Ho et al, 91] J.C. Ho et Y.L. Chang, « A new heuristic for the n-job, m-machine flow-shop problem ». European Journal of Operational Research, vol.52, pp. 194-202, 1991.

[Holland, 75] J.H., Holland, « Adaptation in natural and artificial systems ». PhD, Michigan Press Univ., Ann Arbor, MI, 1975.

[Ischibuchi, 94] H. Ischibuchi, « Genetic algorithms and neighbourhood search algorithms for fuzzy flow-shop scheduling problems ». Fuzzy Sets and Systems, vol. 67, pp. 81-100, 1994.

[Iyer et al, 04] S.K. Iyer et B. Saxena, « Improved genetic algorithm for the permutation flow-shop scheduling problem ». Computers and Operations Research, vol. 31, pp. 593–606, 2004.

[Jain et al, 99] A.S. Jain et S. Meeran, « Deterministic job-shop scheduling : past, present and future ». European Journal of Operational Research, vol. 113, pp. 390-434, 1999.

[Jo, 89] P. Jog, « The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem ». Third IGCA, San Mateo, 1989.

[Johnson, 54] S.M. Johnson, « Optimal two and three stage production schedules with setup times included ». Naval Research and Logistics Quarterly, vol. 1, 1954.

[Kacem, 03] I. Kacem, « Ordonnancement multicritère des job-shops flexibles : formulation, bornes inférieures et approche évolutionniste coopérative ». Thèse de Doctorat, Université des sciences et techniques de Lille1, 2003.

[Kennedy et al, 95] J. Kennedy et R.C. Eberhart, « Particle swarm optimization ». IEEE International Conference on Neural Networks, Piscataway, pp. 1942-1948, 1995.

[Kirkpatrick et al, 83] S. Kirkpatrick, C.D. Jr et M.P. Vecchi « Optimization by simulated annealing ». Science, vol.220, pp. 671-680, 1983.

[Knosala et al, 01] R. Knosala, T. Wal, « A production scheduling problem using genetic algorithm». Journal of Materials Processing Technology, vol. 109, pp. 90-95, 2001.

[Laquerbe et al, 98] C. Laquerbe, L. Pibouleau, P. Floquet et S. Domenech, « Procédures stochastiques en génie des procédés : Méthode du recuit simulé et algorithmes génétiques ». 6ème Colloque Maghrébin sur les Modèles Numériques de l'Ingénieur, C2MNI6, Tunis, pp. 761-766, 1998.

[Le Pape, 95] C. Le Pape, « Three mechanisms for managing resource constraints in the library for constraint-based scheduling ». INRIA/IEEE Conference on Emerging Technologies and Factory Automation, Paris, pp. 980-995, 1995.

[Lian et al, 06 a] Z. Lian, X. Gu et B. Jiao, « A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize Makespan ». Chaos, Solitons and Fractals, pp. 1-11, 2006.

[Lian et al, 06 b] Z. Lian, X. Gu et B. Jiao, « A similar particle swarm optimization algorithm for permutation flow-shop scheduling to minimize Makespan ». Applied Mathematics and Computation, vol. 175, pp. 773–785, 2006.

[Lian et al, 06 c] Z. Lian, B. Jiao et X. Gu, « A similar particle swarm optimization algorithm for job-shop scheduling to minimize Makespan ». Applied Mathematics and Computation, vol. 183, pp. 1008–1017, 2006.

[Mabed et al, 01] M. Mabed, M. Rahoual, E. Talbi et C. Dhaenens, « Algorithmes génétiques multicritères pour les problèmes de flow-shop ». 3e Conférence Francophone de Modélisation et SIMulation MOSIM'01 – du 25 au 27 avril 2001 – Troyes.

[Mellouli et al, 04] K. Mellouli, A. El Kamel, P. Borne, « Programmation linéaire et applications. Eléments de cours et exercices résolus ». Editions Technip, 2004.

[Mesghouni, 99] K. Mesghouni, « Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production ». Thèse de Doctorat, Université des Sciences et Technologies de Lille, 1999.

[Murata et al, 96] T. Murata, H. Ishibuchi et H. Tanaka, « Multi-objective genetic algorithm and its applications to flow-shop scheduling ». Computers Industrial Engineering, vol. 30, n° 4, pp. 957-968, 1996.

[Nawaz et al, 83] M. Nawaz, E.E. Enscore et I. Ham, « A heuristic algorithm for m-machine, n-job flow-shop sequencing problem ». OMEGA, vol. 11, pp. 91-98, 1983.

