N° Ordre ……../ Faculté/ UMBB/ 2013

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté des **S**ciences

Mémoire de Magister

Présenté par

ALIOUAT Wahiba

Filière : Systèmes Informatiques et Ingénierie des logiciels

Option : Spécification Logiciels et Traitement de l'Information

---

Le problème conjoint de l'ordonnancement de la production et de planification de la maintenance : cas du Flow Shop Flexible.

Devant le jury :

MEZGHICHE	Mohamed	Prof	UMBB	Président
BOUDHAR	Mourad	Prof	USTHB	Examinateur
HENTOUS	Hamid	MC/A	EMP	Examinateur
BERRICHI	Ali	MC/B	UMBB	Encadreur

Année Universitaire : 2013/ 2014

A la mémoire de mes parents & de mon ange sœur Assma

A mes très chers frères, Salah Eddine & Amani

A tous ceux qui m'aiment et ceux que j'aime.

Remerciements

Ce n'est pas par tradition que cette page figure au préambule de ce mémoire, mais c'est plutôt un devoir moral et une reconnaissance sincère qui me pousse à la faire. Je serais en effet ingrate si je n'exprime pas ma reconnaissance et ma gratitude à tous ceux, de prés ou de loin, qui ont facilité ma tâche et m'ont permis de mener à bien ce travail.

Je tiens à exprimer toute ma gratitude à Monsieur BERRICHI Ali, Maître de Conférences à l'université M'hamed Bougara de Boumerdès, qui a encadré ce mémoire. Il a su orienter aux bons moments mes travaux de recherches en me faisant découvrir l'ordonnancement. Je le remercie infiniment pour ses conseils, pour sa confiance, pour ses encouragements, pour ses commentaires précieux qui m'ont permis de surmonter les difficultés et de progresser, et pour ses corrections avisées de mes travaux et de ce mémoire.

Mes remerciements vont également à tous les membres de jury:
Monsieur MEZGHICHE Mohamed, de m'avoir accordé l'honneur de présider mon jury de soutenance, Monsieur BOUDHAR Mourad et Monsieur HENTOUS Hamid, d'avoir accepté la tâche dévaluer, en qualité d'examinateurs, mon travail.
Je ne saurais oublier de remercier Mme Farida ZERARI, la coordinatrice de la Bibliothèque Universitaire pour ses encouragements durant toute la période de ce mémoire. Merci à Z. Nadia et l'équipe informatique de la BU pour toute la confiance que vous avez porté en moi. Merci tout simplement pour votre sincère amitié et vos précieuses qualités humaines.

Il me tient tout particulièrement à cœur de remercier ma grande mère pour m'avoir soutenu dans cette aventure et pour toute la confiance qu'elle m'a apportée. Je n'oublierai pas mon frère, ma sœur qu'ils m'excusent pour tout le temps que j'ai consacré à mon mémoire et que je leurs ai volé, j'espère qu''ils sont fiers de moi, ainsi que toute ma famille pour leurs encouragements. Mille Merci.
Enfin, je tiens à exprimer ma reconnaissance et mon amour à tous mes ami(e)s, mes collègues et à tous ceux qui de près ou de loin m'ont encouragé, m'ont supporté et soutenu dans des moments difficiles. Ils sont nombreux et tous présents dans mon cœur.

Je garderai toujours un vif souvenir de cette merveilleuse aventure qui m'a beaucoup apporté et m'a surtout tant appris sur moi-même. J'ai essayé d'en vivre pleinement et d'en apprécier chaque moment même les plus durs.

RESUME

Les fonctions d'ordonnancement de la production et de planification de la maintenance représentent deux enjeux majeurs dans l'industrie manufacturière. Le problème de l'ordonnancement est de trouver un programme d'exécution approprié des tâches de production sur des machines, réalisant au mieux un ou plusieurs objectifs généralement liés aux temps de fin des tâches. De nombreuses études sont dédiées à résoudre ces problèmes selon les différents types d'ateliers (une seule machine, des machines parallèles, flow shop, job shop et open shop ainsi que des systèmes hybrides). La plupart des problèmes d'ordonnancement sont des problèmes NP-difficiles. La plupart des travaux de recherche supposent que les machines sont toujours disponibles durant l'horizon de planification. Cependant, dans les systèmes industriels réels, cette hypothèse n'est pas réaliste car les machines doivent être arrêtées pour des actions de maintenance ou pour d'autres considérations. Malgré la relation d'interdépendance entre les deux fonctions, ces deux activités sont généralement planifiées et exécutées séparément. Ces dernières années, les décideurs donnent une plus grande importance à la fonction maintenance vu sa contribution substantielle à la productivité. Pour éviter des conflits entre ces deux activités, une collaboration et une flexibilité doivent être envisagées entre les deux services. Récemment, un certains nombre d'études et de travaux ont été établis traitant les problèmes d'ordonnancement et de maintenance conjointement. L'objectif de ce mémoire est l'étude de ce problème dans le cas d'un atelier de type flow shop flexible. Plusieurs industries telles que l'industrie des cosmétiques, pharmaceutiques, agroalimentaire et textile ainsi que des systèmes informatiques peuvent être modélisés avec un Flow Shop Flexible (Flow Shop Multiprocesseur). Dans ce travail, l'optimisation considérée est bi-objectif de type Pareto afin de trouver un compromis pour le service d'ordonnancement et le service de maintenance. Des modèles de fiabilité ont été introduits pour l'optimisation de l'aspect maintenance. Deux méthodes de résolution ont été proposées: les algorithmes génétiques et les systèmes immunitaires artificiels. Nous avons aussi proposé d'intégrer la logique floue pour améliorer le paramétrage de l'algorithme immunitaire développé. Dans ce mémoire, nous avons étudié les différentes approches et comparé leurs résultats.

Mots clés : ordonnancement, maintenance, optimisation multi-objectif, méta-heuristiques, logique floue.

ABSTRACT

Production scheduling and maintenance planning are two major challenges in manufacturing industry. The scheduling problem is to find an appropriate execution program of production tasks on machines, meeting at best one or more objectives generally related to tasks completion times. Many studies are dedicated to solving these problems considering different types of workshops (one machine, parallel machine, flow shop, job shop and open shop as well as hybrid systems). Most of scheduling problems are NP-hard. Most research works assume that machines are always available during the planning horizon. However, in real industrial systems, this assumption is not realistic because the machines must be stopped for maintenance tasks or for other considerations. Despite the interdependent relationship between the two functions, these two activities are usually planned and executed separately. In recent years, decision makers give greater importance to maintenance function regarding its substantial contribution to productivity. To avoid conflicts between these two activities, collaboration and flexibility should be considered between the two services. Recently, a number of studies and works have been established dealing with production scheduling and maintenance jointly. The objective of this thesis is the study this problem in flexible flow shop environment. Many industries such as cosmetics industry, pharmaceutical, agrifood and textile and computer systems can be modeled as flexible flow shops (multiprocessor flow shop). In this work, optimization considered is bi-objectives of Pareto type in order to find compromise solutions between the production objectives and maintenance ones. Reliability models are used to take into account the maintenance aspect of the problem. Two methods of resolution were proposed: genetic algorithms and artificial immune systems. We also proposed to integrate fuzzy logic to improve the parameter setting of the developed immune algorithm. In this thesis, we studied the different approaches and compared their results.

Keywords: scheduling, maintenance, reliability, multi-objective optimization, meta-heuristics, fuzzy logic.

ملخص

إن ترتيب الأشغال في عمليةِ الإنتاج و برمجة الصيانة يعتبران من أكبر التحديات الصناعية. مسألة برمجة الإنتاج تكمن في إيجاد برنامج مناسب لانجاز الأشغال بواسطة الآلات الصناعية ، لتحقيق على أحسن وجه هدف أو عدة أهداف متعلقة عموما بزمن نهاية الأشغال. عدة دراسات أجريت لحل هذا النوع من المسائل لمختلف نماذج الو رشات الصناعية (آلة واحدة، آلات متوازيةِ، flow shop ، job shop ,open shop, الأنظمة الهجينة). لقد برهن في المراجع العلمية أن أغلب هذه المسائل جد صعبة. معظم الأبحاث العلمية تفترض تواجد الآلات خلال زمن انجاز الأشغال دائما, غير أنه في واقع الأنظمة الصناعية، هذه الافتراضية غير صحيحة دائما لأن الآلات يمكن أن تتوقف لأجل الصيانة أو اعتبارات أخرى. على الرغم من العلاقة المتبادلة بين العمليتين، هاتين المهمتين تبرمجان و تنجزان منفصلتين. في السنوات الأخيرة، أرباب العمل أولو أهمية كبيرة لعملية الصيانة نظرا لمساهمتها الجوهرية في الإنتاجية. لتفادي أي خلاف أو تعارض بين المهمتين يتوجب التعاون و المرونة بينهما. حديثا، عدد من الدراسات والأعمال قد أنجزت تعالج مسألة برمجة عمليتي الإنتاج و الصيانة معا و في آن واحد. الهدف من هذه المذكرة هو حل مثل هذه المسائل في حالة flow shop flexible. العديد من الصناعات مثل صناعة مستحضرات التجميل، صناعة الأدوية، صناعة المواد الغذائية، النسيج، أنظمة الإعلام الآلي يمكن تمثيلها ب flow shop flexible. طريقة الحل المتبعة ثنائية الهدف من نوع Pareto من أجل إيجاد حلول وسط (تسوية) بين أهداف الإنتاج و أهداف الصيانة. اعتمدنا على نموذج الفاعلية modèle de fiabilité لأخذ بعين الاعتبا ر الصيانة في المسألة. اقترحنا طريقتين للحل : الخوارزميات الجينية و أنظمة المناعة الصناعية. بالإضافة إلى اقتراح دمج la logique floue لتحسين ضبط خوارزمية المناعة المقترحة. درسنا في هذه المذكرة جميع الطرق المقترحة و عرضنا نتائج المقارنة بينها.

كلمات البحث :ترتيب الأشغال، صيانة، حل متعدد الأهداف، فاعلية، خوارزميات تقريبية

Table des matières

Chapitre 2. L'ORDONNANCEMENT DANS LES ATELIERS DE PRODUCTION : CAS DU FLOW SHOP HYBRIDE

Chapitre3. L'ORDONNACEMENT CONJOINT PRODUCTION/MAINTENANCE

Liste des tables

Liste des figures

INTRODUCTION GENERALE

Dans le domaine de la production industrielle, les tendances actuelles indiquent que les systèmes manufacturiers performants doivent s'adapter rapidement aux fluctuations du marché (demandes aléatoires) et aux perturbations internes (pannes des machines). Les machines doivent pouvoir fabriquer plusieurs types de produits simultanément et de manière efficace [KAA04].

Dans un tel contexte, rester toujours performant, passe obligatoirement par le maintien en état de fonctionnement de l'outil de production, qui reste toujours la préoccupation majeure de tous les gestionnaires dans un monde industriel où les notions de réactivité, de coûts et de qualité ont de plus en plus d'importance, et où il est vital de pouvoir s'appuyer sur un système de production performant à tout instant [BEN05]. Ainsi, la détermination simultanée d'un meilleur programme de réalisation des commandes et d'une politique de maintenance appropriée pour le système de production est devenue un problème préoccupant dans le domaine de l'optimisation des systèmes manufacturiers [BER09]. Nous sommes donc face à un problème bi-objectif, d'une part ordonnancer la production sous les contraintes de respect des délais, coût et qualité des produits et d'autre part planifier la maintenance sous les contraintes de sureté de fonctionnement des équipements qui assurent la pérennité de l'outil de production [BEN05].

Les activités de production et de maintenance apparaissent à priori comme totalement antagoniste et par la même incompatibles puisqu'elles agissent sur les mêmes ressources. Ces deux activités ne peuvent donc pas être accomplies au même moment sur un même système, ce qui laisse augurer bien des conflits dans l'utilisation du système par l'un ou l'autre des services liés à la maintenance ou à la production. Ce type de conflit entraîne naturellement des querelles qui nuisent à la productivité globale de l'entreprise. Malgré l'interdépendance qui existe entre les deux services, ces deux activités sont généralement planifiées séparément, leur intégration dans le fonctionnement de l'atelier pose un problème qui est souvent résolu par négociation entre les responsables respectifs des deux services et de manière séquentielle. Les besoins de la production sont donc quelque peu antagonistes. Pour satisfaire dans les délais les clients, il doit utiliser de façon optimale l'ensemble des installations, mais il doit également prévoir les interventions de maintenance dans le planning de production [BEN05].

Notre travail s'inscrit dans ce souci d'optimiser l'ordonnancement de production et de maintenance en anticipant tout conflit pouvant se présenter entre ces deux services et ceci peut être réalisé en créant une certaine coopération entre les deux services production et maintenance. Dans le cadre de l'étude théorique de ce mémoire, nous nous focalisons sur l'ordonnancement conjoint de la production et de la maintenance dans le cas d'atelier de type Flow Shop Hybride avec machines en parallèle identiques à chaque étage et nous nous intéressons à la maintenance préventive systématique.

Le but de ce travail est de proposer une modélisation et une approche de résolution bi-objective intégrée au problème. En effet, les services de production et de maintenance doivent collaborer pour atteindre un but commun, celui de la maximisation de la productivité du système de production. Par conséquent, les objectifs de production et de maintenance doivent être considérés avec le même niveau d'importance. Donc, les solutions au problème conjoint production-maintenance doivent être des solutions de compromis entre les objectifs des deux services. En outre, le but final des décideurs est d'avoir un système de production qui soit le plus disponible possible lui permettant de réaliser les tâches de production le plus tôt possible. Des modèles de fiabilité ont été utilisés pour prendre en considération l'aspect maintenance [BER09].

Ce mémoire est organisé en cinq chapitres comme suit :

Le premier chapitre sera présenté en deux parties. La première partie abordera succinctement les systèmes de production à travers la définition de ce qu'est le système de production manufacturier au sein de l'entreprise. La deuxième partie sera consacrée à la maintenance des systèmes de production, où nous y présenterons les concepts de la maintenance liés à la sûreté de fonctionnement et nous détaillerons le concept de fiabilité et de disponibilité.

Le deuxième chapitre se présentera aussi en deux parties: la première relative aux problèmes d'ordonnancement de production, leur typologie et leur difficulté. Pour délimiter les contours de notre étude, la deuxième partie sera consacrée à l'étude du Flow Shop Hybride en présentant de manière détaillée ce problème.

Le troisième chapitre s'intéresse à la nécessité de la mise en œuvre d'une relation de coopération entre les services de production et de maintenance. Nous rappellerons en premier lieu les différentes politiques d'ordonnancement conjoint de la production et de la maintenance recensées de la littérature. Un état de l'art sur ces travaux, dans le cas déterministe et stochastique sera présenté. Par la suite, nous présenterons le modèle intégré pour la solution conjointe du problème d'ordonnancement de la production et de la maintenance dans le cas du Flow Shop Hybride. Nous terminerons par des notions de base et les différentes méthodes d'optimisation multi objectifs.

Dans le quatrième chapitre, une résolution du problème à l'aide des algorithmes génétiques sera présentée. Une description complète du fonctionnement des algorithmes génétiques sera faite. Nous présenterons une mise en œuvre de deux algorithmes génétiques multi objectif élitistes pour l'ordonnancement conjoint production et maintenance dans un Flow Shop Hybride, nous terminons par une série de tests pour évaluer la méthode de résolution proposée.

Le cinquième chapitre présentera des algorithmes basés sur l'approche immunitaire, suivi d'une explication du mécanisme du système immunitaire des vertébrés qui a été la source d'inspiration pour nos algorithmes. A la fin nous discuterons les résultats obtenus par l'application de l'approche immunitaire pour la résolution de notre modèle en effectuant des comparaisons avec les algorithmes génétiques multi objectifs développés au chapitre précédent. Pour améliorer les résultats de l'approche immunitaire, nous proposerons un contrôleur flou pour le réglage dynamique des paramètres de l'algorithme immunitaire proposé.

Le rapport se termine par des conclusions ainsi que des perspectives de recherche envisagées.

Chapitre 1

LA MAINTENANCE DES SYSTEMES DE PRODUCTION

Introduction

La principale conséquence du développement industriel est la complexité croissante des machines et des équipements de production. Ainsi, pour satisfaire une demande de produits avec une meilleure qualité et à des prix compétitifs tout en respectant les délais de livraison, le développement des ateliers manufacturiers doit intégrer à la fois automatisation et flexibilité. D'où l'augmentation du risque d'occurrence des pannes qui se traduit par un temps croissant de détection et de réparation des machines. Cela aurait, dans ce cas, pour effet la diminution de la disponibilité du système global. Les entreprises sont de plus en plus sensibilisées à l'importance des coûts induits par les défaillances accidentelles des systèmes de production [BEN05]. Pour maintenir les coûts de production bas tout en ayant des produits de qualité, des actions de maintenance sont souvent effectuées sur ces systèmes [BER09].

Alors que la maintenance, jusqu'à très récemment, était considérée comme un centre de coûts, nous sommes de plus en plus conscients qu'elle peut contribuer d'une manière significative à la performance globale de l'entreprise. La complexité des mécanismes de dégradation des équipements a fait en sorte que la durée de vie de ces derniers a toujours été traitée comme une variable aléatoire. Cet état de fait a incité plusieurs entreprises à adopter des approches plutôt réactives, n'étant pas en mesure de justifier économiquement les avantages que peut procurer la mise en place d'une maintenance préventive [BEN05].

Ce chapitre est présenté en deux parties. Dans la première partie, on abordera succinctement **les systèmes de production,** quand à la deuxième partie elle sera consacrée à **la maintenance des systèmes de production**.

1.1 LES SYSTEMES DE PRODUCTION

1.1.1 Généralités sur les systèmes de production

1.1.1.1 Production

Giard [GIA88] définit la production comme étant une transformation de ressources appartenant à un système productif et conduisant à la création de biens et de services.

Les ressources peuvent être de quatre types :

– des équipements (machines, ...),
– des hommes (opérateurs, ...),
– des matières (matières premières et composants),
– des informations techniques ou procédurales (gammes, fiches opératoires, ...).

1.1.1.2 Un processus de production

IL est généralement composé d'un grand nombre d'opérations ou de transformations organisées en réseau. Ces opérations assurent des transformations de forme (modification des produits eux même), des transformations dans le temps (fonction de stockage) ou dans l'espace (fonction de transport).

1.1.1.3 Un système de production

Un système de production représente l'ensemble du processus grâce auquel l'entreprise transforme un ensemble de matières premières ou de produits semi finis en produits finis à l'aide de moyens de production [BER09].

1.1.2 Classification des systèmes de production

1.1.2.1 Classification selon les processus de production

De manière générale, nous pouvons classer les systèmes de production en trois grandes classes [AMO99] :

– **Les processus continus** tels que la production électrique, la chimie ou la papeterie ;

– **Les processus discrets** tels que l'usinage et toutes les activités d'assemblage, etc.

– **Les processus discontinus** qui se situent par définition à mi-chemin entre les processus continus et les processus discrets. Les deux types de processus sont couplés : la production est continue mais il y a un conditionnement discret des produits.

La figure 1.1 décrit cette classification et précise la classe de l'industrie manufacturière, domaine de nos recherches.

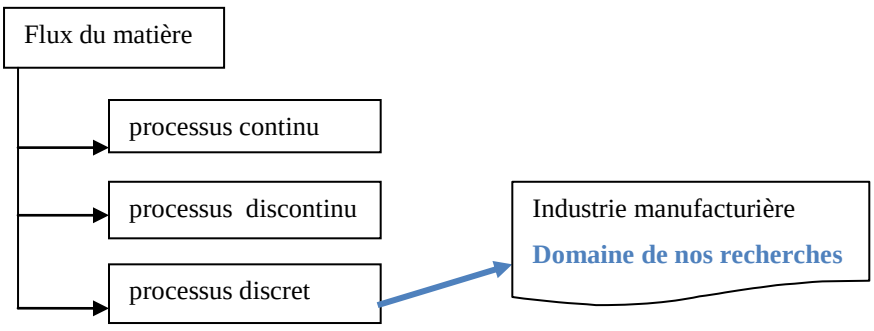

Figure 1.1. Classification des systèmes de production

Face à la diversité des processus de production, une classification s'impose. Giard [GIA88] propose deux typologies : une première liée à l'origine de la demande et une seconde liée aux ressources.

1.1.2.2Classification selon le mode de déclenchement de la production

Les ordres de fabrication peuvent émaner, soit pour satisfaire des commandes sur mesure, soit pour mettre à niveau les stocks des produits finis. Pour cette catégorie, les systèmes de production peuvent être regroupés en deux classes [BER09]:

– ***Systèmes de production à la commande :*** la production est déclenchée soit par les commandes fermes des clients soit par des demandes aléatoires. Autrement dit, la fonction de production est pilotée par l'aval. L'objectif est de satisfaire les délais de livraison négociés avec le client. L'avantage de ces systèmes est l'inexistence de produits en stock.

– ***Systèmes basés sur la production pour stock :*** la fonction de production est pilotée par l'amont, c'est-à-dire que le processus de production répond à un cahier des charges prédéfini ce qui peut se traduire par la constitution de stocks de produits finis. Contrairement à la production sur commande, les délais de livraison sont nuls mais il y a des stocks à gérer.

1.1.2.3 Classification selon l'organisation des ressources

Cette typologie dépend de la manière dont les ressources sont organisées pour traiter les matières premières ou les produits semi finis. Elle est divisée en quatre types de systèmes de production [BER09] :

– ***Système à production unitaire:*** ce type de système concerne les grands projets (uniques ou réalisés à de petites séries) nécessitant des périodes de temps assez longues et des moyens relativement importants. La réalisation du produit se fait sur mesure en fonction de la demande du client (bâtiment, construction navale, etc.). Pour ce type d'organisation, l'objectif est de réaliser le projet dans un délai optimal. Pour cela, la tâche principale de l'entreprise est de planifier les différentes opérations composant le projet en respectant des contraintes temporelles et de succession entre les opérations. Les méthodes de modélisation les plus répandues sont le diagramme de Gantt, les méthodes des potentiels [HAR95] et la méthode PERT [BAV02].

– ***Système de production en petite ou moyenne série ou atelier :*** ce type de production est rencontré dans l'industrie manufacturière où il faut fabriquer une grande variété de produits en faible quantité en utilisant les mêmes moyens de production. Les ateliers sont classés selon l'ordre de passage sur les machines des différents produits à fabriquer (gamme de fabrication). On distingue les ateliers à cheminement unique (*flowshop*) où toutes les gammes de fabrication sont identiques, les ateliers à cheminements multiples où chaque produit ou famille de produits est réalisé selon une gamme spécifique (*jobshop*) et les ateliers à cheminements multiples et libres (*openshop*) où le produit ne possède pas une gamme spécifique.

– ***Système de production en grande série ou masse :*** ce type de production est adopté quand il s'agit de fabriquer des produits standards à grande consommation (moteurs, composants industriels, etc.). Ces systèmes sont organisés en ligne de fabrication où les produits passent par une même séquence de postes de travail.

– ***Système de production continue ou processus :*** dans ce système de production peu de produits sont fabriqués mais en grande quantité. La production est déclenchée par l'amont tel que l'industrie des boissons ou de l'acier. Dans ce type d'organisation le niveau de stock d'en-cours est quasiment nul.

1.1.3 Modèle conceptuel d'un système de production

Il est possible de décomposer les systèmes de production en trois sous systèmes [BER09]. : **Le système physique de production**, **le système d'information** et **le système de décision** [ROB88] (Figure1.2). Le système couramment appelé **système de gestion de production** est constitué par la partie du système de décision et du système d'information traitant des fonctions rattachées directement à la production (par exemple, les achats, les approvisionnements, la planification, la gestion des ressources, la maintenance, etc.).

En termes de système, le **système physique** transforme les matières premières en produits finis. Pour effectuer cette transformation, il est commandé par le **système de gestion** qui transforme les informations à caractère commercial en ordres de fabrication et ordres d'approvisionnements. Le système est bouclé puisqu'en retour, il reçoit les informations de suivi du système physique pour pouvoir effectivement piloter ce dernier.

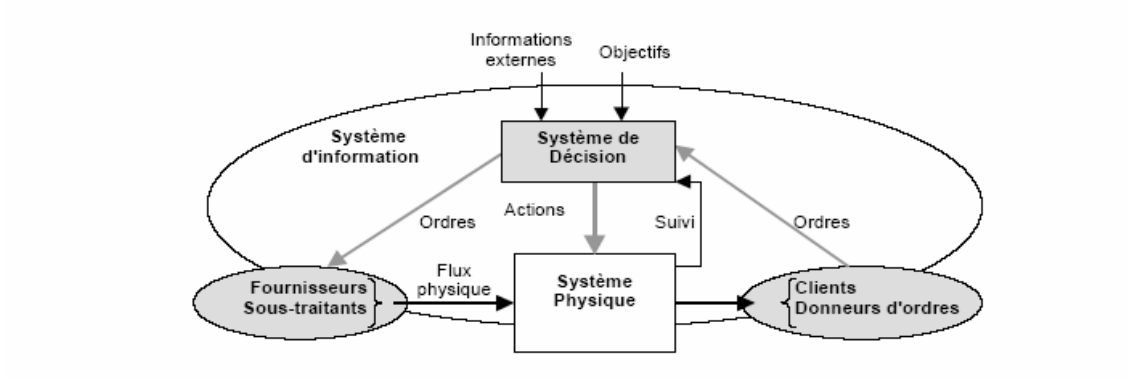

Figure 1.2.Modèle conceptuel d'un système de production [BER09].

En termes de flux, un ensemble de flux régulés parcourent le système de production. Tout d'abord, le **flux physique** ou **de matière** qui transforme la matière première et les composants en produits finis, puis le **flux d'information** ou **de suivi** qui permet la circulation des informations nécessaires au contrôle et à la prise de décision . Enfin, le **flux de décision** ou

ordre qui contrôle et pilote le système physique. Le système de gestion est composé de différentes activités telles que [BEN05] :

– L'élaboration du Plan Directeur de Production (PDP);

– Le calcul des besoins bruts, nets et d'approvisionnements ;

– La gestion des stocks ;

– Les achats ;

– L'élaboration du plan de charge ;

– **L'ordonnancement** ;

– Et enfin le lancement.

1.1.4 Organisation hiérarchique d'un système de gestion de production

En gestion de production, pendant très longtemps, l'objectif principal a été la recherche d'une plus grande efficacité de l'outil de production. Elle était, à l'image de l'organisation de l'entreprise toute entière, structurée par fonctions ou services, spécialisés et liés hiérarchiquement. En fait, cette hiérarchie obéit à un modèle de résolution de problème, décomposant les décisions en trois niveaux [GIA03]:

– **Les décisions stratégiques** se traduisent par la formulation de la politique à long terme de l'entreprise (vision à plus de deux ans, en général), ce qui implique une définition volontaire et cohérente du portefeuille d'activités et d'investissements.

– **Les décisions tactiques** correspondent à un ensemble de décisions à moyen terme parmi lesquelles on trouve: **la planification de la production** qui est une programmation prévisionnelle de la production établie sur un horizon variant généralement entre 6 et 18 mois; **la préparation du plan de transport** qui correspond à un ensemble de tournées-types de distribution ou d'approvisionnement qui seront utilisées dans les entreprises. Ces décisions s'inscrivent dans un cadre logique dessiné par les décisions stratégiques, mais l'horizon de planification est normalement trop court.

– **Les décisions opérationnelles** assurent la flexibilité quotidienne nécessaire pour faire face aux fluctuations prévues de la demande et des disponibilités de ressources (mode prévisionnel) et réagir aux aléas (mode correctif) dans le respect des décisions tactiques. Parmi ces décisions opérationnelles on cite: **la gestion des stocks,** qui assure la mise à disposition des matières premières et des composants; **l'ordonnancement,** qui consiste en une programmation prévisionnelle détaillée des ressources mobilisées (opérateurs, équipements, et outillages) dans

l'exécution des opérations nécessaires à la production élémentaire de biens ou de présentation de service, sur un horizon ne dépassant pas quelques dizaines d'heures.
La Figure 1.3 décrit cette hiérarchie et montre l'interaction qui existe entre les activités du système de gestion.

Nos travaux se situent au niveau opérationnel pour transmettre des ordonnancements prévisionnels au niveau pilotage temps réel.

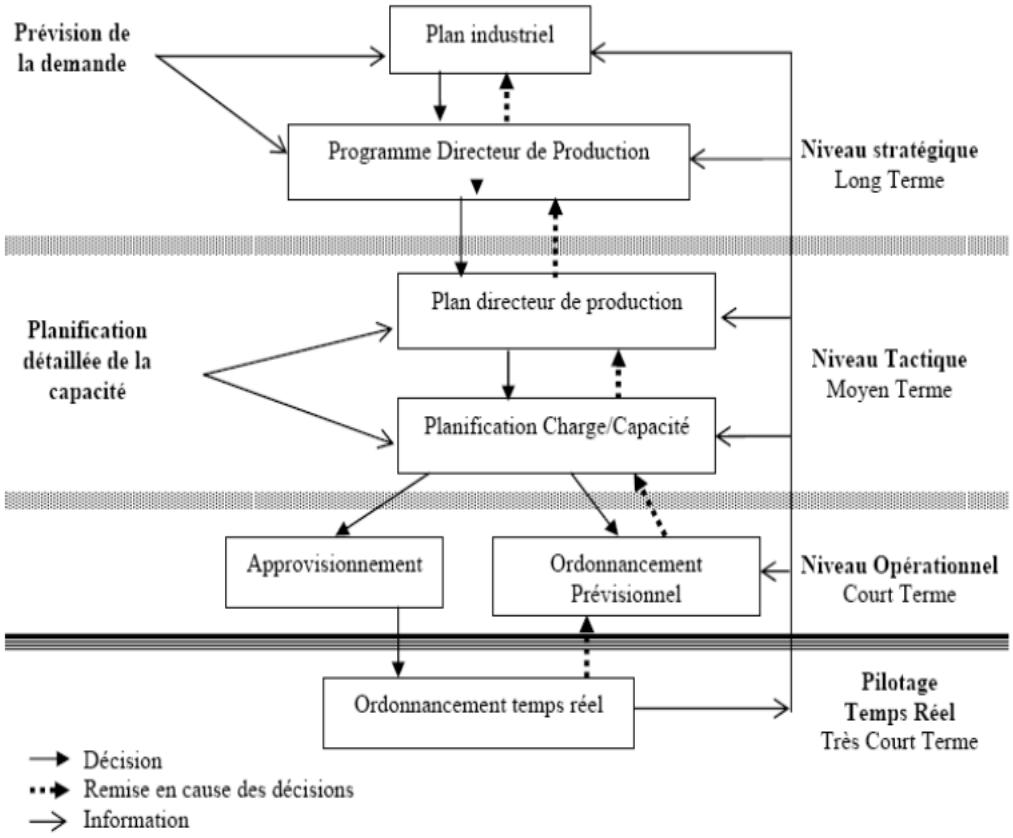

Figure 1.3. Organisation hiérarchique d'un système de gestion de production [BEN05].

1.1.5 Organisation de la production

L'organisation de la production est passée par plusieurs étapes en fonction des tendances de marché qui caractérisaient l'environnement économique. Les nouvelles formes d'organisation du travail inspirées du Toyotisme (développé par les responsables de production de la firme japonaise Toyota en 1950) s'inscrivent dans une logique de qualité totale qui consiste à développer un processus continu d'amélioration du processus de production à partir de la mobilisation de l'ensemble des personnels impliqués que ce soit au niveau des méthodes de

gestion de la qualité que de l'assurance-qualité [GM05]. Cette recherche de qualité totale est symbolisée par le principe des « 5 zéros » [BER09]:

Zéro défaut : le processus de fabrication doit limiter au maximum les défauts de fabrication de manière à éviter le gaspillage des ressources utilisées pour produire. Le contrôle de la qualité du processus productif ne se limite donc plus à un contrôle à la fin du processus productif mais est intégré dans l'ensemble de la chaîne de production.

Zéro panne : la recherche d'une efficacité optimale du processus productif (afin de garantir des gains de productivité) se traduit par la mise en place d'un système de maintenance préventif qui vise à éviter les pannes plutôt qu'à intervenir une fois celles-ci constatées (ce qui entraîne alors un arrêt temporaire de la production).

Zéro délai : les gains de productivité vont par ailleurs être obtenus par la mise en place d'un système de production en continu ce qui dans le cadre d'un mode de production flexible se traduit par l'élaboration de processus de production facilement reprogrammables et adaptables.

Zéro stocks : des gains de productivité peuvent aussi être obtenus par la suppression des stocks de produits finis ou de produits intermédiaires qui coûtent chers à l'entreprise. La production va donc être organisée selon le principe de la production au « juste-à-temps ».

Zéro papier : la flexibilité de l'outil de production ne pourra être obtenu que par une organisation plus souple et donc moins dépendante de procédures administratives complexes qui ralentissent le processus de décision. Le système d'information devient donc moins formel et moins vertical.

1.1.6 Critères d'évaluation d'un système de production

La conception et l'exploitation des systèmes de production nécessitent des techniques d'évaluation basées sur deux principaux critères: la performance et la sûreté de fonctionnement [BEN05]:

1.1.6.1 La performance

La performance d'un système peut être définie comme étant l'efficacité à fournir un service attendu à un instant donné et dans des conditions prédéterminées. Elle intègre les notions de coût, délais, qualité, flexibilité (capacité de changement rapide aux modifications de l'outil de production) et valeur (la satisfaction du client). Une façon de la mesurer consiste à mesurer la distance entre ce que l'entreprise obtient avec les moyens dont elle dispose et ce qu'elle souhaite atteindre comme objectif.

1.1.6.2 La sûreté de fonctionnement

La sûreté de fonctionnement (SdF) des systèmes de production automatisés est une nécessité économique. Elle consiste à assurer le respect du cahier des charges en terme de productivité, en tenant compte des perturbations (défaillances, aléas, etc.) infectant un atelier en assurant une qualité et une disponibilité maximale. Optimiser la commande devient illusoire si l'outil de production tombe souvent en panne.

Elle intègre les notions de disponibilité, fiabilité, maintenabilité et sécurité. Elle consiste à connaître, détecter, évaluer, prévoir, mesurer et maîtriser les défaillances des machines. L'intégration d'un service de maintenance est indispensable pour l'amélioration de la SdF et l'augmentation des performances des systèmes de production.

1.2 LA MAINTENANCE DES SYSTEMES DE PRODUCTION

1.2.1 Définition de la maintenance

Une première définition normative de la maintenance fut donnée par l'AFNOR en 1994 (*norme NF X 60-010),* à savoir « l'ensemble des actions permettant de maintenir ou de rétablir un bien dans un état spécifié ou en mesure d'assurer un service déterminé ». AFNOR se fait plus précise en apportant un complément avec le document *X 60-000* « Bien maintenir, c'est assurer ces opérations au coût optimal ». Depuis 2001, elle a été remplacée par une nouvelle définition, désormais européenne *(NF EN 13306 X 60-319)* : « Ensemble de toutes les actions techniques, administratives et de management durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise. »

1.2.2 Objectifs de la maintenance

Selon la norme AFNOR *FD X 60-000,* les objectifs de la maintenance sont :

- la disponibilité et la durée de vie du bien ;
- la sécurité des hommes et des biens ;
- la qualité des produits ;
- la protection de l'environnement ;
- l'optimisation des coûts de maintenance ;
- etc.

Les missions principales de la fonction maintenance [BER09] sont de :

– accroître la **fiabilité** du système, c'est-à-dire « l'aptitude du système à accomplir dans des conditions données et pendant un temps donnée, une fonction requise ».

– assurer la **disponibilité** du système, c'est-à-dire « son aptitude à être en état d'accomplir sa fonction ».

– accroître la **maintenabilité** du système**,** c'est-à-dire « l'aptitude du bien à être rétabli ou maintenu dans un état dans lequel il peut accomplir la fonction requise ».

Les concepts de fiabilité, disponibilité et maintenabilité sont des mots clés d'une nouvelle discipline qu'est la sûreté de fonctionnement.

1.2.3 Concepts de sûreté de fonctionnement

On s'intéresse aux concepts de fiabilité et de disponibilité :

1.2.3.1 Fiabilité (*Reliability*)

La norme *NF X 60-500* définit la fiabilité comme « l'aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné ».

L'entité (*E*) désigne au sens large un composant, sous-système ou système, et la fonction requise est la ou les fonctions que doit accomplir le dispositif pour pleinement remplir la tâche qui lui est assignée.

Considérons l'instant *T* d'occurrence de la défaillance; cette variable aléatoire permet de définir la notion de fiabilité qui s'interprète comme la probabilité que l'entité considérée ne tombe pas en panne avant un instant *t* donné ou bien comme la probabilité qu'elle tombe en panne après l'instant *t*.

Par extension, on appelle également fiabilité la probabilité associée ***R(t)*** à cette notion alors qu'elle n'en est qu'une mesure. Elle est définie par :

R(t) = *P* (*E* non défaillante sur la durée [0, *t*], en supposant qu'elle n'est pas défaillante à l'instant *t* = 0).

Ce qui peut s'exprimer par : $\boldsymbol{R(t) = P\ (T > t)}$

– **Fonction de répartition**

Nous distinguons par ***F(t)*** la fonction de répartition de *T* ou probabilité de défaillance. C'est l'aptitude contraire de la fiabilité : $\boldsymbol{F\ (t) = 1 - R\ (t) = P\ (t < T)}$

– **Densité de défaillance**

La densité de probabilité de l'instant de la défaillance *T* s'obtient en dérivant la fonction de répartition *F (t)*: $\boldsymbol{f(t) = \frac{dF(t)}{dt} = -\frac{dR(t)}{dt}}$

Ce terme est appelé la densité de défaillance.

– **Moyenne de temps de vie avant la première défaillance (*MTTF*)**

Une grandeur moyenne associée à la fiabilité souvent utilisée est le temps moyen de fonctionnement d'une entité ou moyenne de temps de vie avant la première défaillance (*Mean operating Time To Failure*) : $\boldsymbol{MTTF = \int_0^{+\infty} R(t)d(t)}$

– **Taux de défaillance**

A partir de la connaissance des termes *R (t), f (t)* et *F (t)*, on peut définir la notion de taux de défaillance au temps *t* qui est noté universellement par $\boldsymbol{\lambda(t)}$. Formellement $\lambda(t)dt$ représente la probabilité d'avoir une défaillance entre (*t, t + dt*), sachant que l'entité n'a pas été défaillante entre 0 et *t*. En appliquant le théorème des probabilités conditionnelles, il vient, si *dt* est petit [SMI06]: $\boldsymbol{\lambda(t) = -\frac{1}{R(t)}\frac{d\,R(t)}{dt}}$.

Pendant la durée de vie d'une entité, on constate que son taux de défaillance évolue au cours du temps. Après la première mise en service, sur une période appelée **période de jeunesse**, le taux de défaillance est élevé puis il a tendance à décroître. Cela correspond à la période de rodage des systèmes mécaniques ou au déverminage de cartes électroniques. Ensuite le taux de défaillance de la majorité des entités est caractérisé par une valeur constante, ce qui signifie que la probabilité d'une défaillance est identique pour chaque instant considéré. C'est la **période de défaillance à taux constant**. Ensuite, en raison du vieillissement des matériaux des composants ou de leur usure, on observe généralement un taux de défaillance qui se met à croître de façon significative. Cette période de la vie d l'entité est appelée la **période de vieillissement** ou d'usure. Cette évolution est connue sous le nom de « courbe en baignoire » comme le montre la figure 1.4.

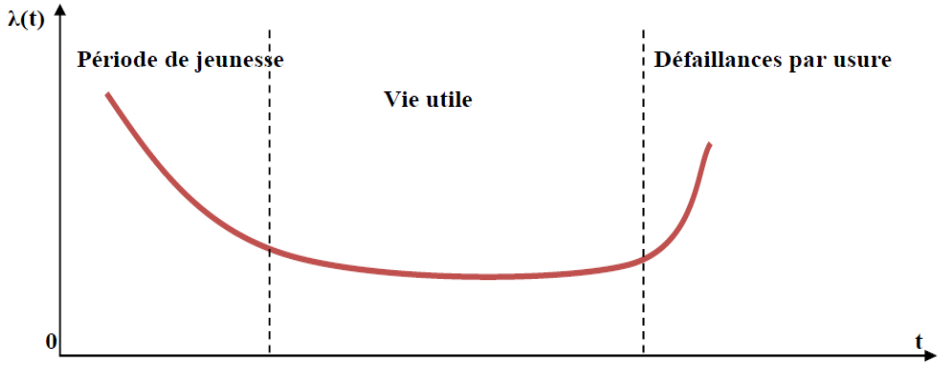

Figure 1.4. Evolution du taux de défaillance $\lambda = f(t)$ (Courbe en baignoire) **[SMI06].**

– **Loi de fiabilité exponentielle**

La durée de vie d'un composant est souvent modélisée à l'aide d'une loi de probabilité réelle des observations. Dans la pratique, il est souvent admis qu'une durée de vie obéit à une loi exponentielle. Le taux de défaillance est constant $\boldsymbol{\lambda(t) = \lambda}$ Alors $\boldsymbol{f(t) = \lambda e^{-\lambda t}}$; $\boldsymbol{R(t) = e^{-\lambda t}}$ **;** $\boldsymbol{MTTF = \frac{1}{\lambda}}$ **.**

1.2.3.2 Disponibilité

La norme AFNOR *X 60-500* définit la disponibilité comme « l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens extérieurs nécessaires de maintenance soit assurée ».

La probabilité associée ***A(t)*** à l'instant *t* est aussi appelée disponibilité et s'exprime par :

$A(t) = P\ (E\ non\ défaillante\ \text{à}\ l'instant\ t)$.

L'aptitude contraire est appelée indisponibilité et est définie par **:** $\boldsymbol{\bar{A}(t) = 1 - A(t)}$

– Les grandeurs moyennes associées à la disponibilité les plus courantes sont:

▪ **le temps moyen de disponibilité** (*TMD*) ou durée de bon fonctionnement après réparation, ou *mean up time* (*MUT*) : durée moyenne de fonctionnement après la réparation et la défaillance suivante.

▪ **le temps moyen d'indisponibilité** (*TMI*) ou durée moyenne d'indisponibilité, ou *mean down* (*MDT*) : durée moyenne entre une défaillance et la remise en état suivante.

▪ **la durée moyenne entre défaillance** notée *MTBF* (*mean time between failure*) : durée moyenne entre deux défaillances consécutives de l'entité. En général, on a la relation : $MTBF = MUT + MDT$

▪ **Temps moyen de réparation** noté *MTTR* (*mean time to repair*) est la durée moyenne jusqu'à la réparation d'une entité réparable.

La figure 1.5 montre les différentes mesures de fiabilité et de disponibilité.

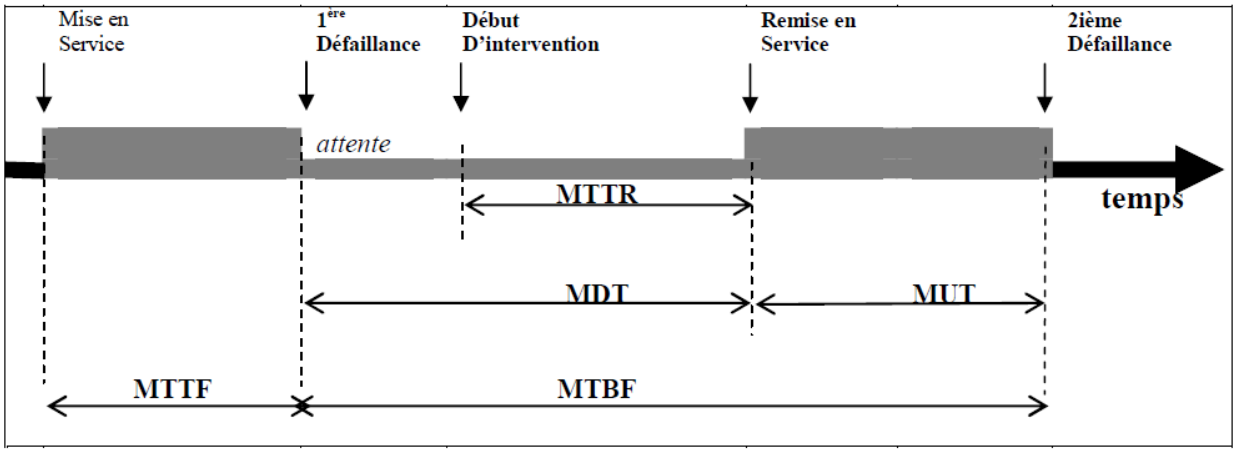

Figure 1.5. Les différentes mesures de fiabilité [JAM04].

– **Le taux de réparation**

Le taux de réparation μ se rapporte également à l'indisponibilité : il s'agit de la durée d'indisponibilité exprimée sous forme de fraction, sous la formule : $\boldsymbol{\mu = \frac{1}{MTTR}}$ [SMI06].

– **Types de disponibilités**

Trois types de disponibilités peuvent être distingués [EBE97]:

▪ **la disponibilité instantanée prévisionnelle** *A(t)* (définie précédemment) :

$A(t) = P\ (E\ non\ défaillante\ à\ l'instant\ t)$

▪ **la disponibilité asymptotique** *A*, si elle existe: $A = \lim_{t \to +\infty} A(t)$. La disponibilité asymptotique peut s'exprimer en fonction de *MTTF* et de *MTTR* : $A = \frac{MTTF}{MTTF+MTTR}$

▪ **la disponibilité moyenne** est la moyenne sur un intervalle de temps donné [t1, t2] de la disponibilité instantanée prévisionnelle, ou mesurée en phase opérationnelle par la durée de fonctionnement effectif divisée par la durée donnée.

▪

– Par supposition de réparation des composants, on peut distinguer deux catégories de systèmes: les systèmes non réparables et les systèmes réparables.

- **Systèmes non réparables**

Les systèmes non réparables sont les systèmes dont les composants défaillants ne seront pas réparés, mais seront directement remplacés. Un matériel non réparable est un matériel que l'on ne veut pas ou que l'on ne peut pas remettre en état. La disponibilité *A(t)* et la fiabilité *R(t)* se confondent dans les systèmes non réparables.

- **Systèmes réparables**

Lorsqu'on a affaire à des systèmes réparables, la durée jusqu'à la première défaillance du système *T* constitue aussi la première période de fonctionnement, nous la notons U_1. Elle est suivie d'une période de remise en service V_1. Par la suite, ces deux types de périodes se succèdent en alternance comme le montre la figure 1.6.

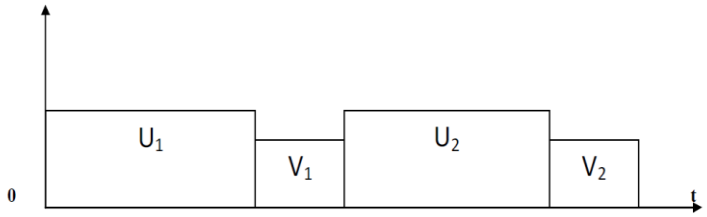

Figure 1.6. Alternance des périodes fonctionnement/réparation [RUE89].

1.2.4 Types de maintenance des systèmes réparables

Il existe deux types de maintenances : la maintenance corrective et la maintenance préventive. La différence entre elles réside dans le moment d'intervention vis-à-vis de la panne. Le premier type de maintenance est appliqué après l'occurrence de la panne, alors que le deuxième type s'applique avant cette dernière.

1.2.4.1 Maintenance corrective

Selon la norme AFNOR *NF EN 13306 X 60-319*, c'est une « maintenance exécutée après détection d'une panne et destinée à remettre un bien dans un état dans lequel il peut accomplir une fonction requise ».

La maintenance corrective est le type de maintenance s'apparentant le plus à l'entretien traditionnel dans la mesure où on intervient sur le matériel après l'apparition d'une défaillance en vue de le remettre en service. On en distingue deux types:

– *Maintenance palliative :* concernant principalement des opérations de dépannage (action sur un bien en panne en vue de le remettre en état de fonctionnement, provisoirement avant réparation) dont l'objectif est de supprimer les effets de la défaillance, elles sont de caractère provisoire.

– *Maintenance curative* : regroupe les opérations de réparation (intervention définitive et limitée de maintenance corrective), dont l'objectif est de ramener le matériel à un niveau de performance donné, elles sont de caractère définitif.

1.2.4.2 Maintenance préventive

Selon la norme AFNOR *NF EN 13306 X 60-319*, c'est une « maintenance exécutée à des intervalles prédéterminés ou selon des critères prescrits et destinée à réduire la probabilité de défaillance ou la dégradation du fonctionnement d'un bien ». Donc c'est une intervention prévue, préparée et programmée en fonction de différents paramètres en vue d'éviter l'apparition probable d'une défaillance identifiée. Il existe deux formes principales de la maintenance préventive :

– *Maintenance préventive systématique* : « Maintenance préventive exécutée à des intervalles de temps préétablis ou selon un nombre défini d'unités d'usage mais sans contrôle préalable de l'état du bien » *(norme NF EN 13306 X 60-319)*. Cette maintenance comprend des inspections périodiques et des interventions planifiées.

– *Maintenance préventive conditionnelle* : « Maintenance préventive basée sur une surveillance du fonctionnement du bien et/ou des paramètres significatifs de son fonctionnement intégrant les actions qui en découlent » *(norme NF EN 13306 X 60-319).* Dans ce cas il n'y a pas d'échéancier mais c'est le franchissement d'un seuil qui provoque l'intervention. Elle peut être appliquée à des matériaux dont le comportement est peu ou pas connu.

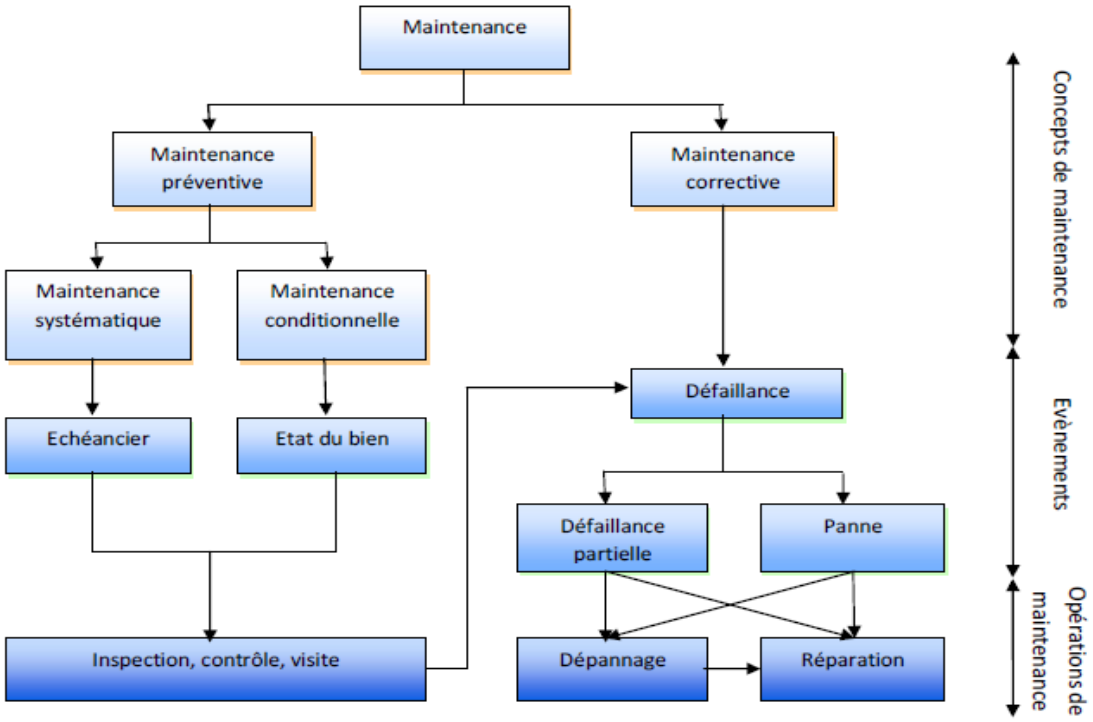

Figure1.7 . Types de la maintenance [AFNOR].