[Negenman, 01] E. G. Negenman, « Local search algorithms for the multiprocessor flow-shop scheduling problem ». European Journal of Operational Research, vol. 128, pp. 147-158, 2001.

[Neumann et al, 05] K. Neumann, C. Schwindt et N. Trautmann, « Scheduling of continuous and discontinuous material flows with intermediate storage restrictions ». European Journal of Operational Research, vol. 165, pp. 495–509, 2005.

[Nicholson, 71] T. Nicholson, « Optimization in industry ». Editions Longmann Press, Londres, 1971.

[Onwubolu et al, 04] G.C. Onwubolu et M. Clerc, « Optimal operational path for automated drilling operations by a new heuristic approach using particle swarm optimization ». International Journal of Production Research, vol. 42, n° 3, pp. 473-491, 2004.

[Onwubolu et al, 06] G. Onwubolu et D. Davendra, « Scheduling flow shops using differential evolution algorithm ». European Journal of Operational Research, vol. 171, pp. 674–692, 2006.

[Osman et al, 94] I.H. Osman et N. Christofides, « Capacitated clustering problems by hybrid simulated annealing and tabu search ». International Transactions in Operational Research, vol.1, pp. 317-336, 1994.

[Osman et al, 89] I.H. Osman et C.N. Potts, « Simulated annealing for permutation flow-shop scheduling ». OMEGA, vol. 17, pp. 551-557, 1989.

[Pan et al, 08] Q.K. Pan, M. F. Tasgetiren et Y. C. Liang, « A discrete particle swarm optimization algorithm for the no-wait flow-shop scheduling problem ». Computers and Operations Research, vol. 35, Issue 9, pp. 2807-2839, 2008.

[Pinedo, 55] M. Pinedo, « Scheduling : Theory, Algorithms and systems ». Prentice-Hall, Englewood Clis, New Jersey, 1955.

[Rodammer et al, 88] F.A. Rodammer et K. Preston White, « A recent survey of production scheduling ». IEEE Transaction on Systems, Man and Cybernetics, pp. 6-18, 1999.

[Ross, 97] P. Ross, « What are genetic algorithms good at? ». Informs Journal on Computing, vol. 9, n°3, 1997.

[Roy, 70] B. Roy, « Algèbre moderne et théorie des graphes, volume 2 ». Editions Dunod, Paris, 1970.

[Roy et al, 93] B. Roy et D. Bouysson, « Aide multi-critères à la décision : Méthodes et cas ». Collection Gestion Série : Production et technologie quantitatives appliquées à la gestion. Edition Economica, Paris, 1993.

[Ruiz et al, 06] R. Ruiz, C. Maroto et J. Alcaraz, « Two new robust genetic algorithms for the flow-shop scheduling problem ». Omega, vol. 34, pp. 461 – 476, 2006.

[Ruiz et al, 07] R. Ruiz, J. C. García-Díaz et C. Maroto, « Considering scheduling and preventive maintenance in the flow-shop sequencing problem ». Computers and Operations Research, vol. 34, pp. 3314-3330, 2007.

[Saad et al, 06] I. Saad et M. Benrejeb, « Optimisation multicritère par Pareto-optimale des problèmes d'ordonnancement en tenant compte du coût de production », Revue Sciences et Technologies de l'Automatique, e-STA, vol. 3, n°1, 2006.

[Saad, 07] I. Saad, « Conception d'un système d'aide à l'ordonnancement tenant compte des impératifs économiques ». Thèse de Doctorat, Ecole Centrale de Lille, Ecole Nationale d'Ingénieurs de Tunis, 2007.

[Saad et al, 07] I. Saad, H. Boukef et P. Borne, « The comparison of criteria aggregative approaches for the multi-objective optimization of flexible job-shop scheduling problems ». Fourth Conference on Management and Control of Production and Logistics, MCPL 2007, Sibiu, pp. 603-608, 2007.

[Sakarovitch, 84] M. Sakarovitch, « Graphes et Programmation Linéaire ». Edition Hermann, Paris, 1984.

[Sakarovitch, 84] M. Sakarovitch, « Programmation Discrète ». Edition Hermann, Paris, 1984.

[Sha et al, 06] D. Y. Sha et C. Y. Hsu, « A hybrid particle swarm optimization for job shop scheduling problem ». Computers and Industrial Engineering, vol. 51, pp. 791–808, 2006.