Conclusion

Dans ce chapitre, nous avons présenté, de manière générale, les systèmes de production tant au niveau fonctionnel qu'au niveau structurel. La maintenance étant une fonction complexe, nous avons énoncé dans la seconde partie un certain nombre de concepts concernant la maintenance des systèmes de production. L'objectif premier de ce chapitre est la mise en place des deux concepts de base de ce mémoire : la production et la maintenance.

Le chapitre suivant présentera les problèmes d'ordonnancement dans les ateliers de production.

Chapitre 2

L'ORDONNANCEMENT DANS LES ATELIERS DE PRODUCTION : CAS DU FLOW SHOP HYBRIDE

Introduction

La théorie de l'ordonnancement est une branche de la recherche opérationnelle et de la gestion de production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. Le problème d'ordonnancement consiste à organiser, dans le temps, la réalisation des tâches compte tenu des contraintes de temps et de ressources, afin d'optimiser un ou plusieurs objectifs. Des modèles mathématiques et des méthodes de résolution sont conçus pour résoudre les problèmes posés. L'intérêt porté à cette thématique est largement motivé par l'apparition de ces problèmes dans des domaines aussi variés que nombreux, notamment le secteur industriel Pinedo [PIN95] et celui de l'informatique Blazewicz et al. [BEPSW96]. Parmi les nombreux ouvrages de référence qui ont été publies, on trouve Rinnooy Kan [RIN76], Carlier et Chrétienne [CC88], Esquirol et Lopez [EL99], Blazewicz et al. [BEPSW07].

Ce chapitre se présentera en deux parties : nous présenterons, dans la première, quelques notions de base concernant les problèmes d'ordonnancement dans les ateliers de production. Nous nous intéresserons dans la deuxième partie à l'étude du Flow Shop Hybride.

2.1 L'ORDONNANCEMENT DANS LES ATELIERS DE PRODUCTION

2.1.1 Présentation des problèmes d'ordonnancement

En dehors de tout contexte d'application, les problèmes d'ordonnancement sont définis comme : la programmation dans le temps de l'exécution d'une série de tâches (activités) sur un ensemble de ressources physiques (humaines et techniques), en cherchant à optimiser certains critères ou objectifs (financiers ou technologiques), tout en respectant les contraintes de fabrication et d'organisation [EL99]. Établir un ordonnancement revient donc à coordonner l'exécution de toutes les tâches, en utilisant au mieux les ressources disponibles.

En d'autres termes, il s'agit de : « déterminer ce qui va être fait, quand, où et avec quels moyens. Etant donné un ensemble de tâches à accomplir, le problème d'ordonnancement consiste à déterminer quelles tâches doivent être exécutées et à assigner des dates et des ressources à ces tâches de façon à ce que les tâches soient, dans la mesure du possible, accomplies en temps utile, au moindre coût et dans les meilleures conditions **»** [PAR85].

Les différentes données sont donc : *les tâches*, *les ressources*, *les contraintes* et *les objectifs*.

Une solution à un tel problème consiste à trouver une planification des tâches sur les ressources en optimisant les objectifs et en respectant les contraintes.

Dans le cadre de notre étude, nous nous intéressons principalement aux problèmes d'ordonnancement d'atelier.

2.1.2 Formulation d'un problème d'ordonnancement

2.1.2.1 Les tâches et les opérations

La réalisation d'un ordonnancement d'atelier est décomposable en tâches (dits aussi travaux, pièces ou produits). Une opération est une entité élémentaire d'une tâche qui a une date de début et une date de fin. Son exécution est caractérisée par une durée appelée temps opératoire où temps d'exécution et nécessite l'utilisation d'une ou plusieurs ressources. Les opérations peuvent être exécutées par morceaux, on parle alors d'*opérations morcelables* ou encore *de problèmes préemptifs*. Dans ce cas, l'exécution d'une opération peut être interrompue et reprise ultérieurement tout en poursuivant l'exécution de l'opération à partir de l'état où elle a été interrompue [CC88]. Il existe aussi certains cas où la préemption d'une opération est permise mais son exécution nécessite de recommencer l'opération totalement ou partiellement. Dans d'autres cas, les opérations doivent être exécutées sans interruption. Il s'agit d'opérations *non morcelables* [EL99] qui ne peuvent pas être interrompues avant que leur exécution sur une ressource donnée ne soit terminée.

Une tâche $\boldsymbol{J_j}$ est localisée dans le temps par :

$\boldsymbol{p_{ij}}$: La durée opératoire de la tâche J_j sur la machine M_i.

$\boldsymbol{r_{ij}}$: La date de début au plutôt ou date de disponibilité de la tâche J_j sur la machine M_i.

$\boldsymbol{d_{ij}}$: La date de fin au plus tard ou *(deadline*) de la tâche J_j sur la machine M_i.

$\boldsymbol{t_{ij}}$: La date de début d'exécution de la tâche J_j sur la machine M_i.

$\boldsymbol{c_{ij}}$: La date de fin d'exécution de la tâche J_j sur la machine M_i.

$\boldsymbol{T_{ij}}$: Le retard vrai de la tâche J_j sur la machine M_i $T_{ij} = max\{0, c_{ij} - d_{ij}\}$.

$\boldsymbol{U_{ij}}$: L'indice de retard Tel que U_{ij} =1 si la tâche J_j est en retard, 0 sinon.

$\boldsymbol{W_j}$: Le poids de la tâche J_j.

2.1.2.2 Les ressources

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources [EL99]:

– ***Les ressources renouvelables*** qui sont à nouveau réutilisable après avoir réalisé une tâche (tels que les équipements, les ouvriers, etc.). Parmi ces ressources, nous retrouvons les

ressources disjonctives (non partageables) qui ne peuvent exécuter qu'une seule tâche à un instant donné et les ressources cumulatives (partageables) qui peuvent exécuter un nombre limité de tâches simultanément.

– ***Les ressources consommables*** qui s'épuisent après leur utilisation et ne sont plus disponibles pour les tâches restants à exécuter (comme la matière première, budget, etc.).

– ***Les ressources doublement contraintes ou limitées*** qui combinent les contraintes liées aux deux catégories précédentes. Elles présentent une limitation de leur utilisation instantanée et de leur consommation globale. C'est le cas des ressources d'énergie (pétrole, électricité, etc.).

2.1.2.3 Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision, liées aux tâches et aux ressources. Il s'agit des conditions à respecter lors de la construction d'un ordonnancement pour qu'il soit réalisable. Il existe deux catégories de contraintes : *les contraintes temporelles* et *les contraintes de ressources* [EL99]. Le premier type concerne les contraintes d'antériorité ou de cohérence technologique, qui décrivent des relations d'ordre relatif entre les différentes tâches, les délais de fabrication imposés et les contraintes d'interdiction, d'autorisation ou d'interruption des tâches, etc. Quant au second type, il est lié aux caractéristiques d'utilisation et de disponibilité des ressources utilisées par les tâches. Deux types de contraintes de ressources se distinguent par rapport à la nature disjonctive ou cumulative des ressources.

2.1.2.4 Les critères d'optimisation

Pour évaluer la qualité d'un ordonnancement, il existe plusieurs critères généralement liés aux temps, aux ressources ou aux coûts. Il s'agit, le plus souvent, de considérer le maximum ou la somme (qui peut être une somme totale ou pondérée) sur toutes les tâches d'une certaine mesure ou d'une combinaison de plusieurs mesures. Parmi les critères les plus utilisés, nous retrouvons :

– ***la durée totale de l'ordonnancement*** (*makespan*) définie par $\boldsymbol{Cmax = max\ C_j}$, C_j étant la date de fin d'exécution de la tâche *j* qui représente la date d'achèvement de la tâche la plus tardive. En minimisant ce critère, nous avons une utilisation plus efficace des ressources [RA09].

– ***la somme des dates de fin des tâches*** $\sum_{1\leq j\leq n} C_j$.

– ***le plus grand retard algébrique*** défini par $L_{max} = \max L_j, tel\ que\ L_j = C_j - d_j$ est le retard algébrique (*lateness* en anglais) et d_j désigne la date de fin souhaitée ou encore la date d'échéance (*due date* en anglais).

– ***le plus grand retard vrai*** $T_{max} = \max_{1\leq j\leq n} T_j$, où $T_j = max(0, C_j - d_j)$ est le retard vrai (*tardiness* en anglais) du tâche *j* ;

– ***la somme des retards*** $\sum_{1\leq j\leq n} T_j$.

– ***le nombre de tâches en retard*** $\sum_{1\leq j\leq n} U_j$, U_j étant l'indicateur de retard qui est égal à 1 si $T_j>0$ et 0 sinon ;

– ***la somme des avances*** $\sum_{1\leq j\leq n} E_j$, tel que $E_j = max(0, d_j - C_j)$ représente l'avance (*earliness* en anglais) de la tâche *j* ;

– ***la somme des encours*** $\sum_{1\leq j\leq n} F_j$, définie par le temps de séjour des tâches (*flowtime* en anglais) dans l'atelier. $F_j=C_j-r_j$, r_j étant la date de disponibilité de la tâche *j*.

Une caractéristique intéressante pour un critère est sa régularité. Un critère est dit ***régulier*** si la valeur de la fonction objectif ne peut pas diminuer par l'insertion de temps mort entre deux tâches consécutives [HOO05]. Il s'agit d'un critère dont le coût d'une solution est une fonction croissante des dates d'achèvement des tâches [GKLWS02].

2.1.3 Caractéristiques générales des ordonnancements

- **Ordonnancement admissible:** un ordonnancement est dit admissible s'il respecte toutes les contraintes du problème, à savoir les dates limites, précédences, limitation des ressources, etc.
- **Ordonnancement actif:** dans un ordonnancement actif, aucune tâche ne peut commencer plutôt sans reporter le début d'une autre.
- **Ordonnancement semi-actif:** dans un ordonnancement semi actif, on ne peut avancer une tâche sans modifier la séquence sur la ressource.
- **Ordonnancement sans retard:** dans un ordonnancement sans retard, on ne doit pas retarder l'exécution d'une tâche, si celle-ci est en attente et si la ressource est disponible.

2.1.4 Classification des problèmes d'ordonnancement

Une classification 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. Un atelier est définit par le nombre de machines qu'il contient et par son type.

Les différents types possibles sont les suivants :

– **Une machine ∅:** chaque tâche est constitué d'une seule opération.

– **Machines parallèles :** elles remplissent, à priori, toutes les mêmes fonctions, selon leur vitesse d'exécution, on distingue :

▪ **Machines identiques (*P*) :** la vitesse d'exécution est la même pour toutes les machines M_i et toutes les tâches J_j.

▪ **Machines uniformes *(Q*):** chaque machine M_i a une vitesse d'exécution propre et constante, la vitesse d'exécution est la même pour toutes les tâches J_j d'une même machine M_i.

▪ **Machines indépendantes (*R*) :** la vitesse d'exécution est différente pour chaque machine M_i et pour chaque tâche J_j.

– **Machines dédiées** : elles sont spécialisées dans l'exécution de certaines opérations. Dans cette catégorie, chaque tâche est constitué de plusieurs opérations. En fonction du mode de passage des opérations sur les différentes machines, trois ateliers spécialisés sont différenciés, à savoir :

▪ **flow Shop de permutation (*F*)** (figure 2.2) : c'est un cas particulier du problème d'ordonnancement d'atelier pour lequel le cheminement des tâches est unique: les *n* tâches utilisent les *m* machines dans l'ordre 1,2, . . ., *m* (ligne de production).

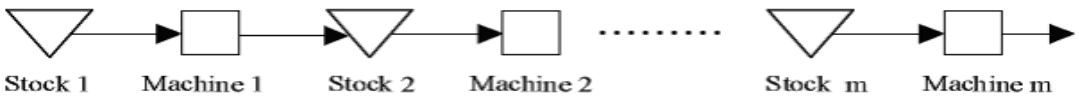

Figure 2.1. Représentation d'un atelier flow Shop.

▪ **job Shop (*J*)** (figure 2.3) : les *n* tâches doivent être exécutes sur les *m* machines, sous des hypothèses identiques à celles du flow shop, la seule différence est que les séquences opératoires relatives aux différents tâches peuvent être distinctes et sont propres à chaque tâche.

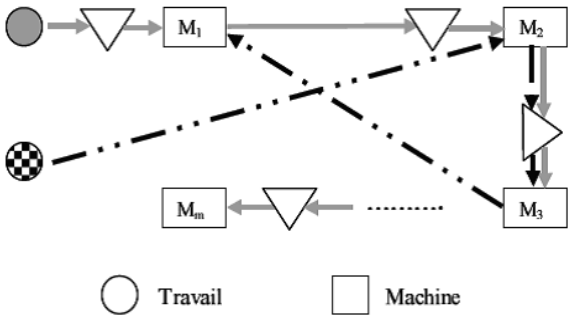

Figure 2.2. Représentation d'un atelier job Shop.

▪ **Open Shop (*O*)** : c'est un modèle d'atelier moins contraint que le flow shop et le job shop, car l'ordre des opérations n'est pas fixé à priori. Le problème d'ordonnancement consiste d'une part à déterminer le cheminement de chaque tâche et d'autre part à ordonnancer les tâches en tenant compte des gammes. Notons que ces deux problèmes peuvent être résolus simultanément. En comparaison aux autres modèles d'ateliers multi-machines, l'open shop qui n'est pas non plus courant dans les entreprises, n'est pas très étudié dans la littérature.

Notation : une notation proposée par [RIN76] (Voir aussi [BEPSW96]) est couramment utilisée pour distinguer un problème d'ordonnancement de manière synthétique et précise. Elle est composée de trois champs d'identification, qui sont notés par le triplet $\boldsymbol{\alpha}$ **/**$\boldsymbol{\beta}$ **/**$\boldsymbol{\gamma}$

– $\boldsymbol{\alpha} = \boldsymbol{\alpha_1}\boldsymbol{\alpha_2}$: décrits les caractéristiques des machines.

$\boldsymbol{\alpha_1} \in \{\emptyset, P, R, O, F, J\}$

$\emptyset$ **:** ordonnancement sur une seule machine.

P **:** ordonnancement sur plusieurs machines parallèles et identiques.

Q **:** ordonnancement sur plusieurs machines parallèles et uniforme.

R **:** ordonnancement sur plusieurs machines parallèles et indépendantes.

O **:** il s'agit d'un problème (Open Shop).

F **:** il s'agit d'un problème (Flow Shop).

J **:** il s'agit d'un problème (Job Shop).

α_2 : le nombre de machines.

– $\boldsymbol{\beta}$ **:** l'ensemble des contraintes.

– $\boldsymbol{\gamma}$ **:** le critère à optimiser.

Exemples :

1// $\sum w_i C_i$: Représente le problème de minimisation de la somme pondérée des dates de fin des tâches indépendantes sur une seule machine.

$F3|V_i|\sum T_i$: Représente les problèmes de minimisation de la somme des retards en Flow Shop à 3 machines avec contrainte des dates de disponibilité.

2.1.5 Représentation d'un ordonnancement

En ordonnancement, il est utile de disposer d'une représentation graphique plus précise des solutions établies. Ce but est atteint par l'utilisation d'un diagramme de Gantt. Ce diagramme constitue un formalisme graphique qui a été mis au point par Henry Gantt en 1910. Il s'agit d'un outil qui est couramment utilisé pour représenter la solution d'un problème d'ordonnancement. Il permet de visualiser l'utilisation des machines, l'avancement de l'exécution des opérations sur celles-ci ainsi que les dates de début et de fin de chaque opération. La Figure ci-après représente le diagramme de Gantt associé à l'ordonnancement de trois tâches (J_1, J_2, J_3) dans un atelier flow-shop à deux machines. Les durées opératoires sont données dans le tableau suivant.

	J_1	J_2	J_3
M_1	4	2	4
M_2	2	6	4

Table 2.1 Temps opératoires pour l'exemple d'un flow-shop.

À partir de ce diagramme on peut déterminer par exemple la valeur de la date d'achèvement de la tâche la plus tardives.

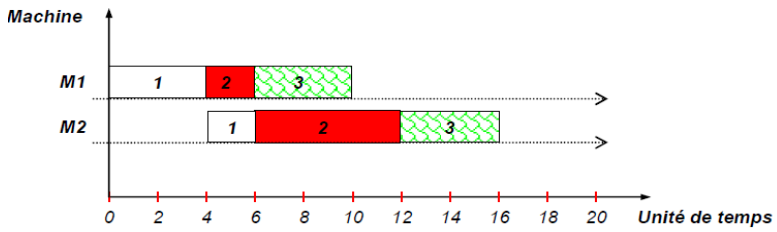

Figure 2.3.Exemple d'un diagramme de Gantt.

2.1.6 Théorie de la complexité des algorithmes

Les problèmes d'ordonnancement d'ateliers, sont des problèmes combinatoires, extrêmement difficiles et il n'existe pas de méthode universelle permettant de résoudre efficacement tous les cas, beaucoup d'entre eux peuvent prendre un temps considérable pour être résolus. La théorie de la complexité des algorithmes a donné un sens précis aux termes d'algorithme efficace et de problème difficile.

– Les classes P et NP

Pour pouvoir exposer la notion de classe de problèmes, il est tout d'abord nécessaire de distinguer les **problèmes de décision** des **problèmes d'optimisation.** Un problème de décision est un problème pour lequel la réponse est "oui" ou "non". Il est possible d'associer à chaque problème d'optimisation, un problème de décision en introduisant un seuil *k* correspondant à la fonction objectif *f*. Le problème de décision devient : "existe-t-il un ordonnancement réalisable *(S)* tel que $f(S) \leq k$?".

Il est alors possible de définir la classe *P* qui regroupe les problèmes de décision résolubles par des algorithmes polynomiaux [FLI98]. Un algorithme polynomial est défini comme un algorithme dont le temps d'exécution est borné par *O(p(x))* ou *p* est un polynôme et *x* la longueur d'entrée d'une instance du problème. Les algorithmes dont la complexité ne peut pas être bornée polynomialement sont qualifiés d'exponentiels.

La classe *NP* regroupe les problèmes qui peuvent être résolus en temps polynomial par un algorithme non déterministe (Algorithme qui comporte des instructions de choix) [GIA03], [FLI98]. Pour ces algorithmes, si à chaque instruction le bon choix est effectué, le temps de calcul est polynomial. Si au contraire tous les choix sont énumérés, l'algorithme devient déterministe et son temps de calcul devient exponentiel. Les algorithmes "ordinaires" sont évidemment des cas particuliers des algorithmes non déterministes [FLI98]. Aussi tout problème de décision qui peut être résolu par un algorithme polynomial, et qui donc appartient à la classe *P*, appartient également à la classe *NP*. D'ou $P \subseteq NP$.

Parmi les problèmes de la classe *NP*, une large classe de problèmes, les problèmes ***NP*-complets**, sont équivalents entre eux quant à l'existence d'un algorithme polynomial pour les résoudre. C'est à dire, s'il existe un algorithme polynomial pour résoudre un seul de ces problèmes, alors il en existe un pour chaque problème de la classe *NP*. Afin de décrire cette

classe d'équivalence, définissons tout d'abord la réduction polynomiale entre deux problèmes. Soient *P1* et *P2*, deux problèmes de décision. On dit que *P1* se réduit polynomialement à *P2* (noté $P1 \propto P2$) s'il existe un algorithme de résolution de *P1*, qui fait appel à un algorithme de résolution de *P2*, et qui est polynomial lorsque la résolution de *P2* est comptabilisée comme une opération élémentaire. On dit alors qu'un problème de décision est *NP*-complet si tout problème de la classe *NP* se réduit polynomialement à lui [GIA03]. Les problèmes d'optimisation combinatoire dont le problème de décision associé est *NP*-complet sont qualifiés de ***NP*-difficiles.**

2.1.7 Méthodes de résolution

Les problèmes d'ordonnancement d'atelier sont des problèmes d'optimisation combinatoire qui nécessitent d'effectuer un nombre important de calculs pour obtenir un ordonnancement admissible (ou réalisable) qui optimise le (ou les) critère(s) retenu(s) en tenant compte des contraintes. Le développement de la théorie de complexité a permis de classifier ces problèmes selon leurs difficultés [RIN76]. Dans la plupart des cas, ils ont été démontrés NP-difficiles [LLRS93]. 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 et les méthodes approchées. La programmation dynamique, les méthodes par séparation et évaluation ("Branch-and- Bound") et la programmation linéaire en nombres entiers, sont les principales méthodes de résolution exacte. Elles nous permettent de trouver une solution optimale grâce à une exploration efficace de l'espace des solutions. De manière générale, ce type d'approches permet de résoudre les problèmes d'ordonnancement de petite taille, mais il reste difficile de fournir une résolution exacte en un temps raisonnable lorsque les problèmes sont de taille importante. D'où la nécessité d'adopter des méthodes de résolution approchée proposant des solutions acceptables en un temps de calcul réduit.

2.2 LE PROBLEME D'ORDONNANCEMENT DU FLOW SHOP HYBRIDE

2.2.1 Présentation

Une organisation de type Flow Shop Hybride (**FSH**) constitue notre cas d'étude. Il s'agit d'ordonnancer un ensemble de *n* traveaux types qui sont représentatifs des scénarios de

production possibles $J_1, J_2, ..., J_n$ dans un atelier de production composé de k étages ($k \geq 2$). Chaque tâche J_j est constituée de k opérations types O_{je}: O_{je} est l'opération de la tâche j traitée par l'étage e. L'étage e contient un ensemble de $N_e \geq 1$ machines parallèles identiques, ce qui signifie que les traveaux peuvent être exécutés indifféremment sur l'une ou l'autre des machines d'un même étage. Chaque tâche doit passer sur une des machines dans chaque étage (voir figure 2.5)

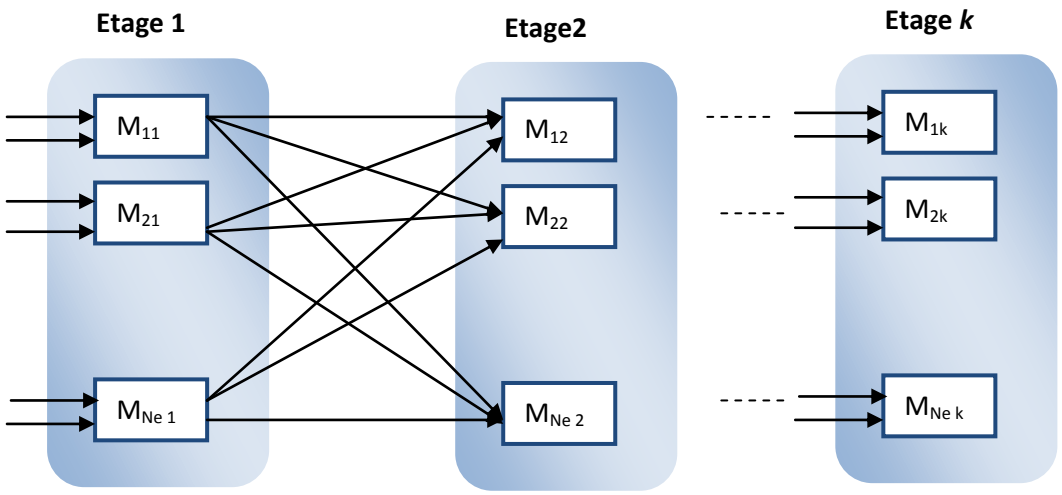

Figure 2.4 . Flow Shop Hybride à k étages.

Une fois que l'exécution d'une tâche ait débuté sur une machine, celle-ci ne peut pas être interrompue. Ceci revient à dire que l'ordonnancement est non préemptif. La tâche ne peut pas être exécutée sur différentes machines. Aucune tâche ne peut commencer sur une machine avant la fin de la tâche en cours. Pour chacun des tâches, l'ordre de passage des opérations dans les k étages est le même : $O_{j1}, O_{j2}, ..., O_{jk}$. La machine M_{ie} avec $i = 1, 2, ..., N_e$ et $e = 1, 2, ..., k$ ne peut exécuter qu'une seule tâche à la fois. Le temps opératoire de la tâche j à l'étage e est noté P_{je} (les temps de montage, de transport, de décharge, etc. sont contenus dans le temps opératoire de la tâche).

2.2.2 Notations

La notation $\alpha / \beta / \gamma$ [RIN76] est insuffisante pour prendre en considération le problème du Flow Shop Hybride, ainsi elle a été reprise par [VBP99] dans le but de l'enrichir. L'extension

proposée porte essentiellement sur le champ α qui décrit l'environnement des machines. Il se compose de quatre paramètres que l'on représente ainsi : $\boldsymbol{\alpha = \alpha_1 \alpha_2 \left(\left(\alpha_3 \alpha_4{}^{(i)} \right)_{i=1}^{\alpha_2} \right).}$

– $\boldsymbol{\alpha_1}$ représente les caractéristiques des machines, on le note FH dans le cas du Flow Shop Hybride ;

– $\boldsymbol{\alpha_2}$ indique le nombre d'étages $\boldsymbol{\alpha_2 = k}$;

– $\boldsymbol{\alpha_3} \in \{\emptyset, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{R}\}$ précise le type des machines sur l'étage *i* : s'il s'agit de problème à une machine ($\emptyset$) ou à machines parallèles qui sont soit identiques (*P*), soit uniformes (*Q*), ou soit encore, non reliées (*R*) ;

– $\boldsymbol{\alpha_4 = N_e}$ est le nombre de machines à l'étage *e*.

Le champ $\boldsymbol{\beta}$ permet, comme dans les notations classiques, de définir l'ensemble des contraintes et le champ $\boldsymbol{\gamma}$ indique le critère que l'on cherche à optimiser.

L'illustration de cette notation par un exemple de flow-shop hybride à 5 étages, composé de 5 machines identiques aux deux premiers étages, de 2 machines identiques au troisième étage, et de 3 machines identiques pour les autres et dont le critère à minimiser est le $\boldsymbol{C}_{\max}$, est noté de la façon suivante: $\boldsymbol{FH5\left(P5^{(1)}, P5^{(2)}, P2^{(3)}, P3^{(4)}, P3^{(5)}\right) || C_{max}}$.

2.2.3 Complexité des problèmes d'ordonnancement de type Flow Shop Hybride

Les problèmes d'ordonnancement de type Flow Shop Hybride ont fait l'objet de plusieurs études depuis les années 1970. Ils sont classés parmi les problèmes les plus difficiles et les plus rencontrés dans le domaine industriel. Les premiers résultats concernant la complexité du FSH ont été présentés dans [GJ88] et ils ont prouvé qu'il s'agit d'un problème NP-difficile pour le cas à deux étages, où un étage comporte deux machines et l'autre une seule machine, avec pour critère d'optimisation la minimisation de la durée totale de l'ordonnancement. De même Hoogeveen et al. [HLV96] ont démontré que ce même problème est NP-difficile dans le cas où la préemption est autorisée.

Dans un premier temps, le flow-shop hybride à deux étages a suscité l'attention de beaucoup de chercheurs. Ensuite, les travaux se sont orientés vers une généralisation au cas multi étages ($E \geq 3$), avec un nombre de machines quelconques disponibles sur chacun des étages.

Les approches de résolution sont extrêmement variées dans la littérature. Des états de l'art décrivant les différentes méthodes utilisées pour résoudre ce type de problèmes ont été proposés dans [VBP99] et [RV10].

Conclusion

Nous avons présenté dans la première partie de ce chapitre les principales définitions et notations, la classification des problèmes d'ordonnancement, les types et les représentations des ordonnancements et la complexité des problèmes d'ordonnancement. Pour délimiter les contours de notre étude, nous nous somme concentré dans la deuxième partie sur le FSH en donnant quelques notions, notations et termes utiles pour la suite.

Chapitre 3

L'ORDONNANCEMENT CONJOINT PRODUCTION / MAINTENANCE

Introduction

La mise en place de liens entre les fonctions production et maintenance a toujours été une source de conflits dans les entreprises [WC99]. En effet, il existe d'une part un plan de production qui consiste à organiser, dans le temps, la réalisation des tâches compte tenu des contraintes de temps et de ressources, afin d'optimiser un ou plusieurs objectifs traduit en termes de délais et de nombre de tâches à accomplir et d'autre part un plan de maintenance, tendant à assurer la pérennité de l'outil de production. Les arrêts du système de production, à des fins de maintenance, sont trop souvent considérés comme une source de perturbation et de perte de productivité. L'optimisation bi objectif de la production et de la maintenance trouve ainsi tout son intérêt, étant donné que les deux objectifs sont complémentaires et contradictoires. Cette nouvelle méthode d'optimisation nous permet de trouver des compromis entre ces deux objectifs contradictoires. Ces compromis permettent de donner plus de flexibilité au niveau des décisions des deux services et donc d'éviter un éventuel conflit entre eux.

Dans la littérature portant sur l'optimisation de l'ordonnancement de la production et la planification de la maintenance, Les tâches de maintenance sont planifiées séparément de l'ordonnancement sans tenir compte des exigences de l'un et l'autre. L'optimisation conjoint de la maintenance et de l'ordonnancement de production n'a attiré que récemment l'attention des chercheurs.

Nous commençons ce chapitre par la description des différentes politiques d'ordonnancement conjoint de la production et de la maintenance recensées dans la littérature. Un état de l'art sur ces travaux sera donné dans la section suivante. Par la suite, nous présenterons le modèle intégré pour la solution conjointe du problème d'ordonnancement de la production et de la maintenance. Enfin, nous présenterons quelques notions de base et les différentes méthodes d'optimisation multi objectifs.

3.1 POLITIQUES D'ORDONNANCEMENT CONJOINT DE LA PRODUCTION ET DE LA MAINTENANCE

L'objectif de l'ordonnancement conjoint de la production et de la maintenance est de planifier l'exécution des tâches de maintenance, en altérant le moins possible le plan de production, et tout en respectant au mieux la périodicité de maintenance des équipements. Trois politiques de

planification ont été recensées dans la littérature, l'ordonnancement séparé, le séquentiel et l'intégré [LC00].

– **ordonnancement séparé:** actuellement la maintenance et la production sont le plus souvent traitées de manière indépendante au sein de l'entreprise [BEM02]. Les ordonnancements correspondants à ces deux activités sont donc réalisés de manière séparée et interférent bien souvent l'un avec l'autre entrainant des retards de la production ou de la maintenance. Cette méthode implique la mise en place d'une communication accrue entre les services de production et de maintenance pour limiter les conflits dans l'immobilisation des ressources aussi bien humaines que matérielles ;

– **ordonnancement séquentiel:** cette politique consiste à planifier l'une des deux activités, production ou maintenance, et à utiliser cet ordonnancement comme une contrainte supplémentaire d'indisponibilité des ressources dans la résolution du problème d'ordonnancement de l'ensemble des deux types de taches. De manière générale, la maintenance est planifiée en premier, ensuite l'ordonnancement de la production est réalisé en prenant les opérations de maintenance comme des contraintes fortes d'indisponibilité des ressources [AGG02].

– **ordonnancement intégré** [SS00]: cette politique consiste à créer un ordonnancement conjoint et simultané des tâches de production et de maintenance. Une telle politique de planification limite les risques d'interférence entre la production et la maintenance et permet ainsi d'optimiser la qualité des ordonnancements.

3.2 ETAT DE L'ART

Les travaux de recherche tenant en compte l'indisponibilité des machines dans l'ordonnancement de la production peuvent être classés en deux catégories: l'approche déterministe et l'approche stochastique. Dans l'approche déterministe, les intervalles de temps des actions de maintenance préventive ainsi que leur nombre sont connus et fixés à l'avance. La majorité de la littérature d'ordonnancement avec maintenance adoptent cette approche souvent appelée "ordonnancement avec contraintes de disponibilité des machines". Toutes les configurations d'ateliers connus ont été étudiées par les chercheurs : une seule machine, machines parallèles, Flow-Shop, Job-Shop, Open-Shop et les systèmes hybrides. Compte tenu du nombre significatif d'articles réalisés dans cette catégorie, nous examinons seulement l'état de l'art concernant les ateliers Flow Shop Hybride.

Dans [VCP97] et [VBPT96], les auteurs proposent pour la résolution du problème FSH à deux étages des méthodes heuristiques en tenant compte de la disponibilité ou non des machines au premier étage. La fonction objectif étant la minimisation du plus grand retard (dans ce cas, des périodes d'inactivité des machines sont prises en compte).

Un problème FSH à deux étages a été étudié par Xie et Wang [XW05] où il y a seulement une période d'indisponibilité sur la machine du premier étage et aucune période d'indisponibilité sur les *m* machines du deuxième étage.

Allaoui et al. [AA06] ont étudié le cas d'une seule machine au premier étage et plusieurs machines au second étage pour minimiser le Makespan. Ils considèrent que chaque machine est soumise à au plus une période d'indisponibilité due aux activités de maintenance corrective ou préventive et que les dates de début et de fin de chaque période sont connues à l'avance. Ils ont discuté la complexité du problème et ils ont développé une approche basée sur la méthode par Séparation et Evaluation. Ils ont programmé trois heuristiques appelées LIST, LPT et H.

Dans [HHD07], les auteurs ont traité le cas d'un flow shop hybride à deux étages. Le premier étage est composé d'une seule machine (machine commune), quant au second il est composé de *m* machines dédiées. Ils supposent que l'une des *m+1* machines est indisponible sur une période prédéterminée. Leur intérêt est la minimisation du Makespan. Ils présentent un ensemble de propriétés ainsi qu'une analyse du pire des cas selon que la période d'indisponibilité concerne la machine commune ou l'une des machines du second étage.

Jungwattanakit et al. [JRCW08] ont étudié le problème de flow shop hybride avec *n* tâches et *m* machines indépendantes parallèles, à chaque étage. Les auteurs ont formulé le problème comme un programme en nombres entiers mixte 0-1 pour des combinaisons de Cmax et ΣUi. La préemption n'est pas autorisée et une seule période d'indisponibilité est possible pour chaque machine à l'instant zéro. Plusieurs heuristiques ont été proposées pour la résolution du problème. La fonction objectif étant la minimisation du Makespan et les algorithmes génétiques ont été aussi proposés.

Dans [JZTO9], les auteurs ont traité le problème du flow shop hybride avec la contrainte d'indisponibilité des machines pendant l'horizon de production due aux différentes politiques de maintenance préventive considérées. SDST (*sequence-dependent setup times*) est une autre contrainte considérée où le temps de configuration de chaque tâche est inclus dans son temps de traitement et dépend de la tâche précédente. Ils ont proposé trois heuristiques basées sur SPT, LPT et la règle de Johnson ainsi que deux méta-heuristiques basées sur les algorithmes génétiques et le recuit simulé afin de minimiser le Makespan.

Besbes et al. [BTL10] ont traité le problème d'ordonnancement du flow shop hybride avec *n* étages, les machines ne sont pas disponibles en permanence en raison de tâches de maintenance préventive. Une approche approximative basée sur un algorithme génétique (GA) est proposée, afin de minimiser le Makespan.

Costa et al. [CCF13] ont étudié le problème d'ordonnancement du flow shop hybride avec *m* étages sous la contrainte d'indisponibilité des machines où les intervalles de temps d'indisponibilité sont connus à l'avance pour chaque machine dans chaque étage. Ils ont supposé aussi le chevauchement entre les travaux du même type et la limite de temps d'attente des travaux dans les tampons entre étages. Ces trois contraintes ne représentent qu'une partie des contraintes qui caractérisent le problème étudié. La fonction objectif étant la minimisation du Makspan.

Dans le cas de l'approche stochastique, les débuts des tâches de maintenance préventive sont considérés comme variables de décision comme pour les tâches de production, l'ordonnancement et la maintenance sont considérés conjointement. Il y a peu d'articles qui s'intéressent à cette approche. Kaabi *et al.*, [KVZ02], [KVZ03] ont respectivement étudié les cas à une seule machine et le flow shop de permutation où les périodes de maintenance doivent être effectués dans un intervalle prédéfini. Cette étude a l'avantage de diminuer le conflit entre les deux services, mais ils favorisent la maintenance au détriment de la production.

Xu *et al.* [XSL08] ont utilisé la même idée que dans [KVZ02] pour le cas de machines parallèles pour minimiser le Makespan. Cassady et Kutanoglu [CK03] ont formulé un modèle mathématique intégré dans le cas d'une seule machine pour minimiser le retard total pondéré de la production. Cependant, l'approche par énumération totale est utilisée. En effet, les auteurs ont fait remarquer que le temps de calcul devient insupportable quand le nombre de tâches excède huit, ce qui n'est pas pratique.

Benbouzid et al. [BBGVZ03] [BVZ06] [BGBVZ11] ont proposé une optimisation conjointe pour le cas de flow shop de permutation où les périodes de maintenance doivent être effectués dans un intervalle prédéfini. Deux heuristiques d'ordonnancement séquentiel (les algorithmes génétiques et l'algorithme NEH) sont proposés dans [BBGVZ03], des algorithmes de colonies de fourmis pour résoudre le problème selon deux stratégies séquentielle et intégrée sont proposés dans [BVZ06], une étude comparative sur les deux stratégies (séquentielle et intégrée) est présentée dans [BGBVZ11]. La même approche est présentée pour le cas de flow shop hybride par Benbouzid et al. [BTM07] avec deux méthodes de résolution selon la stratégie intégrée : la méthode de Recherche Tabou et la deuxième méthode basée sur les algorithmes génétiques.

Dans [RGM07], les auteurs proposent une optimisation conjointe pour le cas de flow shop de permutation afin de minimiser le temps d'exécution. Les périodes de maintenance sont déterminées en conservant un niveau minimum de fiabilité au cours de l'horizon de planification. Cependant, les périodes de maintenance sont établies sans tenir compte des exigences de l'ordonnancement. Elles sont fixées à l'avance pour chaque machine séparément.

Jusqu'à présent, nous pouvons noter que toutes ces études ne se sont intéressées qu'à l'ordonnancement de la production au détriment de la maintenance.

Récemment Berrichi et al. [BAYCM09] ont étudié un modèle bi-objectif du problème conjoint dans le cas de machines parallèles en utilisant des modèles de fiabilité pour prendre en considération l'aspect maintenance. Les deux critères utilisés sont la minimisation d'indisponibilité des machines et la minimisation du temps d'exécution des tâches. Pour le critère de la maintenance, c'est l'indisponibilité de l'ensemble du système qui est optimisée au lieu de fixer un seuil de fiabilité pour chaque machine comme dans [RGM07]. Deux algorithmes génétiques ont été développés pour obtenir une approximation du front Pareto optimal : Un algorithme génétique qui utilise la pondération de deux objectifs et un algorithme génétique de type NSGA II (*Non dominated Sorting Genetic Algorithm*). Les intervalles des périodes de maintenance ainsi que leur nombre pour chaque machine sont optimisés au cours du processus d'optimisation globale ainsi que les tâches de production. De même, Un algorithme de colonie de fourmis est appliqué dans [BYAM10].

Moradi et al [MFZ11] se sont inspiré du modèle développé par [BAYCM09] pour étudier un atelier de type job shop flexible. Quatre méthodes d'optimisation multi objectif ont été comparées pour trouver le front Pareto optimale.

La même approche est présentée pour le cas de machines parallèles par Moradi et Zandieh [MZ10]. En outre, Mokhtari et al. [MMA12] ont étudié le même problème pour le cas de machines parallèles en considérant multiples services de maintenance préventive (MPMS pour *multiple preventive maintenance services*) dans laquelle l'approche de fiabilité / disponibilité est utilisée pour modéliser les aspects de maintenance. L'objectif est de trouver la meilleure attribution des travaux aux machines afin de minimiser le Makespan aussi bien que les politiques appropriées de MPMS afin de réduire au minimum l'indisponibilité du système. La politique de MPMS consiste à décider quel service dut être choisi parmi les services possibles de MP et puis l'exécuter sur les machines.

Dans le présent mémoire, nous utilisons la même idée que [BAYCM09] mais pour le cas d'un atelier de type Flow Shop Hybride. Nous considérons comme critère d'ordonnancement la

minimisation de *Cmax*. Pour le critère de maintenance nous considérons le même critère qui est la minimisation de l'indisponibilité d'atelier.

3.3 MODELISATION DU PROBLEME CONJOINT

Nous proposons dans ce mémoire un modèle bi objectif intégré pour le cas de Flow Shop Hybride, utilisant des modèles de fiabilité pour prendre en compte l'aspect maintenance comme dans **[**BAYCM09**]**. A notre connaissance, il n y pas eu de travaux pour le problème conjoint production / maintenance dédié au cas de Flow Shop Hybride traitant le problème avec des méthodes multicritères de type Pareto prenant en considération l'indisponibilité (au sens fiabilité) des machines (du système) comme critère de performance du système de production.

Cette section décrit la formulation du problème d'ordonnancement de la production et du problème de planification de la maintenance préventive (MP) systématique séparément, ensuite le problème bi objectif intégré.

3.3.1.Le problème d'ordonnancement de la production

Nous examinons le cas du Flow Shop Hybride à $\boldsymbol{k}$ étages avec $\boldsymbol{N_e}$ machines parallèles identiques dans chaque étage $\boldsymbol{e}$. Notre objectif consiste à ordonnancer un ensemble de $\boldsymbol{n}$ tâches $\boldsymbol{j_1, j_2, ..., j_n}$ où une tâche est composé d'une seule opération pour chaque étage, et les opérations doivent être réalisées séquentiellement. Chaque tâche $\boldsymbol{j}$ a une durée opératoire $\boldsymbol{p_{je}}$ sur chaque étage $\boldsymbol{e}$.

Nous supposons que les tâches sont disponibles au début de la production et les préemptions ne sont pas autorisées. L'objectif est de minimiser le temps de fin d'exécution le plus tôt de la dernière tâche dans le système. Ce problème est noté $\boldsymbol{FHk}((\boldsymbol{PN_e}^{(e)})_{e=1}^{k})\|\boldsymbol{Cmax}$ et il est prouvé NP-difficile [GJ88] [HLV96].

3.3.2 Le problème de planification de MP

Nous nous intéressons à la maintenance préventive systématique du fait de son caractère planifiable qui se traduit par des interventions planifiées ou des inspections périodiques permettant d'éviter les pannes des machines. Contrairement à la maintenance corrective qui intervient après la survenue de la panne. Les actions de maintenance préventive contribuent à maintenir les outils de production dans de bonnes conditions d'exploitation (elles augmentent la disponibilité du système) et minimisent les coûts en évitant les pannes imprévues. Le problème

de maintenance est donc de déterminer les dates de maintenance pour chaque machine en minimisant l'indisponibilité du système entier.

La disponibilité $\boldsymbol{A}$ d'une machine $\boldsymbol{M}$ est définie en termes probabilistes, à un point donné au temps $\boldsymbol{t}$ comme suit [BAYCM08] [EBE97]:

$$\boldsymbol{A(t) = P(M\ non\ défaillant\ à\ l'instant\ t)} \quad \textbf{(3.1)}$$

Le contraire de la disponibilité est l'indisponibilité $\overline{A}$, elle est définie comme suit [BAYCM08] [EBE97] : $\boldsymbol{\overline{A}(t) = 1 - A(t)}$ **(3.2)**

On suppose dans notre modèle qu'une action de maintenance subie par une machine la rétablit dans des conditions aussi bonnes que neuves (*as good as new*). Aussi, on fait l'hypothèse que les taux de défaillance et les taux de réparation des machines sont constants: $\lambda_i(t) = \lambda_i$, $\mu_i(t) = \mu_i$. Autrement dit, on considère que le temps jusqu'à défaillance (*resp*. temps jusqu'à réparation) d'une machine M_i suivent des distributions de probabilité exponentielles avec comme paramètre le taux de défaillance λ_i (*resp*. taux de réparation μ_i).

En prenant en compte ces hypothèses, à partir de l'état initial $t = 0$, la disponibilité d'une machine M_i à un instant t est donnée par l'équation (3.3) [BAYCM08] [EBE97] [VIL91]:

$$\boldsymbol{A_i(t) = \frac{\mu_i}{\mu_i+\lambda_i} + \frac{\lambda_i}{\mu_i+\lambda_i} \exp[-(\mu_i + \lambda_i)t]} \quad \textbf{(3.3)}$$

La disponibilité $\boldsymbol{A_i\ (t)}$ d'une machine $\boldsymbol{M_i}$ est une fonction décroissante de temps $\boldsymbol{t}$, par conséquent l'indisponibilité $\boldsymbol{\overline{A}_i(t)}$ **=1-$\boldsymbol{A_i\ (t)}$** est donc une fonction croissante. Donc, si aucune action de MP n'est effectuée sur M_i, son indisponibilité va augmenter.

Si T_i est le temps d'achèvement d'une action de maintenance préventive sur la machine M_i, l'expression de la disponibilité $A_i\ (t)$ à l'instant $t > T_i$ est donnée par l'expression suivante [BAYCM08][EBE97][VIL91]:

$$\boldsymbol{A_i(t) = \frac{\mu_i}{\mu_i+\lambda_i} + \frac{\lambda_i}{\mu_i+\lambda_i} \exp[-(\mu_i + \lambda_i)(t - T_i)]} \quad \textbf{(3.4)}$$

La disponibilité du système dépend de sa structure (parallèle, série ou hybride) ainsi que les caractéristiques de ses composants. Pour le cas d'un atelier Flow-Shop hybride, chaque étage $\boldsymbol{e}$ a une fonction de disponibilité $\boldsymbol{A_e(t)}$.

Pour N_e composants indépendants disposés en parallèle, chacun ayant une disponibilité $\boldsymbol{A_i(t)}$ à l'instant $\boldsymbol{t}$, la disponibilité de l'étage $\boldsymbol{e}$ au temps $\boldsymbol{t}$ est donnée par l'équation [BAYCM08] [EBE97] [VIL91]: $\boldsymbol{A_e(t) = 1 - \prod_{i=1}^{i=N_e}[1 - A_i\ (t)]}$ **(3.5)**

L'indisponibilité de l'étage est donc la suivante:

$$\overline{A_e}(t) = 1 - A_e(t) = \prod_{i=1}^{i=N_e}[1 - A_i(t)] \quad \textbf{(3.6)}$$

Le système est disponible à l'instant *t* si tous les étages sont disponibles à cette date. L'équation de la disponibilité du système A_s (*t*) est donc la suivante:

$$A_S(t) = \prod_{e=1}^{e=k} A_e(t) \quad \textbf{(3.7)}$$

Par conséquent, L'indisponibilité du système est donnée par l'équation :

$$\overline{A_S}(t) = 1 - A_S(t) \quad \textbf{(3.8)}$$

La durée d'une tâche de MP sur une machine M_i est supposée égale au temps moyen d'une action de MP dont la valeur est équivalente à $1/\mu_i$ [BAYCM10].

3. 3.3 Le modèle bi objectif intégré

Deux objectifs sont à minimiser simultanément dans notre étude: la minimisation du Makespan pour l'aspect production et la minimisation de l'indisponibilité du système de production pour l'aspect maintenance, sous les conditions définies dans les sections (3.3.1) et (3.3.2). Donc, deux décisions doivent être prises en même temps. La première est de trouver la meilleure affectation des *n* tâches de production aux machines dans l'objectif de minimiser le Makespan. La seconde est de décider quand il faut effectuer les interventions de MP sur les machines, dont leur nombre n'est pas fixé à l'avance, dans l'objectif de minimiser l'indisponibilité du système.

Soit C_j la date de fin de la tâche de production numéro *j* et C_{max} la date de fin de la dernière tâche de production (le Makespan) : $C_{max} = max_{i=1,n}\{C_i\}$ **(3.9)**

Soit $T = \{0, t_1, t_2, \ldots, t_s, C_{max}\}$ où $t_1, t_2, \ldots, t_s$ sont les dates de début des actions de MP sur toutes les machines. Puisque l'indisponibilité est une fonction croissante dans chaque intervalle $[t_i, t_{i+1}]$, i=0, …, s, avec t_0=0 et $t_{s+1} = C_{max}$, et comme nous avons supposé qu'une machine devient aussi bonne que neuve (*as good as new*) à la fin de chaque action de MP, l'indisponibilité du système est calculée uniquement aux instants $t_1, t_2, \ldots, t_{s+1}$ [BAYCM08].

Les deux fonctions objectifs à minimiser sont :

$F_1 = C_{max}$, *qui est le Makespan.*

$F_2 = max_{t \in T}\left\{\overline{A_s}(t) = 1 - \prod_{e=1}^{e=k} A_e(t)\right\}$, *qui est l'indisponibilité du système.*

Où $A_e(t)$ *est la disponibilité de l'étage* e au temps t : $A_e(t) = 1 - \prod_{i=1}^{i=N_e}[1 - A_i(t)]$.

3.4 L'OPTIMISATION MULTI OBJECTIF

L'optimisation multi objectif est un axe de recherche très important à cause de la nature multi objectif de la plupart des problèmes réels. Les premiers travaux menés sur les problèmes multi objectif furent réalisés au 19éme siècle sur des études en économie par Edgeworth et généralisés par Pareto.

3.4.1 Définitions et Principes de base

Coello Coello *et al.* [CC02] ont déjà présenté les principes et les notions de base de l'optimisation à objectifs multiples. Dans cette section, les points les plus importants sont décrits. Un problème d'optimisation multi objectifs peut être formulé de la manière suivante [CC02]:
Trouver le vecteur $\boldsymbol{x}^* = (x_1^*, x_2^*,...,x_n^*)^T$ qui satisfait les *m* contraintes d'inégalités et les *p* contraintes d'égalités suivantes :

$g_i(\boldsymbol{x}) \geq 0 \quad i = 1,2,...,m$

$h_i(\boldsymbol{x}) = 0 \quad i = 1,2,..., p$

Où: g_i et h_i sont les contraintes exprimées sous forme mathématique en optimisant (minimiser ou maximiser) le vecteur de fonctions suivant : $\boldsymbol{f}(\boldsymbol{x}) = \left(f_1(x), f_2(x), \dots, f_k(x)\right)^T$
sachant que $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ est le vecteur des variables de décision.