[Starkweather, 91] T. Starkweather « A comparison of genetic sequencing operators ». Fourth IGCA, pp. 69-76, 1991.

[Tangour et al, 06] F. Tangour, S. Hammadi, P. Borne et M. Benrejeb, « Ordonnancement dynamique dans un atelier de production agroalimentaire ». Séminaire d'Automatique-Industrie, SAI'06, Matmata, 2006.

[Tangour, 07] F. Tangour, « Ordonnancement dynamique dans les industries agroalimentaires ». Thèse de Doctorat, Ecole Centrale de Lille, Ecole Nationale d'Ingénieurs de Tunis, 2007.

[Tangour et al, 09] F. Tangour, I. Saad et P. Borne, « Optimisation par colonie de fourmis ». Revue de l'Electricité et de l'Electronique, REE, n°4, pp. 39-44, 2009.

[Van den Bergh et al, 00] F. Van den Bergh et A.P. Engelbecht, « Cooperative learning in neural networks using particle swarm optimizers ». South African Computer Journal, vol. 26, pp. 84-90, 2000.

[Xia et al, 05] W. Xia et Z. Wu, « An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems ». Computers and Industrial Engineering, vol. 48, pp. 409–425, 2005.

[Yang, 01] J.B. Yang, « GA-based discrete dynamic programming approach for scheduling in FMS environments ». IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 31, n° 5, pp. 824–835, 2001.

[Watanabe et al, 05] M. Watanabe, K. Ida et M. Gen, « A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem». Computers and Industrial Engineering, vol. 48, pp. 743–752, 2005.

[Widmer, 01] M. Widmer, « Les Métaheuristiques : Des outils performants pour les problèmes industriels ». 3ème Conférence Francophone de MOdélisation et SIMulation MOSIM'01, 25-27 avril 2001, Troyes.

[Widmer et al, 90] M. Widmer et A. Hertz, « A new heuristic method for the flow-shop sequencing problem ». European Journal of Operational Research, vol. 4, pp. 186-193, 1990.

[Wu et al, 05] Z. Wu et M.X. Weng, « Multiagent scheduling method with earliness and tardiness objectives in flexible job shops ». IEEE Transactions on System, Man, and Cybernetics-Part B, vol. 35, n° 2, pp. 293–301, 2005.