Le principe d'une optimisation multi objectif est différent du principe d'une approche mono objectif. Le but principal d'une optimisation mono objectif est de trouver la solution optimale globale qui résulte en la meilleure valeur (plus petite ou plus grande) de la fonction mono objectif. Dans un problème d'optimisation multi objectifs, il y a plus qu'une fonction objectif ($k \geq 2$), chaque fonction objectif pouvant avoir une solution optimale différente. Le but d'un problème multi objectif est de trouver de bons compromis plutôt qu'une seule solution. Lorsqu'il y a plusieurs objectifs, la notion d'optimum change et il est préférable d'utiliser un autre terme, le terme le plus couramment adopté étant l'optimum de Pareto (*Pareto optimum),* [CC02].

Définition 1 (Pareto optimal pour un problème de minimisation): Un vecteur des variables de décision $\boldsymbol{x}^* \in S$ (*S* région réalisable) est un optimum de Pareto si, pour chaque $\boldsymbol{x} \in S$ et $I=\{1,2,...,k\}$, soit $\forall_{i \in I}(\boldsymbol{f}_i(\boldsymbol{x}) = \boldsymbol{f}_i(\boldsymbol{x}^*))$ ou bien, il existe au moins un $i \in I$ tel que: $\boldsymbol{f}_i(\boldsymbol{x}) > \boldsymbol{f}_i(\boldsymbol{x}^*)$.

Au lieu d'une unique solution, l'optimisation multi objectif donne lieu à un ensemble de solutions optimales. Toute solution de cet ensemble est optimale dans le sens qu'aucune amélioration ne peut être faite sur un critère de cette solution sans dégrader au moins la valeur d'un autre critère. Ces solutions optimales forment l'ensemble des solutions Pareto optimales.

Définition 2 (La dominance) : Une solution $\boldsymbol{A}$ domine une solution $\boldsymbol{B}$ pour un problème de minimisation si et seulement si :

$\forall\, \boldsymbol{i} \in \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{k}\}: \boldsymbol{f_i(A)} \leq \boldsymbol{f_i(B)} \boldsymbol{et}\ \exists \boldsymbol{j} \in \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{k}\}: \boldsymbol{f_j(A)} < \boldsymbol{f_j(B)}$

Si la solution $\boldsymbol{A}$ domine la solution $\boldsymbol{B}$, on dit que $\boldsymbol{B}$ est dominée par $\boldsymbol{A}$ ou bien $\boldsymbol{A}$ est non dominée par $\boldsymbol{B}$.

Les solutions **Pareto optimales** sont connues sous le nom de solutions **non dominées**. La représentation de ces solutions non dominées dans l'espace des objectifs est appelée **le front de Pareto.**

La figure 3.1 montre l'exemple d'un front de Pareto pour le problème de minimisation de deux objectifs. Les points en blanc représentent le front de Pareto.

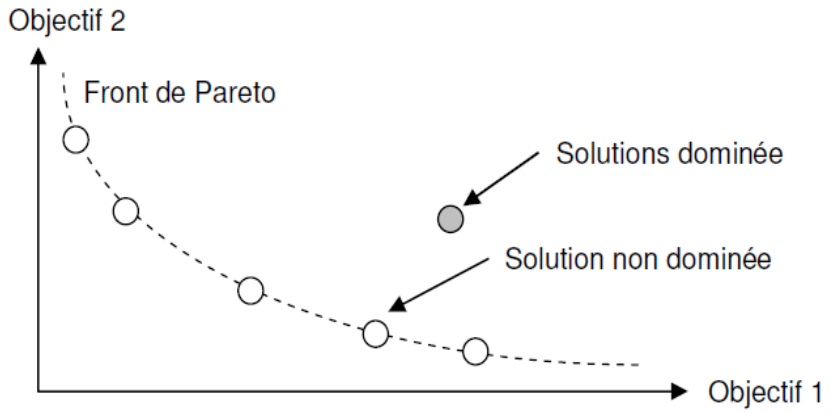

Figure 3.1. Exemple d'un front de Pareto.

3.4.2 Méthodes d'optimisation multi objectif

Les problèmes d'optimisation combinatoire multi objectifs sont pour la plupart NP-difficiles. Leur taille change d'un problème à un autre, en fonction de cette taille deux classes de méthodes ont été distinguées : Les algorithmes exacts pour les petits problèmes et les heuristiques pour résoudre les problèmes de grande taille et/ou les problèmes avec plus de deux objectifs.

La figure 3.2 illustre la classification de quelques méthodes dédiées à l'optimisation combinatoire multi objectif.

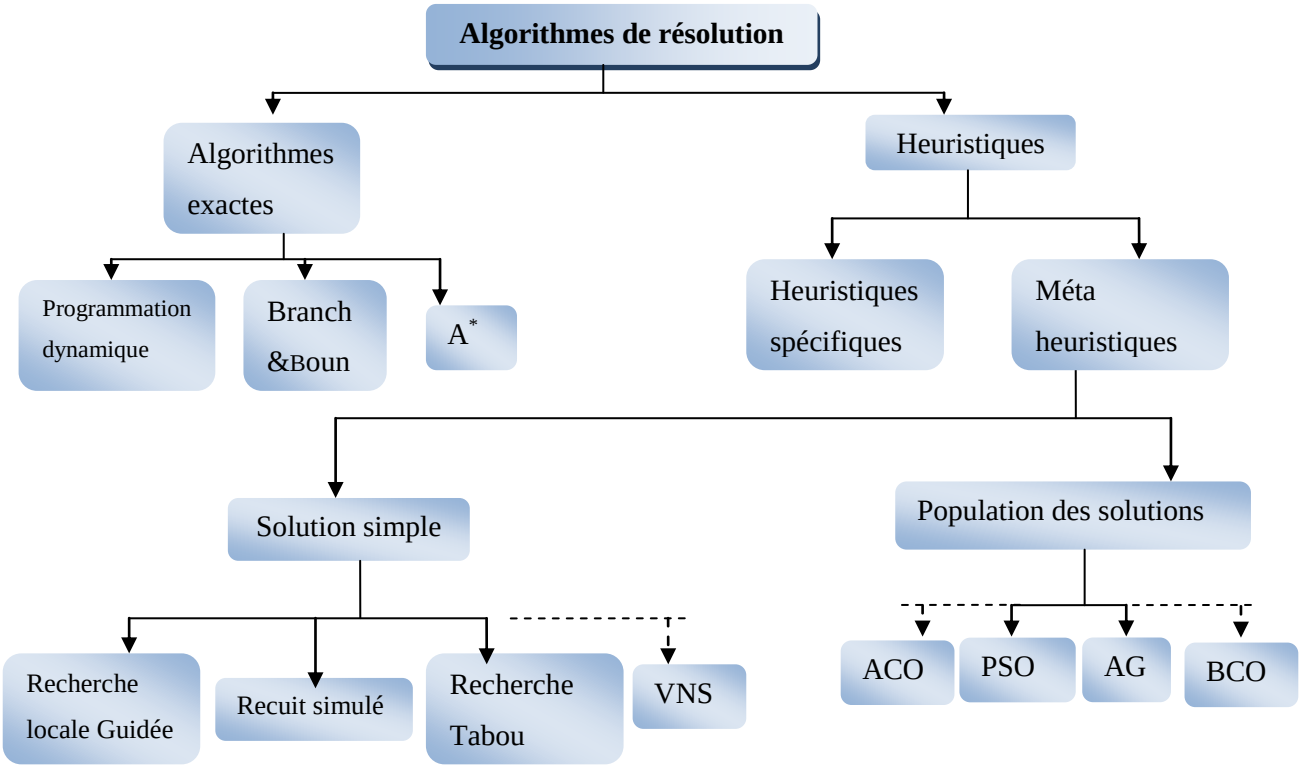

Figure 3.2. Classification des méthodes d'optimisation multi objectif.

3.4.2.1 Méthodes exactes

Ces méthodes ont été proposées pour résoudre les problèmes de petite taille et les petits problèmes à deux objectifs (problèmes bi-objectifs). Parmi ces méthodes on peut citer la méthode par séparation et évaluation (Branch And Bound) [BEN08], l'algorithme A* [SW91] et la programmation dynamique [CMM90].

Une autre méthode intéressante en deux phases a été proposée par B. Ulungu et J.Teghem pour la recherche du front Pareto de problèmes bi objectifs [UT95]. La première phase cherche l'ensemble des solutions Pareto supportées, alors que la deuxième cherche de façon indépendante les solutions non supportées situées entre tous les couples de solutions supportées adjacentes. Cette méthode a été utilisée efficacement sur des problèmes tels que l'affectation ou le sac à dos bi objectif. Cette méthode à ensuite été améliorée afin d'obtenir des fronts complets de façon plus efficace [PGE04].

Les méthodes qu'on vient de citer perdent leur efficacité lorsque la taille du problème et le nombre d'objectifs augmentent. Lorsque c'est le cas, ces problèmes font appel à des méthodes heuristiques.

3.4.2.2 Méthodes heuristiques

Les méthodes heuristiques ne garantissent pas de trouver de manière exacte tout l'ensemble des solutions Pareto, mais une approximation, aussi bonne que possible. Parmi ces dernières nous citons les algorithmes évolutionnaires, la recherche tabou, le recuit simulé, etc. Dans les chapitres suivants de ce mémoire, on s'intéressera à la classe des méthodes heuristiques dites méta-heuristiques pour résoudre notre problème d'optimisation bi objectif, principalement les algorithmes génétiques et les systèmes immunitaires artificiels.

Les méthodes d'optimisation multi objectif (exactes ou heuristiques) peuvent être classées en trois catégories :

– **Approches scalaires, basées sur la transformation du problème en un problème mono objectif :** une classe d'approches basées sur l'agrégation qui combinent les différentes fonctions objectifs du problème en une seule fonction objectif.

– **Approche non-Pareto** : les méthodes de cette approche utilisent des operateurs de recherche de la meilleure solution qui traitent séparément les différents objectifs.

– **Approche Pareto** utilise directement la notion d'optimalité Pareto et la notion de non dominance dans leur processus de recherche et de sélection. Dans ce mémoire, nous nous intéressons à cette catégorie de résolution.

Nous allons dans la section suivante, décrire les principales approches scalaires.

3.4.3 Les approches scalaires

Dans cette section, nous décrivons quelques méthodes traditionnellement utilisées pour la résolution des problèmes d'optimisation multi objectifs. Toutes ces approches passent par la transformation du problème initial en un problème d'optimisation mono objectif. Nous détaillerons, dans ce qui suit, les trois méthodes les plus connues et utilisées : l'optimisation par agrégation pondérée des objectifs, l'approche par ε-contraintes et la méthode Min-Max.

3.4.3.1 La méthode de la somme pondérée

Cette méthode transforme le problème multi-objectif en un problème mono objectif en agrégeant les différents objectifs à optimiser sous forme d'une somme pondérée :

$$(P_w)\begin{cases} Min \sum_{i=1}^{i=m} w_i . f_i(\vec{x}) \\ \vec{x} \in S \end{cases}$$

Les w_i sont des poids vérifiant : $w_i > 0$ et ils peuvent être normalisés tel que : $\sum_{i=1}^{i=m} w_i = 1$.

Ils reflètent l'importance accordée par le décideur aux différents critères. La résolution d'un problème pour un vecteur de poids donnée ne permet de générer que quelques solutions Pareto optimales. Donc il faut effectuer plusieurs exécutions en changeant à chaque fois le vecteur des poids si on veut obtenir un ensemble plus grand de points Pareto optimales. L'avantage de cette méthode est sa simplicité de mise en œuvre et son pouvoir de réutilisation des techniques classiques d'optimisation mono objectif [BER09]. Cependant elles présentent certains désavantages tels que la sensibilité à un changement d'échelle, la compensation entre critères, la difficulté de choix des valeurs des poids pour l'obtention d'un front bien réparti et l'incapacité à découvrir l'intégralité du front Pareto si celui-ci est concave[BER01]. S'il est possible de trouver des remèdes aux deux premières critiques, l'inconvénient principal de cette méthode est son impossibilité à trouver un vecteur des poids *w* pour générer les points se trouvant dans la zone concave du front Pareto. La Figure 2.3 illustre ce cas de figure pour un problème à deux objectifs. Supposons que le front Pareto soit la courbe ABC. Quelques soient les valeurs positives attribuées aux poids *w1* et *w2*, la méthode de la somme pondérée ne peut découvrir que les points *A* ou *C* en fonction de la pente $^{-w_1}/_{w_2}$de la famille de droites $y = w_1 . f_1(\vec{x}) + w_2 . f_2(\vec{x})$ (ou encore $f_2(\vec{x}) = {}^{-w_1}/_{w_2} f_1(\vec{x}) + \frac{y}{w_2}$). Il n'existe pas de vecteur *w* permettant de découvrir le point *B* [ZIT99].

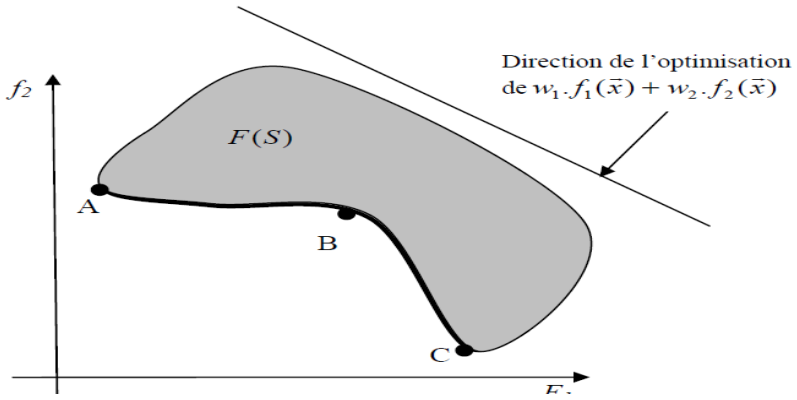

Figure 3.3. Interprétation graphique de non découverte des zones non convexes par la méthode agrégative **[ZIT99].**

3.4.3.2 L'approche ε-contrainte

Dans cette approche, une seule fonction objectif $f_i(\vec{x})$ jugée la plus importante est optimisée tandis que les autres fonctions objectif sont sujettes à des contraintes (Convertir *p-1* des *p* objectifs du problème en contraintes). Le problème (PMO) est alors reformulé comme suit :

$$(P_\varepsilon)\begin{cases} Min\ f_i(\vec{x}) \\ tel\ que\ f_1(\vec{x}) \le \varepsilon_1 \\ \vdots \\ f_{i-1}(\vec{x}) \le \varepsilon_{i-1} \\ f_{i+1}(\vec{x}) \le \varepsilon_{i+1} \\ \vdots \\ f_p(\vec{x}) \le \varepsilon_p \\ \vec{x} \in S \end{cases}$$

L'ensemble Pareto optimal peut alors être obtenu en faisant varier le vecteur des paramètres ε qu'il faut choisir de façon judicieuse. Ceci requiert une bonne connaissance *a priori* du problème. Il est ainsi possible de générer l'ensemble Pareto optimal en n'ayant à résoudre qu'un ensemble de problèmes d'optimisation mono objectif. Il est à noter que cette méthode est capable de générer les solutions non supportées. Elle peut cependant être coûteuse en temps de calcul et la méthode d'optimisation peut être difficile à programmer à cause du nombre de contraintes additionnelles.

3.4.3.3 L'Approche Min-Max

Cette méthode minimise le maximum de l'écart relatif entre le $i^{ème}$ objectif et son *but* T_i qui est un point de référence associé par le décideur. Le programme (*PMO*) est alors transformé comme suit :

$$(MM)\begin{cases} Minimiser\ max_{i\in[1..m]}\left(\dfrac{f_i(\vec{x})-T_i}{T_i}\right) \\ avec \quad \vec{x} \in S \end{cases}$$

Plusieurs variantes de la méthode peuvent être trouvées dans la littérature et notamment dans [CC98]. Les méthodes de résolution utilisant cette approche utilisent souvent le point *idéal* comme point de référence.

3.4.3.4 Discussion

Ce qui rend les méthodes traditionnelles attractives et la raison pour laquelle elles sont populaires peut être attribué au fait que des algorithmes éprouvés pour les problèmes d'optimisation mono objectif peuvent être employés avec les problèmes d'optimisation multi

objectifs. Mais ces méthodes ont aussi des inconvénients. Certaines ne peuvent traiter complètement des problèmes non convexes et sont donc très sensibles à la forme (convexité, discontinuité) du front Pareto [DPAM02]. Un autre inconvénient important est qu'il faille relancer plusieurs fois les algorithmes de résolution avec des valeurs différentes pour certains paramètres (vecteur de poids ou vecteur des paramètres ε par exemple) pour obtenir plusieurs points distincts de la surface de compromis. Ces méthodes nécessitent aussi souvent une bonne connaissance du problème à priori, notamment pour fixer les vecteurs de poids ou les points de référence.

Récemment les métaheuristiques, notamment les algorithmes évolutionnaires, ont permis la réalisation de méthodes de résolution dites Pareto très performantes. Elles permettent de déterminer en une seule exécution une approximation de l'intégralité du front Pareto, et ceci même si les problèmes sont non convexes.

Chapitre 4

OPTIMISATION PAR LES ALGORITHMES GENETIQUES

Introduction

Les Algorithmes Génétiques (AGs) sont certainement la branche des algorithmes évolutionnaires la plus connue et la plus utilisée. Ils ont été créés par analogie avec des phénomènes naturels. Dans ce cas, il s'agit de simuler l'évolution naturelle d'organismes (individus), génération après génération, en respectant des phénomènes d'hérédité et une loi de survie. Dans une population d'individus, ce sont en général les plus forts, c'est à dire les mieux adaptés au milieu, qui survivent et donnent une descendance. Par ailleurs, on suppose que les qualités et les défauts peuvent être hérités des parents de manière stochastique. De tels algorithmes furent développés dés 1950 par des biologistes qui utilisaient des ordinateurs pour simuler l'évolution des organismes. Vers la fin des années 60, John Holland [HOL75] et son équipe, relayés plus tard par d'autres chercheurs [GOL89] et [DAV91], ont adapté ces algorithmes pour la recherche de solutions de problèmes d'optimisation, en développant une analogie entre un individu dans une population et une solution d'un problème dans un ensemble de solutions. Leurs champs d'application sont très vastes vu la simplicité de leurs mécanismes, la facilité de leur mise en application et leur efficacité même pour des problèmes complexes. Récemment, les techniques d'optimisation multi objectifs à l'aide d'algorithmes évolutionnaires, et plus particulièrement d'algorithmes génétiques, suscitent de plus en plus d'intérêt auprès des chercheurs notamment à cause de leur faculté à exploiter de vastes espaces de recherche et à générer des compromis multiples en une seule étape d'optimisation.

Après une description générale du principe de l'algorithme génétique dans la section 4.1, la section 4.2 présentera les différentes phases de cet algorithme. Les algorithmes génétiques sont ensuite décrits comme une technique récente d'optimisation multi objectifs qui possède des caractéristiques intéressantes pour résoudre un Problème Multi Objectif (PMO), La section 4.4 présentera une mise en œuvre d'un algorithme génétique pour l'ordonnancement conjoint production et maintenance dans un Flow Shop Hybride, nous terminons par une série de tests pour évaluer la méthode de résolution proposée.

4.1 CONCEPTS DE BASE D'UN ALGORITHME GENETIQUE

Le but des algorithmes génétiques est d'optimiser une fonction prédéfinie, appelée fonction objectif, ou fitness. Ils travaillent sur un ensemble de solutions candidates, appelé "population" d'individus ou chromosomes (on utilisera indifféremment individu ou chromosome). Ces derniers sont constitués d'un ensemble d'éléments, appelés "gènes", qui peuvent prendre plusieurs valeurs, appelées "allèles". Un chromosome est une représentation ou un codage d'une solution du problème donné. Une première population est choisie soit aléatoirement, soit par des heuristiques ou par des méthodes spécifiques au problème, soit encore par mélange de solutions aléatoires et heuristiques. Cette population doit être suffisamment diversifiée pour que l'algorithme ne reste pas bloqué dans un optimum local. C'est ce qui se produit lorsque trop d'individus sont semblables. Les algorithmes génétiques génèrent de nouveaux individus, de telle sorte qu'ils soient plus performants que leurs prédécesseurs. Le processus d'amélioration des individus s'effectue par utilisation d'opérateurs génétiques, qui sont la sélection, le croisement et la mutation.

Pour mettre en œuvre un algorithme génétique, il est nécessaire de disposer :

– une représentation génétique du problème, c'est-à-dire un codage de solutions utilisé sous la forme de chromosomes.

– un mécanisme de génération de la population initiale. Ce mécanisme est indispensable pour construire une population d'individus non homogène.

– une fonction qui permet d'évaluer l'adaptation d'un chromosome à son environnement, ce qui offre la possibilité de comparer des individus. Cette fonction est construite à partir du critère que l'on désire optimiser. L'application de cette fonction à un élément de la population donne sa fitness.

– un mode de sélection des chromosomes à reproduire. Cette sélection est basée sur la reproduction et sur le codage génétique, qui stocke les informations décrivant l'individu sous forme de gènes.

– d'opérateurs de croisement et de mutation permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche.

– de paramètres qu'utilise l'algorithme : taille de la population, probabilité de croisement et de mutation et nombre total de générations.

4.2 LES ETAPES D'ALGORITHME GENETIQUE

Un algorithme génétique commence par la génération d'une population initiale de P_{size} individus, pour lesquels nous calculons leurs fitness et nous sélectionnons les individus par une méthode de sélection. Ces individus seront manipulés par un opérateur de croisement qui les choisit selon une probabilité P_{cross}. Leurs résultats peuvent être mutés par un opérateur de mutation avec une probabilité de mutation P_{mut}. Les phases de sélection et de recombinaison (croisement et mutation) permettent de générer une nouvelle population d'individus, qui ont de bonnes chances d'être plus forts que ceux de la génération précédente. Les individus issus de la phase de recombinaison seront insérés par une méthode d'insertion dans la nouvelle population, dont nous évaluons la valeur de la fonction objectif de chacun de ses individus. De génération en génération, la force des individus de la population augmente et un test d'arrêt sera appliqué. La figure 4.1 présente un schéma de fonctionnement général d'un algorithme génétique.

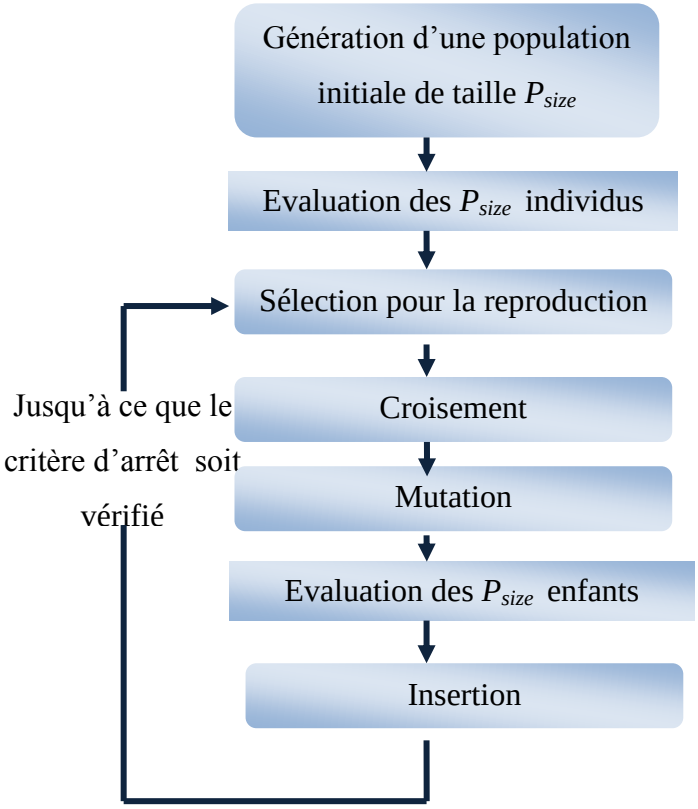

Figure 4.1. Fonctionnement général d'un algorithme génétique.

4.3 LES ALGORITHMES GENETIQUES ET L'OPTIMISATION MULTI OBJECTIF

Les algorithmes génétiques sont très bien adaptés au traitement d'un problème d'optimisation multi objectifs. En témoigne le nombre important d'articles qui ont été publiés sur ce sujet. De plus, ce domaine est très dynamique et ne cesse de se développer.

Dans la plupart des AGs d'optimisation multi objectif développés, il s'agira de satisfaire les deux points suivants [DEB99**]**: 1) trouver des solutions aussi proches que possible des vraies solutions Pareto-optimales, c'est-à-dire converger le plus possible vers le front de Pareto, et 2) trouver un ensemble de solutions très variées, tout le long du front.

Le tout premier algorithme évolutionnaire d'optimisation multi objectifs s'appelle *VEGA* (*Vector Evaluated Genetic Algorithm*, AGEV: Algorithme Génétique à Évaluation Vectorielle) et a été présenté par Schaffer en 1985[CC01]. L'avantage de cet algorithme est qu'il est facile à implémenter mais son inconvénient majeur est qu'il a tendance à générer des solutions qui excellent dans un seul objectif, sans tenir compte des autres objectifs.

Depuis *VEGA*, un nombre considérable d'AG d'optimisation multi objectifs ont été proposés : *NPGA* [HNG94], *NPGA* 2 [EMH01], *NSGA* [SD94], *NSGA-II* [DPAM02], *SPEA2* [ZLT02], les algorithmes *micro-GA* qui réfèrent à des algorithmes avec de petites populations avec réinitialisation.

Les algorithmes génétiques pour l'optimisation multi objectif permettent non seulement l'implémentation d'approches Pareto mais aussi la mise en œuvre des approches classiques (somme pondérée, …) ou non Pareto (VEGA,…).

Dans cette section, on va parler sur les techniques ajoutées au principe de fonctionnement des algorithmes génétiques multi objectifs. Ces techniques essaient de garder les meilleures solutions dans toutes les populations pour ne pas les perdre (l'élitisme), ainsi que de mieux maintenir la diversité de la population pour garder la convergence de l'algorithme et l'exploration de tout l'espace de recherche. Ensuite on expose le principe du *NSGA*-II (*Non Dominated Sorting Genetic Algorithm*-II) qui utilise une approche Pareto élitiste et qui est considéré comme plus efficace que le NSGA, dans la section suivante.

4.3.1 Mécanisme de sélection Pareto (Ranking)

La méthode de Ranking consiste à classer les individus de la population en leur attribuant un rang en se basant sur le concept de la non-dominance, faisant apparaitre la notion de front. Elle a été utilisée dans les AGs pour la résolution de plusieurs PMO. Cette méthode est un

facteur de convergence. Elle permet d'évaluer une solution par rapport à tout l'ensemble de la population. La valeur d'adaptation est alors attribuée à chaque individu en se basant sur son rang. Plusieurs approches de ranking ont été utilisées dans la littérature. Citons par exemple la technique de ranking introduite dans [GOL89] ensuite reprise et implémentée dans NSGA [SD94] et NSGA-II [DPAM02]. Le rang d'un individu est calculé comme suit :

Tous les individus non dominés de la population possèdent le rang 1. Ces individus sont ensuite enlevés de la population et les nouveaux individus non dominés sont identifiés et on leur attribue le rang 2 et ils sont retirés à leur tour de la population. Ce processus est réitéré jusqu'à ce que tous les individus de la population aient un rang.

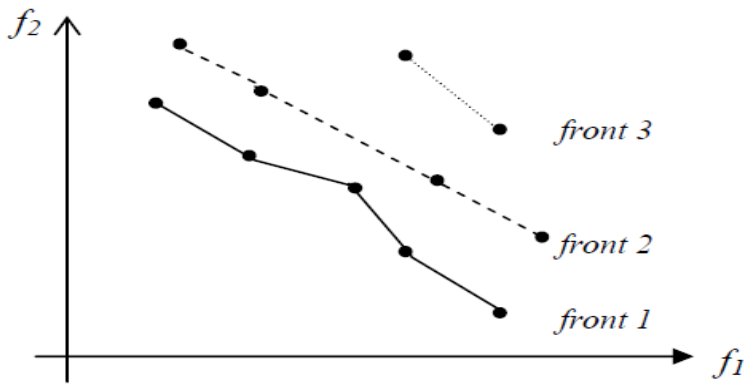

Figure 4.2. Illustration de la méthode de *ranking* [DPAM02].

4.3.2 Maintenir la diversité !

Les méthodes de ranking tendent à favoriser la convergence vers le front Pareto optimal. Cependant, ces méthodes ne sont pas capables de garantir que les solutions soient uniformément réparties dans le sous-espace des solutions Pareto optimales. Pour maintenir une diversité dans la population, les méthodes de Ranking doivent être utilisées en conjonction avec les techniques de maintient de la diversité.

Plusieurs approches visant à maintenir la diversité dans la population ont été proposées dans la littérature : Sharing, Crowding, réinitialisation.

Le sharing [BAR03]**:**

Le sharing consiste à modifier la valeur de coût d'un individu (calculée uniquement à partir de la fonction objective du problème). C'est cette nouvelle valeur qui sera utilisée comme valeur d'adaptation par l'opérateur de sélection. Cette technique, introduite dans [GR87], est largement utilisée aujourd'hui.

Pour éviter qu'un trop grand nombre d'individus ne se concentrent autour d'un même point, il faut pénaliser la valeur d'adaptation en fonction du nombre d'individus au voisinage du regroupement : plus les individus sont regroupés, plus leur valeur d'adaptation est faible, et des individus proches les uns des autres doivent partager leur valeur d'adaptation. Dans la pratique, on estime ce taux de concentration en ouvrant un domaine autour d'un individu, puis on calcule les distances entre les individus contenus dans ce domaine.

Pour déterminer les bornes du domaine ouvert autour de l'individu choisi, on définit une distance maximale, appelée σ_{share} , au delà de laquelle les individus ne seront plus considérés comme faisant parti du domaine ouvert. La distance séparant deux individus i et j est calculée grâce à la fonction *d(i;j).* La valeur d'adaptation *F(i)* d'un individu $i \in P$ (population) est égale à son coût *F'(i)* divisé par sa valeur de niche:

$$\boldsymbol{F(i) = \frac{F'(i)}{\sum_{j \in P} Sh(d(i,j))}}$$

Où la fonction *Sh* est définie comme suit :

$$\boldsymbol{Sh\big(d(i,j)\big) = \begin{cases} 1 - \left(\dfrac{d(i,j)}{\sigma_{Share}}\right)^2 & si\ d(i,j) < \sigma_{Share} \\ 0 & sinon \end{cases}}$$

La fonction *d(i;j)* de calcul de distance peut être définie dans l'espace de recherche, par exemple à l'aide d'une distance de Hamming, ou dans l'espace objectif. Ce choix dépend souvent du problème, car le maintient de la diversité dans l'espace objectif, bien qu'il soit souvent plus simple à réaliser, n'assure pas forcément le maintien de la diversité dans l'espace de recherche.

La réinitialisation :

La réinitialisation est une technique largement utilisée par toutes les métaheuristiques, les algorithmes génétiques n'échappant pas à la règle. Lors d'approches par population, elle consiste à réinitialiser un certain nombre d'individus de la population, par exemple de manière aléatoire. En introduisant de manière régulière certains individus générés aléatoirement, de nouvelles zones de l'espace de recherche, peut-être inexplorées jusqu'ici, peuvent être découvertes. [BAR03]

Le crowding :

L'approche par crowding consiste à déterminer un représentant par niche découverte. A la différence du « sharing », où tous les individus sont susceptibles d'être sélectionnés et de participer aux phases de croisement, mutation et sélection, avec le crowding , seuls les

représentants participeront aux différentes étapes de l'algorithme. Le crowding fut introduit par De Jong [DEJ75] et fut adapté notamment à travers les travaux de Blickle [SD94].

4.3.3 L'élitisme

L'élitisme consiste à conserver les meilleures solutions lors du passage de la génération courante à la prochaine génération. Conserver ces solutions pour les générations futures permet d'améliorer les performances des algorithmes sur certains problèmes. Une des premières implémentations de ce mécanisme dans un algorithme génétique est présentée dans [DEJ75]. En optimisation mono objectif, on a pour habitude de sauvegardes la meilleure solution obtenue au cours du processus d'optimisation. En optimisation multi objectifs, la meilleure solution n'est plus un unique individu, mais tout un ensemble dont la taille peut aller jusqu'à dépasser la taille maximale de la population. Deux adaptations du mécanisme élitiste sont considérées : la première approche regroupe les algorithmes, fondés sur les travaux de De Jong, qui conservent pour les générations futures les *k* meilleurs individus [DPAM02]. Les approches récentes tendent à utiliser une population externe d'individus (archive) dans laquelle est archivé le meilleur ensemble des points non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et les individus stockés continuent à pouvoir être choisis par l'opérateur de sélection. Ils peuvent ainsi se reproduire et transmettre leurs caractéristiques aux générations suivantes. Actuellement, les algorithmes élitistes obtiennent de meilleurs résultats sur un grand nombre de problèmes multi objectifs [BAR03].

4.3.4 L'algorithme NSGA-II (Non-dominated Sorting Genetic Algorithm-II)

[DPAM02] ont proposé une nouvelle version de l'algorithme *NSGA,* le *NSGA*-II, qui est considéré comme étant plus efficace que son prédécesseur car :

– Il utilise une approche élitiste qui permet de sauvegarder les meilleures solutions trouvées lors des générations précédentes.

– Il utilise une procédure de tri basée sur la non-dominance, plus rapide.

– Il utilise un opérateur de comparaison basé sur un calcul de la distance de *crowding* (voir la section 4.3.4.1).

Dans cet algorithme, une population de parents (P_t) de taille (N) et une population d'enfants (Q_t) de taille (N) sont assemblées pour former une population ($R_t = P_t \cup Q_t$), comme le montre la figure 4.3. Cet assemblage permet d'assurer l'élitisme. La population de taille ($2N$) est

ensuite triée selon un critère de non-dominance (*Ranking*) pour identifier les différents fronts F_1, F_2, etc. Les meilleurs individus vont se retrouver dans le ou les premiers fronts. Une nouvelle population parent (P_{t+1}) est formée en inclut intégralement les meilleurs fronts F_i (c'est à dire en commençant à l'indice 1) tant que ceux-ci ne dépassent pas *N*. Si le nombre d'individus présents dans (P_{t+1}) est inférieur à (*N*), une procédure de Crowding est appliquée sur le premier front suivant, (F_i), non inclus dans (P_{t+1}).

Le but de cet opérateur est d'insérer les (*N* - $|P_{t+1}|$) meilleurs individus qui manquent dans la population (P_{t+1}). Les individus de ce front sont utilisés pour calculer la distance de crowding entre deux solutions voisines.

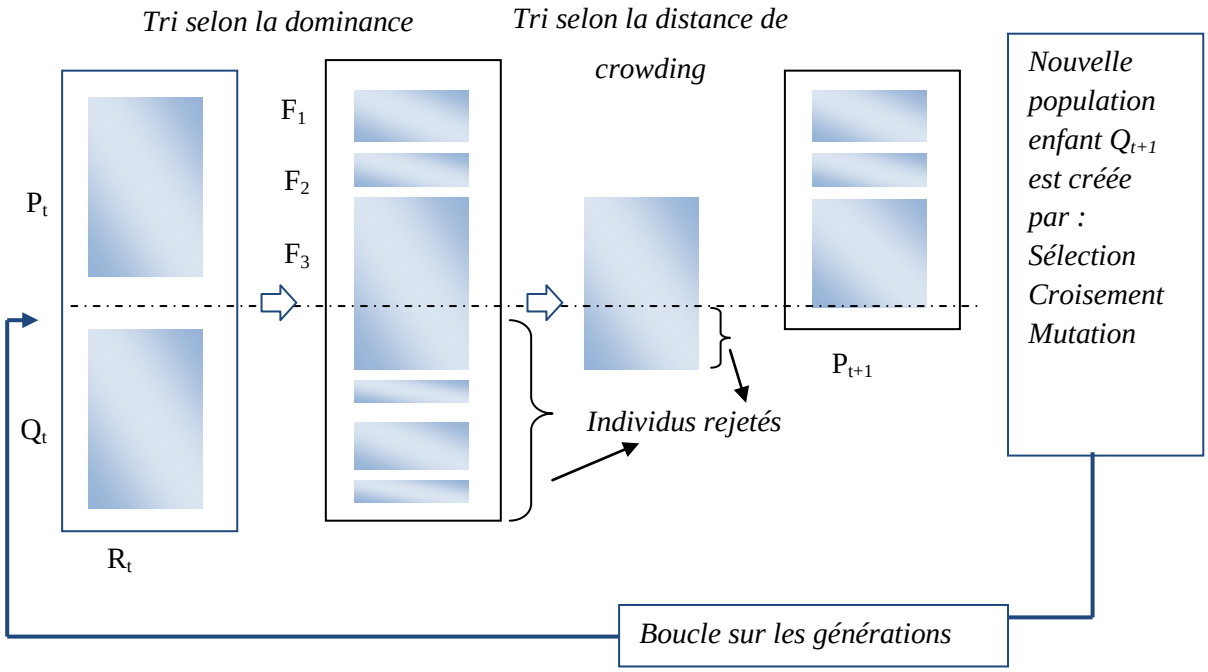

Figure 4.3. Principe de l'algorithme *NSGA*-II [DPAM02].

Une fois que les individus appartenant à la population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1}) est créée par sélection, croisement et mutation. La sélection par tournoi est utilisée mais le critère de sélection est maintenant basé sur l'opérateur de comparaison ($\prec_n$) défini ci-dessus. Le processus continue, d'une génération à la suivante, jusqu'à satisfaire un critère d'arrêt. Le pseudo code de l'algorithme NSGA-II est donné à l'annexe A1.

4.3.4.1 Calcul de la distance de *crowding*

La distance de *crowding* d'une solution (*i*) (ou d'un individu) se calcule en fonction du périmètre formé par les points les plus proches de (*i*) sur chaque objectif. La figure 4.5 montre une représentation à deux dimensions associée à la solution (*i*).

Le calcul de la distance de *crowding* nécessite, avant tout, le tri des solutions selon chaque objectif, dans un ordre ascendant. Ensuite, pour chaque objectif, les individus possédant les valeurs limites (la plus petite et la plus grande valeur de fonction objectif) se voient associés une distance infinie (∞). Pour les autres solutions intermédiaires, on calcule une distance de *crowding* égale à la différence normalisée des valeurs des fonctions objectifs de deux solutions adjacentes. Ce calcul est réalisé pour chaque fonction objectif. La distance de *crowding* d'une solution est calculée en sommant les distances correspondantes à chaque objectif.

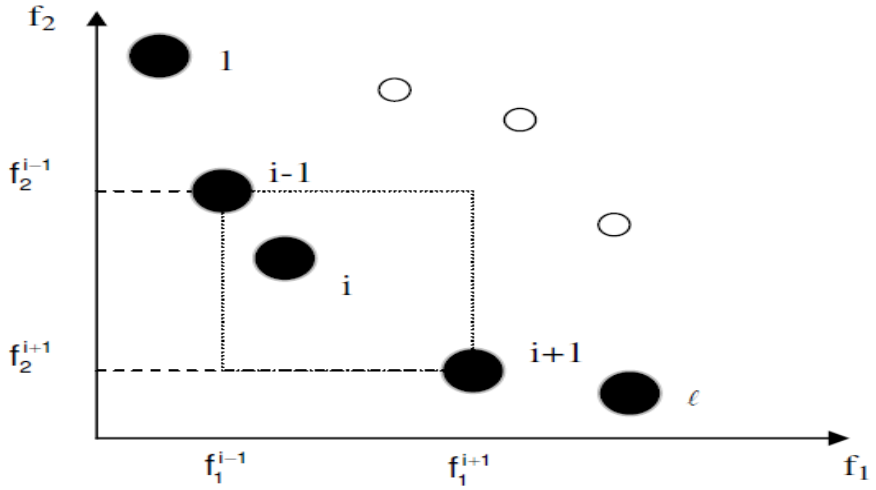

Figure 4.4. Distance de *crowding*, les points noirs sont des solutions appartenant au même front [DPAM02]

4.3.4.2 L'opérateur *crowded-comparison* ($\prec_n$)

Il est utilisé pour guider le processus de sélection comme suit : chaque solution (*i*) de la population est identifiée par son rang (i_{rang}) et la distance de *crowding* ($i_{distance}$). L'opérateur ($\prec_n$) défini ci-dessous permet d'identifier un ordre de préférence entre deux solutions :

$$\boldsymbol{i \prec_n j \ \ si \ (i_{rang} < j_{rang}) \ ou \ \left((i_{rang} = j_{rang}) \ et \ (i_{distance} > j_{distance})\right)}$$

Entre deux solutions de rangs différents, on préfère la solution avec le plus petit rang (ou le plus petit front). Pour deux solutions qui appartiennent au même front, on préfère la solution qui est localisée dans la région où la densité de solutions est moindre, soit l'individu possédant la plus grande valeur de distance de *crowding*.

4.4 ALGORITHME GENETIQUE MULTI OBJECTIF POUR L'ORDONNANCEMENT CONJOINT PRODUCTION / MAINTENANCE : CAS DU FLOW SHOP HYBRIDE

4.4.1 Codage utilisé pour notre algorithme

Chaque individu est codé par une structure à deux champs : la partie production et la partie maintenance. Le codage que nous avons retenu pour la partie production est de longueur égale au nombre total de tâches à réaliser. Ce champ est représenté par une séquence de tâches ***S*** qui représente l'ordre d'exécution des tâches de production. Cette séquence est appliquée seulement au premier étage. Pour les autres étages, c'est-à-dire de l'étage 2 à l'étage *k*, les tâches sont ordonnées selon leurs dates d'achèvement minimales de l'étage précédent. Cela signifie que la règle FIFO (First In First Out) est utilisée pour trouver la séquence des tâches pour les étages suivants. Il est à noter que la séquence des tâches diffère d'un étage à un autre.
La partie maintenance est représentée par une matrice globale ***P*** de dimension ***kxm***, tel que ***k*** est le nombre d'étages et ***m*** est le nombre maximal des machines parallèles identiques dans chaque étage. ***P*** représente les périodes de MP des machines. L'élément ***P[i,j]*** de la matrice ***P*** représente la période de la MP de la machine ***j*** à l'étage ***i***. Les périodes de MP sont générées aléatoirement pour chaque machine ***j*** à l'intérieur de l'intervalle **[*Dj, Cj*]**. ***Dj*** est la durée de traitement de la tâche la plus courte affectée à la machine ***j*** et ***Cj*** est la date de fin d'exécution des tâches sur la machine ***j***.
Pour un problème d'atelier FSH dont la taille est de l'ordre de 10 tâches et 5 étages, avec 2 machines dans chaque étage. Une solution peut être la séquence suivante :

Séquence de production **S**

5	**3**	**9**	**4**	**8**	**2**	**6**	**7**	**10**	**1**

Périodes de MP **P**

4	**12**
37	**49**
23	**16**
79	**21**
23	**84**

P [4,2] = 21 signifie que la période de MP de la deuxième machine au 4ème étage est de 21 unité de temps.

Figure 4.5. Un exemple de Chromosome.

4.4.2 Génération de la population initiale

Dans cette étape, une population initiale doit être générée, où chaque chromosome représente une solution réalisable du problème. La procédure que nous avons utilisée pour générer la population initiale des individus, adaptée à notre problème est une génération aléatoire de P_{size} individus. Ce type de génération est le meilleur choix, car il permet l'hétérogénéité de la population. Chaque chromosome de production est généré aléatoirement parmi les $n!$ séquences possibles et chaque période de MP de chaque chromosome de maintenance est générée aléatoirement dans un intervalle spécifié.

4.4.3 Evaluation des solutions

L'évaluation d'un chromosome comporte deux phases, une phase d'affectation et une phase d'insertion.

– **Affectation des tâches de production:**

L'affectation des tâches sur les machines est faite en appliquant la règle FAM (First available Machine) : nous affectons la tâche à la machine la plus disponible.

– **Insertion de la maintenance:**

Une fois un ordonnancement de production généré, on procède à l'insertion des tâches de maintenance sur ce dernier pour avoir un ordonnancement conjoint de la production et de la maintenance. Les tâches de maintenance sont insérées selon la stratégie rationnelle. Cette stratégie est illustrée comme suit [BAYCM08]:

Soit B_j et C_j les instants de début (*resp*. de fin) d'une tâche de production j. Soit T_e l'instant espéré pour effectuer la tâche de maintenance.

Si $C_j - T_e \geq T_e - B_j$ alors la tâche de maintenance est avancée à la date B_j, sinon elle est retardée à la date C_j.

– **Fonction d'évaluation :**

Deux objectifs sont à minimiser simultanément dans notre étude: la minimisation du makespan ($\boldsymbol{F_1}$) pour l'aspect production et la minimisation de l'indisponibilité du système de production ($\boldsymbol{F_2}$) pour l'aspect maintenance.

Les deux fonctions objectif à minimiser pour $\boldsymbol{k}$ étage, $\boldsymbol{N_e}$ machines parallèles dans chaque étage $\boldsymbol{e}$ sont :

$\boldsymbol{F_1 = C_{max}}$, *qui est le Makespan.*

$\boldsymbol{F_2 = max_{t\in T}\left\{\overline{A_s}(t) = 1 - \prod_{e=1}^{e=k} A_e(t)\right\}}$, *qui est l'indisponibilité du système .*

Où $\boldsymbol{A_e(t)}$ *est la disponibilité de l'étage e* au temps t : $\boldsymbol{A_e(t) = 1 - \prod_{i=1}^{i=N_e}[1 - A_i(t)]}$.

4.4.4 Opérateur de Sélection et Stratégie de reproduction

La sélection par tournoi binaire est utilisée, le critère de sélection est basé sur l'opérateur de comparaison ($\prec_n$) défini dans la section (4.3.4.2). La valeur d'adaptation d'un individu est calculée sur la base de deux mécanismes : Ranking et Crowding.

La méthode de Ranking est un facteur de convergence. Elle consiste d'affecter un rang à une solution en se basant sur le concept de la non-dominance. La méthode de Crowding pour préserver la diversité des solutions dans la phase de réduction de la population et aussi pour la sélection des parents pour la reproduction. Ces deux mécanismes sont utilisés à différents niveaux de l'algorithme pour assurer l'élitisme et la diversité au cours de la reproduction.

4.4.5 Opérateur de Croisement

Pour obtenir de nouveaux individus (enfants) à partir d'une population initiale d'une itération, nous utiliserons l'opérateur de croisement en 2-points avec probabilité P_{cross} pour la partie production et l'opérateur de croisement en 1-points avec probabilité P_{mcross} pour la partie maintenance. Ce croisement s'effectuera comme suit :

Étape 1 : Choisir deux individus de la population actuelle comme parents P1 et P2 pour générer deux enfants E1 et E2.

a. Croisement en 2 points pour la partie production :

Étape 2 : Générer aléatoirement deux positions *k1, k2* ∈ [1, ..., n] tel que *n* est le nombre des opérations à ordonnancer avec *k1* < *k2* et générer une probabilité de croisement α_1.

Étape 3 : Si $\alpha_1 \leq P_{cross}$

Étape 3-1 : Copier toutes les tâches (les tâches qui sont inférieurs à *k1* et supérieurs à *k2*) des parents choisis sur les deux enfants respectivement, ici, nous copions ceux du parent *P1* à l'enfant *E1*, ceux du parent *P2* à l'enfant *E2*.

Étape 3-2 : Compléter la partie restante de l'enfant *E1* par les tâches du parent *P2* et la partie restante de l'enfant *E2* par les tâches du parent *P1* en balayant de gauche à droite et en ne reprenant que les tâches non encore affectées.

Étape 4 : Si $\alpha_1 > P_{cross}$, copier le parent *P1* à l'enfant *E1* et le parent *P2* à l'enfant *E2 (ne pas effectuer de croisement).*

b. Croisement en 1-points pour la partie maintenance :

Pour chaque ligne de la matrice des périodes de la MP des machines:

Étape 2 : Choisir aléatoirement un point de coupure $k \in [1,\ldots,m]$ tel que m est le nombre des machines parallèles et générer une probabilité de croisement α_2.

Étape 3 : Si $\alpha_2 \leq P_{mcross}$

Étape 3-1 : Copier tous les éléments (périodes de MP) de la partie inférieure du parent *P1* à l'enfant *E1*, ceux du parent *P2* à l'enfant *E2*.

Étape 3-2 : Compléter la partie restante de l'enfant *E1* par les éléments du parent *P2* et la partie restante de l'enfant *E2* par les éléments du parent *P1* en balayant de gauche à droite et en ne reprenant que les éléments non encore transmis.

Étape 4 : Si $\alpha_2 > P_{mcross}$, copier le parent *P1* à l'enfant *E1* et le parent *P2* à l'enfant *E2(ne pas effectuer de croisement)*.

L'opérateur de croisement est illustré à la figure 4.7.

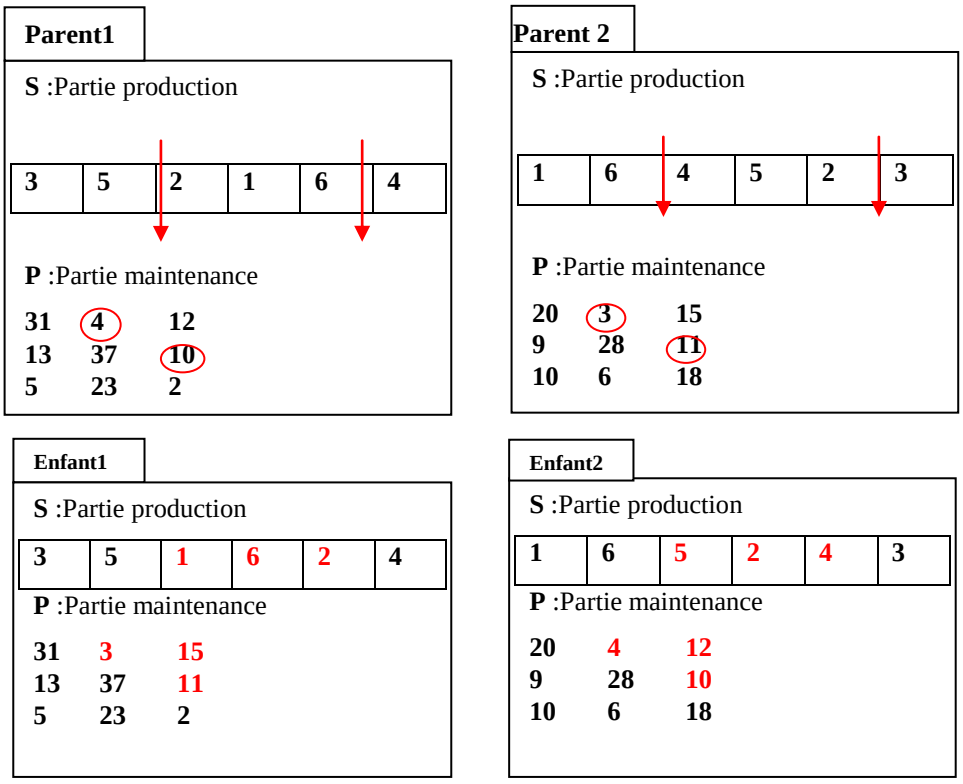

Figure 4.6.L'opérateur de Croisement.