Annexe

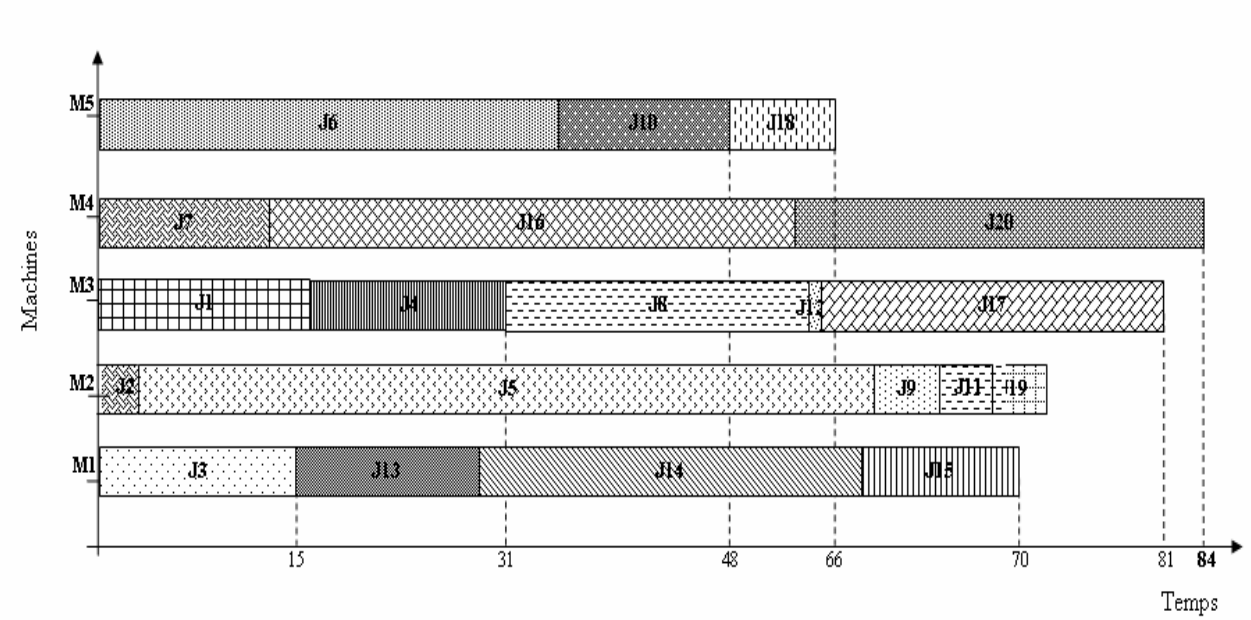

Diagramme de Gantt de la meilleure solution pour le problème FJSP 20x5 par utilisation des Algorithmes génétiques

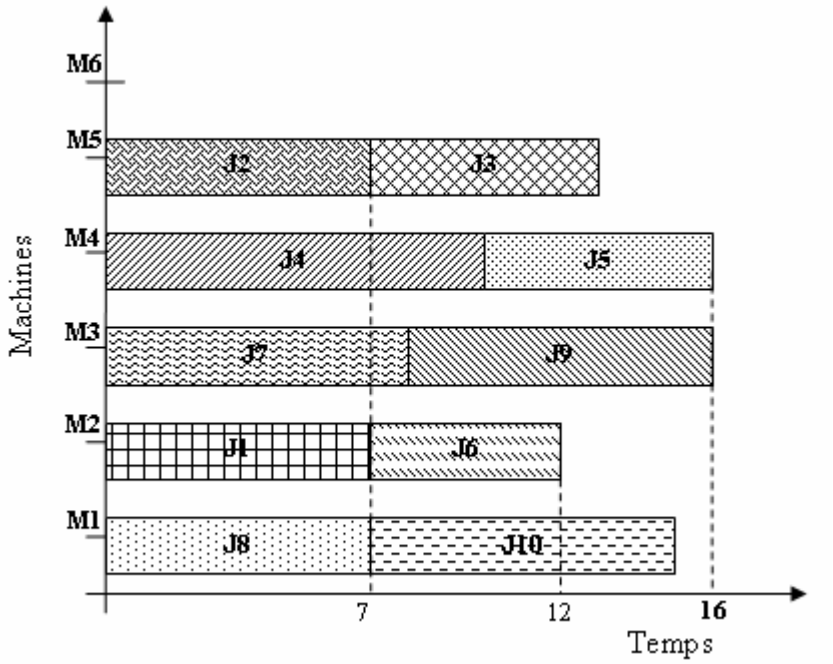

Diagramme de Gantt de la meilleure solution pour le problème FJSP 10x6 par utilisation des Algorithmes génétiques

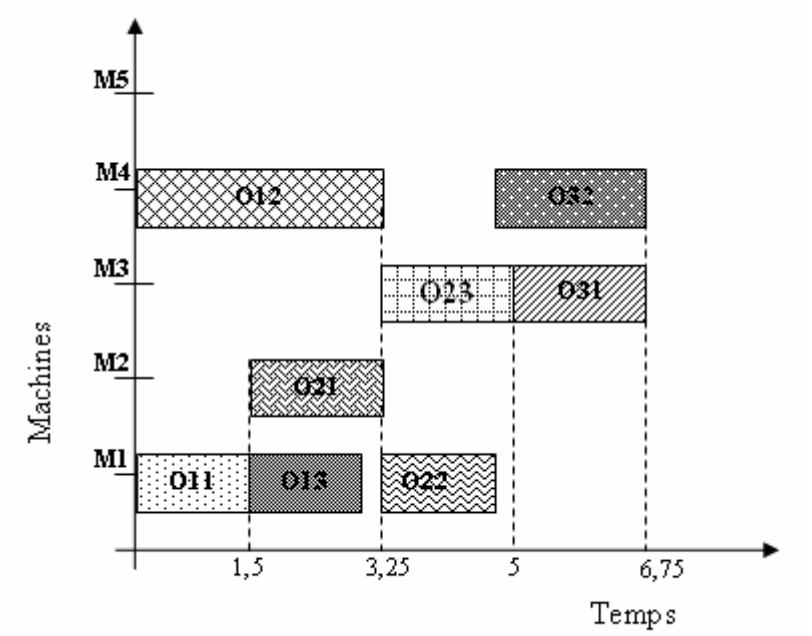

Diagramme de Gantt de la meilleure solution pour le problème FJSP 3x5 par utilisation des Algorithmes génétiques