4.4.6 Opérateur de Mutation

Les individus de la population issue du croisement vont ensuite subir un processus de mutation. Pour la partie production, la *mutation- décalage à droite* (shift-mutation) est utilisée comme suit :

Étape 1 : Générer aléatoirement deux gènes (tâches) *s1* , *s2*$\in$ [1, ...,*n*] avec s1 $\neq$ s2 et générer une probabilité de mutation β.

Étape 2 :

– Si *s1* < *s2* insérer la tâche *s1* après la tâche *s2* pour avoir un enfant muté noté *Emut1* (respectivement *Emut2*), ensuite les autres tâches sont décalées à droite.

– Si *s1* > *s2* insérer la tâche *s1* avant la tâche *s2* pour avoir un enfant muté noté *Emut1* (respectivement *Emut2*), ensuite les autres tâches sont décalées à droite.

Étape 3 :

– Si $\beta \leq Pmut$, muter seulement l'enfant *E1* pour avoir un enfant muté *Emut1*.

– Si $\beta > Pmut$, muter seulement l'enfant *E2* pour avoir un enfant muté *Emut2*.

Pour la partie maintenance, la mutation à un point est réalisée selon une loi de Bernoulli de probabilité *Pm* (appelé probabilité de mutation) comme suit :

Pour chaque ligne de la matrice des périodes de la MP des machines:

Étape 1 : Sélectionner aléatoirement une machine et générer une probabilité de mutation *Pm*.

Étape 2:Si $Pm = succès$

– Remplacer la période de MP de la machine sélectionnée (aléatoirement) par une autre période de MP tirée aléatoirement dans l'intervalle spécifiée.

L'opérateur de *mutation* est illustré à la figure 4.7.

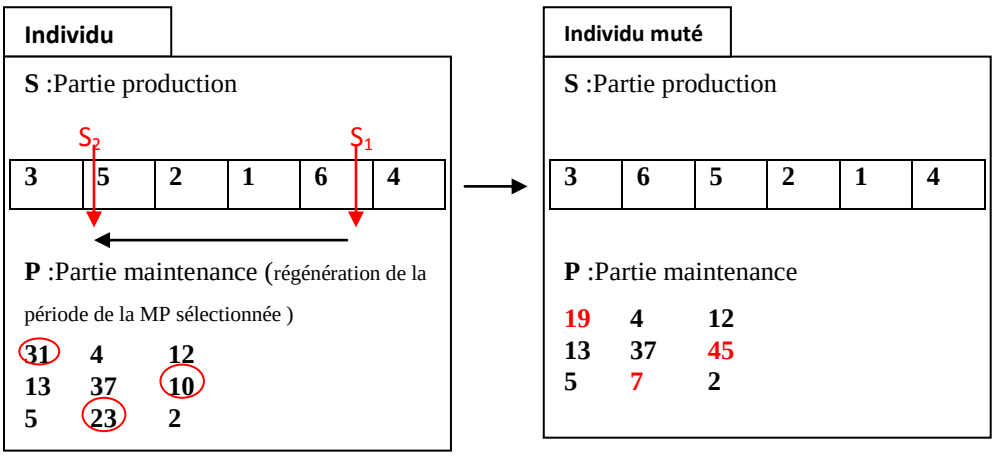

Figure 4.7.L'opérateur de Mutation.

4.4.7 Paramètres de l'algorithme génétique

Pour mettre fin à l'exécution de l'algorithme génétique, un critère d'arrêt est nécessaire. Le choix de ce critère dépend de la complexité du problème traité, du nombre de variables qu'il présente ainsi que des informations connues à priori sur la solution optimale. Si la valeur du critère de la solution optimale est connue d'avance, le critère d'arrêt est évidemment de trouver une solution réalisant cette valeur optimale. De même si on connait une borne inferieure de la solution optimale. L'algorithme s'arrête dés qu'il trouve une solution dont la valeur du critère est la plus proche possible de la borne. Si aucune information n'est connue à priori, le critère d'arrêt peut être un nombre fixe d'itérations. [KAA04]

Pour notre cas, nous avons utilisé un nombre fixe d'itérations comme condition d'arrêt. L'efficacité des algorithmes génétiques dépend étroitement du réglage des différents paramètres à fixer. Ces paramètres sont les probabilités de croisement et de mutation, le nombre des générations et la taille de la population. Pour comparer entre plusieurs méthodes, il est intéressant de tester plusieurs valeurs de la taille en combinaison avec les autres paramètres de l'algorithme.

4.5 EXPERIMENTATIONS ET RESULTATS

Pour évaluer les performances de l'algorithme génétique développé, nous allons le comparer avec un autre algorithme génétique multi objectif basé sur la somme pondérée des critères pour la sélection des parents, noté WSGA (*Weighted Sum Genetic Algorithm*)**.** Ce choix fait référence aux travaux de Berrichi et al. [BAYCM09]. Pour WSGA, les auteurs ont utilisé la somme pondérée des deux fonctions objectif comme fonction de fitness :

$$f(x) = \alpha f_1(x) + \beta f_2(x) \quad \textbf{(4.1)}$$

Où α et β sont les poids non négatifs des deux critères, satisfaisant la contrainte donnée par l'équation (4.2) : $\alpha + \beta = 1$ **(4.2)**

La fonction de fitness sujette à cette contrainte est utilisée pour sélectionner une nouvelle paire de solutions à partir d'une paire de solutions parentes par croisement et mutation. Les valeurs des poids sont aléatoirement spécifiées au moment de la sélection. Une stratégie élitiste est utilisée pour cet algorithme en employant une population secondaire (archive), où les solutions élites (non dominées) sont sauvegardées de génération à génération. Quelques solutions non dominées sont sélectionnées aléatoirement dans la population secondaire et leurs copies sont ajoutées à la population courante. Soit N_{pop} la taille de la population et N_{elite}

le nombre de solutions non dominées (i.e., élites) ajoutée à la population courante. En utilisant ces notations, l'algorithme WSGA est donné à l'annexe A2.

Description des instances

Nos expérimentations ont été effectuées sur un jeu de 13 instances pour le problème d'atelier FSH, dont la taille est de 10 ou 15 tâches et 5 ou 10 étages. Les temps d'exécution sont générés aléatoirement dans l'intervalle [3, 20] [CN00], quel que soit le nombre de machines dans les étages qui varie entre 1 et 3 machine(s). Selon leurs caractéristiques, les instances peuvent être classées en 13 problèmes. Chaque instance est représentée par les lettres : (*j*) indiquant le nombre de tâches, (*c*) le nombre d'étages et *(a, b, c* ou *d*) correspondant à la configuration des machines sur les étages. Nous appelons une configuration de machine une liste de nombres représentant le nombre de machines dans chacun des étages [CN00] (voir tableau)**.**

Types	$j10c5$ et $j15c5$	$j10c10$	$j15c10$
a	22122	2222122222	1222222222
b	12222	1222222222	2222222221
c	33233	3333233333	-
d	33333	-	-

Table 4.1.Différents types de configurations

En se basant sur [CN00], certains des benchmarks sont groupés en tant qu'instances difficiles. Ces instances comprennent les configurations de types *c* et *d* pour les instances de 10 tâches et 5 étages et les instances de 15 tâches et 5 étages (j10c5× et j15c5×). Le reste des instances (de types *a, b* et *j10c10c*) sont identifiées en tant qu'instances faciles.

Paramétrage des algorithmes

En optimisation multi objectif, il n'est pas aisé de comparer deux algorithmes, particulièrement si les deux algorithmes incluent plusieurs paramètres. Nous examinons 27 combinaisons des valeurs des paramètres suivants:

Taille de la population (N_{pop}): 30, 60, 150.

Probabilité de croisement pour la partie production (*pc*): 0.6, 0.8, 1.0.

Probabilité de mutation pour la partie production (*pm*): 0.5, 0.3, 0.1.

Nous avons utilisé un nombre de générations égal à 100 pour les deux algorithmes. Le nombre de solutions élites à chaque génération est égale à 10 (N_{elite} = 10) et la probabilité de croisement (resp. de mutation) concernant la partie maintenance sont prises égales à 0.8 (resp. 0.01) (par bit).

Pour chaque instance nous effectuons 20 exécutions pour chacune des 27 combinaisons des valeurs des paramètres. Donc, 540 ensembles de solutions (fronts Pareto) sont obtenus par chaque algorithme pour chaque problème test. Pour chaque exécution, une nouvelle instance des temps de traitements des tâches est générée.

Mesures de performance

Dans l'optimisation multi objectifs, l'évaluation de la performance est beaucoup plus complexe que dans l'optimisation mono objectif. La mesure des surfaces de compromis est un problème délicat car elle doit représenter la qualité des solutions, la taille du front, la répartition des solutions sur le front, etc.

Pour évaluer la performance de nos algorithmes, nous sommes basés sur trois métriques :

1. Le nombre de solutions non dominées dans chaque front.

2. La métrique d'hyper-volume *H* : cette métrique a été définie dans Zitzler [ZIT99**],** elle permet de mesurer le volume compris sous la courbe formée par les points de l'ensemble à évoluer. Ainsi, dans le cas d'un problème bi objectif, on parle d'hyper-surface et non d'hyper-volume. Dans ce cas, le volume *H* est l'union des rectangles définis par les points (0,0) et ($f_1(x_i)$, $f_2(x_i)$) pour un vecteur de décision x_i (i = 1,n).

Une comparaison équitable entre deux ensembles de solutions avec la métrique *H* requière l'égalité du nombre de solutions des deux ensembles.

3. La métrique *C* de Zitzler [ZIT99]: c'est une métrique relative qui permet de comparer deux surfaces de compromis A et B. Soit A et B deux fronts de solutions non dominées. La mesure de Zitzler *C(A,B)* mesure la proportion de solutions du front B dominées par au moins une solution du front A. Cette équation est définie comme suit:

$$C(A,B) = \frac{|\{b \in B, \exists a \in A : a \succ b\}|}{|B|}$$

La valeur *C(A,B)* = 1 signifie que toutes les solutions de B sont dominées par A. En revanche, *C(A,B)* = 0 signifie qu'aucune des solutions de B n'est dominée par A. Donc, plus proche la valeur de *C (A, B)* à 1, meilleur est le front A par rapport au front B. comme la relation de dominance n'est pas symétrique, *C(B, A)* n'est pas forcément égal à 1- *C(A,B)*. Il est donc

nécessaire de calculer *C(A,B)* et *C(B, A)*. Par conséquent, un front A est meilleur qu'un front *B* si *C (A, B) > C (B, A).*

Analyse des résultats

La table 4.2 résume les meilleures, les moyennes et les pires valeurs du nombre de solutions obtenu, dans l'ensemble des 540 ensembles de solutions. A partir de cette table, on peut voir que les deux méthodes obtiennent de bons résultats dans le cas des moyennes et meilleures valeurs pour ce critère et que s'il n'y pas une large différence dans les pires résultats entre les deux algorithmes, les meilleurs et les moyens résultats obtenus par NSGAII sont bien supérieurs à ceux obtenus par WSGA quelque soit le type d'instance (facile ou difficile).

problème	WSGA			NSGAII		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	19	7.1	2	21	8.9	2
j10c10a	20	8.2	1	24	9.8	1
j15c5a	19	7.4	1	22	10	1
j15c10a	18	7.3	1	28	11	2
j10c5b	15	5.2	1	24	7	1
j10c10b	21	9.8	3	26	13	3
j15c5b	11	5.4	1	18	7.5	1
j15c10b	24	10	3	25	13	4
j10c5c	15	8.3	3	18	9.6	2
j10c10c	16	8.8	3	20	10	1
j15c5c	16	9.7	6	22	12	6
j10c5d	13	8.1	2	17	9.1	2
j15c5d	16	9.6	4	20	11	5

Table 4.2. Meilleures, Moyennes et pires, valeurs du *Nombre de solutions obtenues* sur les 540 ensembles de solutions traités par chaque algorithme pour chaque problème test.

La table 4.3 donne les meilleurs, les moyennes et les plus mauvaises valeurs de la métrique *C* (*C* (WSGA, NSGA-II) et *C* (NSGA-II, WSGA)) obtenues sur les 540 ensembles de solutions. Pour chaque problème test, on peut voir à partir de cette table que NSGAII domine complètement WSGA au moins une fois sur les 27 combinaisons pour chaque problème test alors que WSGA ne domine pas complètement NSGAII sauf dans le cas de deux instances faciles ou WSGA arrive à dominer entièrement NSGA-II. Il est clair que NSGA-II surpasse significativement WSGA en moyenne quelque soit le type d'instance facile ou difficile.

problème	*C* (WSGA, NSGA-II)			*C* (NSGA-II, WSGA)		
	Meill.	**Moy.**	**Mauv.**	**Meill.**	**Moy.**	**Mauv.**
j10c5a	**0.78**	**0.056**	**0**	**1**	**0.89**	**0**
j10c10a	**1**	**0.097**	**0**	**1**	**0.81**	**0**
j15c5a	**0.83**	**0.069**	**0**	**1**	**0.85**	**0**
j15c10a	**0.71**	**0.042**	**0**	**1**	**0.86**	**0**
j10c5b	**1**	**0.14**	**0**	**1**	**0.78**	**0**
j10c10b	**0.79**	**0.073**	**0**	**1**	**0.85**	**0**
j15c5b	**0.75**	**0.016**	**0**	**1**	**0.95**	**0**
j15c10b	**0.7**	**0.069**	**0**	**1**	**0.84**	**0**
j10c5c	**0.75**	**0.12**	**0**	**1**	**0.78**	**0**
j10c10c	**0.8**	**0.1**	**0**	**1**	**0.81**	**0**
j15c5c	**0.67**	**0.093**	**0**	**1**	**0.83**	**0.11**
j10c5d	**0.8**	**0.13**	**0**	**1**	**0.77**	**0**
j15c5d	**0.69**	**0.11**	**0**	**1**	**0.8**	**0.1**

Table 4. 3. Meilleures, Moyennes et Pire valeurs de *la métrique C* sur les 540 ensembles de solutions obtenus par chaque algorithme par chaque problème test.

La table 4.4 présente les meilleurs, les moyennes et les plus mauvaises valeurs de la métrique *H* obtenues sur les 540 ensembles de solutions quand l'égalité du nombre de solutions obtenu est requise pour effectuer les comparaisons. WSGA obtient des résultats similaires à ceux de NSGA-II pour les instances difficiles en termes de la métrique *H*. Toutefois NSGAII surpasse clairement WSGA en moyenne pour les instances faciles comme on peut le voir dans la table4.4.

problème	**WSGA**			**NSGAII**		
	Meill.	**Moy.**	**Mauv.**	**Meill.**	**Moy.**	**Mauv.**
j10c5a	**28**	**37**	**46**	**25**	**36**	**45**
j10c10a	**46**	**58**	**72**	**44**	**56**	**67**
j15c5a	**47**	**54**	**73**	**42**	**52**	**69**
j15c10a	**81**	**97**	**1.1e+002**	**81**	**95**	**1.1e+002**
j10c5b	**29**	**40**	**50**	**28**	**38**	**49**
j10c10b	**58**	**70**	**80**	**54**	**66**	**78**
j15c5b	**49**	**57**	**68**	**43**	**56**	**70**
j15c10b	**79**	**94**	**1.1e+002**	**76**	**90**	**1.1e+002**
j10c5c	**3.2**	**4**	**4.8**	**3.2**	**3.9**	**4.8**
j10c10c	**5.6**	**6.8**	**8**	**5.4**	**6.3**	**7.8**
j15c5c	**4.9**	**5.5**	**6.3**	**4.4**	**5.3**	**6.4**
j10c5d	**0.95**	**1.3**	**1.9**	**0.90**	**1.2**	**1.8**
j15c5d	**1.9**	**2.4**	**3**	**1.7**	**2.3**	**3**

Table 4.4. Meilleures, Moyennes et Pire valeurs de *la métrique H* sur les 540 ensembles de solutions obtenus par chaque algorithme par chaque problème test.

Les meilleures, les moyennes et les pires valeurs de la métrique *C* obtenues par les deux algorithmes sur les 540 ensembles de solutions quand l'égalité du nombre de solutions obtenu est requise pour effectuer les comparaisons sont résumées dans la table 4.5. On peut remarquer que NSGAII surpasse largement WSGA en moyenne, NSGAII domine complètement WSGA au moins une fois sur les 27 combinaisons pour chaque problème test alors que WSGA n'arrive pas à dominer complètement NSGAII.

problème	WSGA			NSGAII		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	0.44	0.062	0	1	0.91	0.5
j10c10a	1	0.071	0	1	0.86	0
j15c5a	0.5	0.088	0	1	0.85	0.33
j15c10a	0.33	0.021	0	1	0.85	0
j10c5b	0.83	0.13	0	1	0.76	0
j10c10b	0.18	0.04	0	1	0.88	0.54
j15c5b	0.29	0.016	0	1	0.96	0.38
j15c10b	0.4	0.047	0	1	0.87	0.43
j10c5c	0.57	0.091	0	1	0.83	0.25
j10c10c	0.5	0.074	0	1	0.86	0.4
j15c5c	0.43	0.081	0	1	0.85	0.3
j10c5d	0.5	0.087	0	1	0.85	0.4
j15c5d	0.27	0.046	0	1	0.88	0.5

Table 3.5. Meilleures, Moyennes et Pire valeurs de *la métrique C* sur les 540 ensembles de solutions obtenues par chaque algorithme par chaque problème test.

Conclusion

Dans ce chapitre nous avons proposé une méthode d'optimisation pour le cas d'atelier flow shop hydride. L'optimisation considérée est bi objectif minimisant le Makspan et l'indisponibilité des machines à la fois afin de trouver un compromis pour le service d'ordonnancement et le service de maintenance. Nos variables de décision sont l'ordre de tâches de production sur les machines et les dates de maintenance préventive de chaque machine. Nous avons alors développé un algorithme génétique basé sur le concept de dominance de Pareto (NSGAII) et un autre basé sur la somme pondérée des critères (WSGA). Des tests ont été menés sur des instances que nous avons générées aléatoirement pour valider nos méthodes. Nos expérimentations ont permis de comparer nos algorithmes entre eux et il en résulte que la meilleure méthode est celle basée sur le concept de dominance de Pareto (NSGAII*)* quelque soit le type d'instance (difficile ou facile).

Chapitre 5

OPTIMISATION MULTI OBJECTIF BASEE SUR LES SYSTEMES IMMUNITAIRES ARTIFICIELS

Introduction

Les mécanismes immunitaires biologiques ont inspiré le développement des systèmes immunitaires artificiels (SIA) dans plusieurs domaines relatifs aux systèmes de production, notamment la détection d'anomalies, le diagnostic d'erreurs, la planification et l'ordonnancement mono objectif et multi objectif.

L'objectif de ce chapitre est d'adapter les principes immunitaires qui font le succès de l'immunité biologique aux problèmes d'ordonnancement conjoint de la production et de la maintenance pour le cas de Flow Shop Hybride. Pour ce faire, nous proposerons des analogies de structure et de mécanismes entre l'immunité biologique et le problème considéré. Nous présenterons deux algorithmes basés sur le principe de la sélection clonale et le mécanisme de la maturation d'affinité utilisé dans une réponse immunitaire pour résoudre ce problème. Vu la difficulté et la nécessité de régler les paramètres de l'algorithme immunitaire, un contrôleur flou sera proposé. Une série de tests pour évaluer les méthodes de résolution proposées clôture ce chapitre.

5.1 LE SYSTEME IMMUNITAIRE ARTIFICIEL

Le système immunitaire biologique constitue une arme contre les intrus qui pénètrent le corps humain, pour cela plusieurs cellules contribuent pour l'élimination de cet intrus nommé *antigène*. Ces cellules participent à ce qu'on appelle *la réponse immunitaire biologique*. Le système immunitaire naturel dispose de deux sortes de défenses : les défenses innées, à large spectre d'action mais d'efficacité moyenne, et les défenses acquises qui sont très spécifiques mais aussi beaucoup plus efficaces. La réponse immunitaire innée a rarement intéressé les informaticiens, à cause de ses capacités statiques, réagissant à l'infection sans pour autant apprendre à y réponde de manière plus efficace. Par contre, la réponse immunitaire adaptative est d'un grand intérêt dans le domaine informatique, principalement grâce à ses capacités d'apprentissage adaptatif.

Les systèmes immunitaires artificiels (AIS) sont des systèmes informatiques inspirés par les principes et les processus du système immunitaire naturel des vertébrés. Pour une bonne simulation du ce dernier, il est évident de bien comprendre son fonctionnement naturel d'abord, ce qui n'est pas simple, car cette simulation se base aussi sur des concepts

mathématiques et bio-inspirés à la fois. Plusieurs tentatives ont vu le jour par l'effort de plusieurs chercheurs.

Les systèmes immunitaires artificiels ont comme principales propriétés [THSC08]: l'apprentissage et la mémorisation, l'auto organisation, l'adaptation, la reconnaissance, la robustesse et l'évolutivité.

La plupart des chercheurs se sont concentrés sur les mécanismes de l'apprentissage et la mémoire du système immunitaire en étudiant les processus de la sélection clonale et des réseaux immunitaires. Mais également sur le principe de sélection négative pour la génération de détecteurs capables de classer les changements du soi. Le travail de De Castro et Von Zuben [DFV02] sur la sélection clonale est devenu notable en 2002. Le premier livre sur les systèmes immunitaires artificiels a été édité par Dasgupta en 1999.

Deux algorithmes basés sur l'approche immunitaire, inspirés du principe de la sélection clonale et du principe de la maturation d'affinité du système immunitaire naturel ont été développés dans le cadre de ce travail pour résoudre le modèle bi objectif intégré du problème conjoint production-maintenance. Le premier algorithme basé sur la non dominance et le deuxième algorithme basé sur la somme pondérée des critères pour la sélection et le clonage.

5.1.1 Principe de la sélection clonale

Le principe de la sélection clonale peut être résumé comme suit [DFV02] (Figure1.5): Lorsque le corps est exposé à un antigène, les lymphocytes B (cellules du Système Immunitaire Naturel) répondent par la production des Anticorps. Chaque cellule sécrète un type unique d'anticorps spécifique pour l'antigène. Par la liaison à ces anticorps (récepteurs de la cellule) et avec un signal des cellules accessoires T-Helper, l'antigène stimule la cellule B à proliférer (diviser) et se différencier en cellules sécréteurs d'anticorps appelées les cellules Plasma. Le processus de la prolifération (mitosis) génère le clone qui est une cellule ou ensemble de cellules qui sont les cellules filles d'une seule cellule. Le taux de prolifération des cellules est directement proportionnel au degré d'affinité (liaison) avec l'antigène. Les lymphocytes B, en plus de leur différentiation en cellules plasma, elles se différentient en cellules mémoires qui vont être sélectionnées pour produire des anticorps dans le cas d'une attaque par les mêmes antigènes.

Les propriétés principales de la théorie clonale sources d'inspiration de l'algorithme sont :

– Prolifération et différenciation sur stimulation des cellules avec les antigènes.

– Génération de nouveau changement génétique qui donne de nouveaux modèles d'anticorps par le biais d'une mutation somatique (processus appelé maturation d'affinité).

– Mémorisation des lymphocytes ayant une plus grande affinité avec l'antigène.

– Elimination des lymphocytes issus de la prolifération qui présentent une faible affinité avec l'antigène.

La figure 5.1 montre le principe de la sélection clonale.

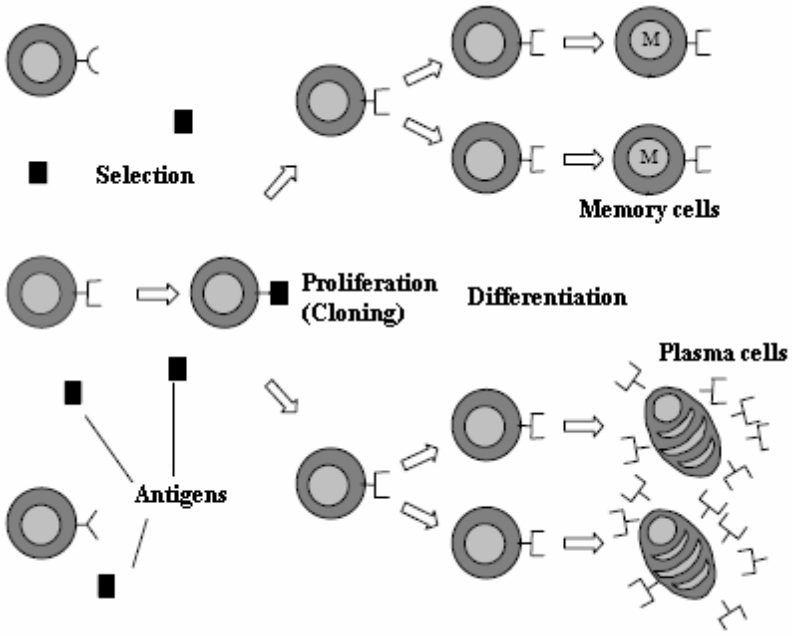

Figure 5.1. Principe de la sélection clonale [DFV02]

5.1.2 Maturation d'affinité

La maturation d'affinité est le processus de mutation et de la sélection de la progéniture qui reconnaît mieux l'antigène. Les deux mécanismes de base de la maturation d'affinité sont l'hyper-mutation et la génération des récepteurs [DFV02].

– *L'hyper-mutation somatique* est le changement aléatoire (mutation) dans la région variable des gènes des molécules de l'anticorps. Ces changements sont des événements mutationnels qui causent des changements structurels de la cellule dans l'objectif d'augmenter l'affinité de l'anticorps avec l'antigène. Le taux de l'hyper-mutation somatique est inversement proportionnel à l'affinité de la cellule : plus haut le degré de l'affinité de l'anticorps avec l'antigène moins le taux de mutation et vice versa. Avec cette stratégie le système immunitaire garde les cellules de la progéniture qui présentent une haute affinité et garantit un taux de mutation pour celles qui présentent une faible affinité pour obtenir une meilleure affinité.

– *La génération des récepteurs* est due à la nature aléatoire du processus de l'hyper-mutation somatique. Un grand nombre d'anticorps obtenus sont pauvres ou non utiles, ces cellules seront éliminées et remplacées par d'autres générées aléatoirement.

Donc, la mutation permet d'explorer les régions locales et la génération des récepteurs permet d'éviter le problème de l'optimum local.

5.1.3 Correspondances des mécanismes

Les acteurs importants entrant dans un système immunitaire artificiel sont les **antigènes**, les **anticorps** et les cellules **B mémoire**.

Système immunitaire naturel **SIN**	Système immunitaire artificiel pour notre problème d'ordonnancement **bi objectif**
Antigène	Fonctions objectifs à optimiser (Cmax, Indisponibilité)
Anticorps	Solutions, individus
Affinité	La valeur d'adaptation d'un individu (**rang**(Ranking) et **distance** (Crowding))
Cellules B mémoire	Individus non dominés de rang =1

Table 5.1. Correspondances des mécanismes.

5.2 L'APPROCHE IMMUNITAIRE MULTI OBJECTIF PROPOSEE

Dans cette section, nous proposerons deux algorithmes multi objectif s'inspirant du principe de la sélection clonale et du principe de maturation d'affinité du système immunitaire naturel, notés **MOIA1** et **MOIA2** (pour *Multi Objective Immune Algorithm*). Les deux algorithmes sont basés sur le concept de non dominance au sens de Pareto. Néanmoins, le premier est basé sur la non dominance pour la sélection et le clonage des anticorps alors que le deuxième est basé sur la somme pondérée des critères.

5.2.1 Algorithme MOIA1 proposé

Cet algorithme utilise deux populations, une population courante P avec N_p anticorps (individus) comme population de clonage et une population secondaire S qui contient les N_s meilleurs anticorps trouvés du début jusqu'à l'itération courante (l'élite), elle est utilisée pour accumuler les solutions non dominées et à partir d'elle que B% d'anticorps sont sélectionnés aléatoirement pour remplacer les B% d'anticorps ayant les plus faibles affinités de la population P, supprimés lors du processus de génération des récepteurs. Les anticorps nés de

l'application du clonage, de la mutation et de la génération des récepteurs aux anticorps de la population P formeront la nouvelle population courante P_{t+1}. La nouvelle population secondaire S_{t+1} est constituée des solutions non dominées se trouvant dans $P_{t+1} \cup S_t$.

Dans cette section, nous donnons le pseudo code de l'algorithme **MOIA1** (Figure 5.2) pour la résolution du modèle bi objectif intégré pour l'optimisation de l'ordonnancement conjoint production- maintenance dans le cas de FSH.

Pseudo code de l'algorithme MOIA1

Entrées :

Gmax : nombre de générations
Popsize : taille de la population d'anticorps
B : pourcentage d'élimination des anticorps
Tmut : nombre de mutation

Sorties : ensemble Pareto optimal

Etape 1 : Initialisation
- créer aléatoirement une population ***P*** initiale des anticorps
- une population secondaire ***S*** initialement vide
- initialiser le nombre des générations gen =0

Etape 2 : **Test d'arrêt de l'Algorithme**
Si gen > Gmax aller à l'étape **9** Sinon aller à l'étape **3**

Etape 3 : incrémenter le nombre de génération gen

Etape 4 : Evaluation
- calculer pour chaque individu de la population d'anticorps ***P*** son affinité selon la procédure du *Ranking* et *crowding*

Etape 5 : Mémorisation
- copier les individus non dominés (de rang = 1) dans la population secondaire ***S***

Etape 6 : Sélection et Clonage
- sélection des anticorps ayant les meilleures affinités dans la population ***P***
- cloner chaque anticorps sélectionné proportionnellement à son affinité
- muter les anticorps clonés avec un degré anti proportionnel à leurs affinités
- placer les clones dans une nouvelle population d'anticorps

Etape 7 : Suppression Clonale (Principe de génération des récepteurs)
- calculer pour chaque individu de la nouvelle population d'anticorps ***P*** son affinité selon la procédure du *Ranking* et *crowding*
- éliminer les B % d'anticorps de plus faibles affinité de la nouvelle population
- sélectionner aléatoirement B% d'anticorps de la population secondaire
- remplacer les anticorps éliminés par ceux sélectionnés (aléatoirement)

Etape 8 : Retour à l'étape **2**

Etape 9 : Approximation du front Pareto optimal
- afficher les solutions non dominés dans la population secondaire comme approximation de l'ensemble Pareto optimale
- fin de l'Algorithme

Figure 5.2. Pseudo code de l'algorithme MOIA1

5.2.2 Mécanismes d'évolution

L'évolution de la population d'anticorps est basée sur les deux principes du système immunitaire naturel : la sélection clonale et la maturation d'affinité.

5.2.2.1 Processus de sélection clonale

Dans le système immunitaire biologique, le clonage signifie qu'un groupe de cellules identiques est généré à partir d'un seul ancêtre commun et uniquement les anticorps avec une forte affinité seront clonés pour attaquer les agents pathogènes. Dans l'algorithme MOIA1, à chaque anticorps de la population de clonage *P* est associé une valeur d'adaptation qui fait référence à sa valeur d'affinité. Cette valeur est calculée sur la base de deux concepts *Ranking* et *Crowding* définis dans les sections 4.3.1 et 4.3.2.

La valeur d'adaptation *f* d'un anticorps *i* de la population de clonage *P* est calculée comme suit : $\boldsymbol{f(i) = Rang(i) + Densité(i)}$ $(\mathbf{5.1})$

Où la valeur de densité est calculée sur la base de l'inverse de la distance de Crowding ***d(i)*** (voir la section 4.3.5.1) : $\boldsymbol{Densité(i) = \frac{1}{d(i)+2}}$ $(\mathbf{5.2})$

La valeur 2 est ajoutée au dénominateur pour éviter la division par zéro et s'assurer que $\boldsymbol{Densité(i)} < 1$

Le nombre de clones q_i pour chaque anticorps *i* de la population *P* est calculé comme suit :

$$\boldsymbol{q_i = N_c \times \frac{fit(i)}{\sum_{j=1}^{|P|} fit(j)}} \quad (\mathbf{5.3})$$

Où N_c est la taille de la population de clones et *fit(i)* représente la valeur d'affinité (ou de fitness) de l'anticorps *i* calculée comme suit: $fit(i) = \frac{1}{f(i)}$ $(\mathbf{5.4})$

Dans l'algorithme, la taille de la population des clones générés est fixée égale à la taille de la population des anticorps. Le nombre de clones générés pour chaque anticorps change selon sa valeur d'affinité. Donc les anticorps avec une grande affinité ont plus de clones dans l'ensemble des clones générés.

5.2.2.2 Processus de maturation d'affinité

Les deux mécanismes de base de la maturation d'affinité sont l'hyper-mutation et la génération des récepteurs.

a. L'hyper-mutation

Chaque clone généré va subir un processus de mutation :

Pour la partie maintenance : la mutation à un point est réalisée selon une loi de Bernoulli de probabilité P_m (appelé probabilité de mutation) comme suit :

Pour chaque ligne de la matrice (pour chaque étage) des périodes de MP des machines:

Étape 1 : Sélectionner aléatoirement une machine et générer une probabilité de mutation Pm;

Étape 2:Si $P_m = succès$

– Remplacer la période de MP de la machine sélectionnée (aléatoirement) par une autre période de MP tirée aléatoirement dans l'intervalle spécifié.

Pour la partie production: une procédure de mutation en deux phases est utilisée où chaque clone généré subit une mutation simple dans la première phase et une mutation large dans la deuxième phase.

- **Mutation simple** (par changement de 2 positions) : soit *s* une séquence, *i* et *j* deux positions dans cette séquence choisies aléatoirement, alors le voisinage de *s* est obtenu par l'interchangement des tâches des deux positions.

- **Mutation large** : soit *s* une séquence des tâches de production. Le voisinage de *s* est obtenu par la mutation de *s*, T_{mut} fois.

Si la nouvelle solution (nouveau anticorps) obtenue (après la mutation de la partie maintenance et la mutation simple de la partie production) domine la solution d'origine (le clone généré pour un anticorps) alors la nouvelle solution obtenue remplace la solution d'origine, sinon la partie production de la nouvelle solution subit la mutation large.

Dans le cas où l'algorithme n'améliore pas la solution après les deux phases de mutation alors il garde la solution d'origine (clone généré).

b. Génération des récepteurs

Après le processus de clonage et de mutation, un pourcentage (B%) d'anticorps de la population sont éliminés et remplacés par d'autres anticorps sélectionnés aléatoirement à partir de la population secondaire.

L'ensemble des clones forme la population des anticorps pour la prochaine génération après les processus de sélection, de mutation et de génération des récepteurs.

5.2.3 Analyse de la complexité algorithmique de l'algorithme MOIA1

La complexité algorithmique d'un algorithme est le nombre d'opérations élémentaires effectuées par ce dernier. Ce nombre s'exprime en fonction de la taille *N* des données. Dans cette section, nous analysons la complexité algorithmique de l'algorithme MOIA1. En

suppose que la taille de la population des anticorps, la taille de la population des clones et la taille de la population secondaire est *N*, le nombre de fonctions objectifs est *M*. En nous basant sur le pseudo code de l'algorithme MOIA1 de la figure 5.2, l'analyse de la complexité est donnée comme suit:

– **Initialisation de la population :**

La procédure d'initialisation de la population est de créer aléatoirement *N* anticorps et d'évaluer leurs fonctions objectifs, elle est de complexité ***O(N) + O(M.N).***

– **Calcul de la fonction d'affinité :**

La valeur d'affinité pour chaque individu de la population d'anticorps est calculée sur la base de deux concepts *Ranking* et *Crowding.* La procédure de *Ranking* partitionne une population d'individus en sous ensembles (fronts) et affecte un rang à chaque front. Les individus non dominés dans la population entière ont un rang égal à un (les meilleurs). Si on enlève temporairement les individus du front 1 alors les individus non dominés dans le reste de la population constituent le front numéro 2 (de rang 2) et ainsi de suite. La complexité temporelle de cette procédure est de $\boldsymbol{O(M.N^2)}$**.** La distance de *Crowding* pour chaque individu est calculée par la procédure de *Crowding* de complexité ***O (M.N log(N))*** [DPAM02].

– **Clonage, hypermutation, génération des récepteurs :**

Les complexités de ces procédures sont de ***O(N).***

En se basant sur l'analyse ci-dessus, dans une seule génération, la complexité au pire des cas de **MOIA1** est de $\boldsymbol{O(M.N^2)}$**.**

Nous notons que les complexités au pire des cas des algorithmes **NSGA-II**, **PAES**, **SPEA**, **SPEA2** sont $\boldsymbol{O(M.N^2)}$, $\boldsymbol{O(M.N)}$, $\boldsymbol{O(M.N^2)}$, $\boldsymbol{O(M.N^3)}$, respectivement.

5.2.4 L'Algorithme MOIA2 proposé

Cet algorithme utilise une population courante *P* avec N_p anticorps (individus) comme population de clonage. Les anticorps nés de l'application du clonage, de la mutation et de la génération des récepteurs aux anticorps de la population *P* formeront la nouvelle population courante P_{t+1}.

Nous avons traité MOIA2 avec les mêmes opérateurs et avantages que MOIA1: même opérateurs de sélection et clonage, même processus de maturation d'affinité, néanmoins leurs fonctions d'évaluation sont différentes. Dans MOIA1, la valeur d'affinité des anticorps est

calculée sur la base de deux concepts *Ranking* et *Crowding* alors que l'algorithme MOIA2 calcule cette valeur sur la base de la somme pondérée des deux fonctions objectifs.

La résolution du problème par la méthode de la somme pondérée consiste à transformer le problème bi objectif en un problème mono objectif en agrégeant les deux critères à optimiser, le Makspan (*F1*) et l'indisponibilité du système (*F2*), sous forme d'une somme pondérée en prenant en compte la fonction d'évaluation suivante :

$$pour\ \alpha \in [\mathbf{0}, \mathbf{1}]$$

$$\boldsymbol{F = \alpha\ \frac{F_1}{BS} + \ (1 - \alpha)F_2} \qquad \textbf{(5.5)}$$

BS : est la valeur maximale du Makspan de tous les anticorps de la population, il est utilisé pour normaliser la fonction objectif *F1*. Les valeurs de *F2* sont déjà dans l'intervalle [0,1].

α : est un paramètre qui va mesurer les contributions respectives de la production et de la maintenance dans la fonction objectif globale *F*, Les valeurs des poids sont aléatoirement spécifiées au moment de la sélection et du clonage pour chaque génération de l'algorithme. Si *N* est le nombre de générations, cela veut dire que *N* directions de recherche sont explorées en une seule exécution de l'algorithme. Toutefois, si les valeurs des poids sont constantes alors la direction de recherche est fixée.

A chaque anticorps est associée une valeur d'adaptation qui fait référence à sa valeur d'affinité. Cette valeur est calculée par une fonction d'affinité suivant l'équation :

$$\textbf{Affinité} = \frac{1}{1+F} \qquad \textbf{(5.6)}$$

Le dénominateur est égal à (*1+ F*) pour éviter une division par zéro, dans le cas où la fonction objectif *F* est nulle. L'algorithme MOIA2 est donné à l'annexe A3.

5.3 IMPLEMENTATION ET TESTS

Dans cette section, nous présentons l'ensemble des tests expérimentaux que nous avons réalisés pour l'évaluation et la validation des méthodes proposées précédemment pour la résolution du problème d'ordonnancement conjoint production /maintenance dans un environnement de type Flow Shop Hybride.

Nous présentons les résultats obtenus en deux volets : le premier concerne la comparaison entre les deux algorithmes immunitaires proposés (**MOIA1 et MOIA2**), et dans le second la comparaison entre l'approche immunitaire implémentée par l'algorithme **MOIA1** et l'approche génétique implémentée par l'algorithme **NSGAII**.

Tous les algorithmes sont codés en langage C++ dans l'environnement Windows et exécutés sur un PC avec un Pentium (R) Dual Core CPU (2. 30 GHz).

Pour effectuer les comparaisons entre les algorithmes, nous avons considéré le même cadre d'expérimentations que dans le chapitre précédent. Le même jeu d'instances pour le problème d'atelier FSH dont la taille est de 10 ou 15 tâches, et 5 ou 10 étages. Les temps d'exécution sont générés aléatoirement dans l'intervalle [3, 20] comme dans [CN00].

5.3.1Comparaison des deux algorithmes immunitaires proposés

Dans cette section, nous comparons les performances des deux algorithmes immunitaires proposés **MOIA1** et **MOIA2.**

Paramètres de test :

Nous examinons 27 combinaisons des valeurs des paramètres suivants :

– Taille de la population (**Npop**): **30, 70, 200.**

– Le pourcentage d'élimination **B** : **10%, 20%, 50%**

– Le nombre de fois de mutations (large) $\mathbf{T_{mut}}$: **2, 4, 7**

Nous avons utilisé un nombre de générations égal à **100**.

Chacun des deux algorithmes est appliqué à chaque problème test **10** fois pour chacune des **27** combinaisons des valeurs des paramètres. Donc, **270** ensembles de solutions (fronts Pareto) sont obtenus par chaque algorithme pour chaque problème test. Pour chaque exécution, une nouvelle instance des temps de traitements des tâches est générée.

Mesures de performance :

Pour évaluer la performance des deux algorithmes immunitaires, nous sommes basés sur les trois métriques utilisées dans le chapitre 4: la métrique d'hyper-volume *H*, la métrique *C* et le nombre de solutions obtenu (le cardinal du front Pareto).

Analyse des résultats

Comme indiqué sur les tableaux (**5.2, 5.3, 5.4, 5.5)** , pour des temps de traitement générés aléatoirement dans l'intervalle [3, 20], MOIA1 s'avère plus efficace qu' MOIA2 en termes de convergence qu'en termes de diversité selon la métrique C (**Table 5.3, Table 5.5)** et la métrique Hypervolume (**Table 5.4)** pour tout type d'instances (facile et difficile). Selon la métrique *nombre de solutions obtenues*, les deux algorithmes obtiennent de bons résultats dans le cas des moyennes et meilleures valeurs pour ce critère, néanmoins MOIA2 surpasse MOIA1 en moyenne (**Table 5.2)**.

problème	MOIA2			MOIA1		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	18	8.25	1	14	5.31	1
j10c10a	17	8.29	1	14	5.23	1
j15c5a	23	10.43	3	19	8.6	1
j15c10a	21	8.6	1	11	4.55	1
j10c5b	18	6.94	1	13	5.01	1
j10c10b	19	9.92	4	14	6.37	1
j15c5b	14	7.45	2	10	4.51	1
j15c10b	23	13.31	6	18	8.71	2
j10c5c	16	8.43	3	13	6.97	1
j10c10c	19	8.77	2	12	5.91	1
j15c5c	18	9.82	5	14	6.51	1
j10c5d	16	8.11	1	11	5.83	1
j15c5d	17	9.04	3	13	6.45	1

Table 5.2. Meilleures, Moyennes et Pire valeurs du *Nombre de solutions obtenues* sur les 270 ensembles de solutions traités par chaque algorithme immunitaire pour chaque problème test.

La table 5.3 donne les meilleurs, les moyennes et les plus mauvaises valeurs de la métrique *C* (*C* (WSIA, MOIA) et *C* (MOIA, WSIA)) obtenues sur les 270 ensembles de solutions.

problème	C (MOIA2, MOIA1)			C (MOIA1, MOIA2)		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	0.71	0.13	0	1	0.76	0
j10c10a	0.67	0.07	0	1	0.71	0
j15c5a	0.67	0.06	0	1	0.72	0.18
j15c10a	0.8	0.07	0	1	0.66	0
j10c5b	1	0.12	0	1	0.71	0
j10c10b	0.75	0.06	0	1	0.73	0.15
j15c5b	0.75	0.15	0	1	0.75	0
j15c10b	0.6	0.09	0	1	0.77	0.17
j10c5c	0.6	0.10	0	1	0.75	0.17
j10c10c	0.71	0.12	0	1	0.75	0.3
j15c5c	0.6	0.10	0	1	0.77	0.29
j10c5d	0.71	0.09	0	1	0.75	0
j15c5d	0.6	0.08	0	1	0.77	0.2

Table 5. 3. Meilleures, Moyennes et Pire valeurs de *la métrique C* sur les 270 ensembles de solutions obtenus par chaque algorithme immunitaire par chaque problème test.

La table 5.4 présente les meilleurs, les moyennes et les plus mauvaises valeurs de la métrique *H* obtenues sur les 270 ensembles de solutions quand l'égalité du nombre de solutions obtenu est requise pour effectuer les comparaisons.

problème	MOIA2			MOIA1		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	29.44	36.80	44.39	26.58	35.39	44.6
j10c10a	50.8	60.41	69.25	48.07	55.85	63.92
j15c5a	47.28	57.64	64.53	46.27	53.95	61.88
j15c10a	85.27	101.14	116.07	84.55	96.68	112.12
j10c5b	31.32	40.1	45.81	30.79	38.82	45.31
j10c10b	55.85	69.01	83.02	53.48	62.17	76.42
j15c5b	44.86	56.96	70.47	45.43	56.38	68.17
j15c10b	76.5	86.31	93.60	75.26	82.35	92.25
j10c5c	3.61	4.14	4.96	3.1	3.92	4.548
j10c10c	5.52	6.7	7.81	5.87	6.34	7.24
j15c5c	5.39	6.03	7.06	4.85	5.4	6.04
j10c5d	1.02	1.37	1.7	0.86	1.1	1.40
j15c5d	2.06	2.42	2.97	1.48	2.01	2.49

Table 5.4. Meilleures, Moyennes et Pire valeurs de *la métrique H* sur les 270 ensembles de solutions obtenus par chaque algorithme immunitaire par chaque problème test.

Les meilleures, les moyennes et les pires valeurs de la métrique *C* obtenues par les deux Algorithmes immunitaires sur les 270 ensembles de solutions quand l'égalité du nombre de solutions obtenu est requise pour effectuer les comparaisons sont résumées dans la table 5.5.

problème	MOIA2			MOIA1		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	0.67	0.23	0	1	0.69	0.25
j10c10a	0.25	0.06	0	1	0.73	0.14
j15c5a	0.33	0.08	0	0.9	0.67	0.45
j15c10a	0.5	0.1	0	1	0.58	0
j10c5b	0.67	0.20	0	1	0.67	0.33
j10c10b	0.75	0.11	0	1	0.73	0.44
j15c5b	0.75	0.16	0	1	0.65	0.33
j15c10b	0.27	0.10	0	0.9	0.71	0.57
j10c5c	0.5	0.13	0	1	0.72	0.25
j10c10c	0.5	0.12	0	1	0.67	0.33
j15c5c	0.5	0.15	0	1	0.74	0.4
j10c5d	0.4	0.11	0	1	0.73	0.2

Table 5.5. Meilleures, Moyennes et Pire valeurs de *la métrique C* sur les 270 ensembles de solutions obtenues par chaque algorithme immunitaire par chaque problème test.

*5.3.*2 Comparaison de l'algorithme immunitaire MOIA1 et l'algorithme génétique NSGAII

L'étude que nous présentons dans cette section compare les approches immunitaire et génétique dans le même cadre d'expérimentations présenté dans le chapitre précédent.

Paramètres de tests :

Après plusieurs tests préliminaires en utilisant l'approche par essai et erreur (*trial and error*), les valeurs suivantes ont été choisies empiriquement comme paramètres de notre algorithme immunitaire en se basant sur la métrique *C* pour mesurer la qualité des solutions obtenues : c'est à partir de T_{mut} = 4 qu'on obtient de bonnes solutions, notre choix du nombre de mutations est égal à 6 . Le pourcentage d'élimination *B* est égal à 10%.

Concernant les paramètres de l'algorithme génétique NSGAII, la majorité des études utilisant les algorithmes génétiques montrent une certaine tendance pour certaines valeurs sur la base d'expérimentations. Par exemple, le taux de croisement est souvent choisi supérieur à 70 % et le taux de mutation est généralement choisi inférieur à 10 % par bit et moins de 50 % par chromosome. Pour nos expériences, nous fixons le taux de croisement à 0.8 comme dans [DAPM00] et [ACE07]. Le taux de mutation est fixé à 0.3 pour la production et à 0.01 pour la partie maintenance comme dans la plupart des applications des AGs aux problèmes de permutation [IM98].

La taille de population choisie égale à 100. Le test d'arrêt pour tous les algorithmes est le nombre de générations fixé à 100.

Chaque algorithme est appliqué à chaque problème de test 20 fois (20 exécutions). Pour chaque exécution, une nouvelle instance des durées des tâches de production est générée.

Métriques pour comparer les performances

En plus de la métrique *C* et la métrique du *nombre de solutions non dominées* obtenu utilisées dans le chapitre 4 pour évaluer les performances des algorithmes en terme de convergence et de taille de front respectivement, nous avons adopté dans cette section les deux métriques utilisées dans [BER09], la métrique *OS* (*Overall Pareto Spread metric*) pour tenir compte de l'aspect dispersion et la métrique $D1_R$ pour l'aspect distribution des solutions du front.

La métrique $D1_R$ est une mesure qui permet d'évaluer aussi bien la distribution d'un front *A* que la distance de *A* au front de référence (i.e., le front Pareto optimal ou le front Pareto approché). Soit *S** l'ensemble des solutions de référence. La mesure $D1_R$ d'un front *A* est donnée par l'équation:

$$\mathrm{D1_R(A)} = \frac{1}{|S^*|}\sum_{y \in S^*} \min\{d_{xy}, x \in A\}$$

Où d_{xy} est la distance entre une solution *x* et une solution de référence *y* dans l'espace des objectifs *p*-dimensionnel normalisé.

$$d_{xy} = \sqrt{\left(f_1^*(y) - f_1^*(x)\right)^2 + \cdots + \left(f_p^*(y) - f_p^*(x)\right)^2}$$

Où $f_i^*(.)$ est le $i^{ième}$ objectif qui est normalisé en utilisant l'ensemble des solutions de référence *S**. Comme dans [BER09], quand on compare les valeurs des métriques de deux fronts non dominés *A* et *B*, si le front Pareto n'est pas connu, nous combinons les deux fronts et sélectionnons toutes les solutions non dominées pour former l'ensemble *S**. Plus petite la valeur $D1_R$ *(A)* comparée à $D1_R$ *(B),* meilleur est le front *A*.

La métrique *OS (A, B)* mesure la dispersion relative de deux fronts *A* et *B* dans l'espace des objectifs. Dans le cas de problèmes à deux objectifs, cette mesure est donnée par l'équation ci-après. Nous avons rajouté de façon arbitraire la valeur de 0.5 à la formule originale afin d'éviter la division par zéro ou la forme indéterminée (le premier cas se présente quand le front *B* se compose d'une seule solution et le second cas, quand les deux fronts contiennent une seule solution chacun). Mentionnons ici que la valeur 0.5 n'a pas d'influence sur les résultats par rapport à la formule originale.

$$OS(A,B) = \frac{\prod_{i=1}^{2}\left|\max_{x\in A}\left(f_i(x)\right) - \min_{x\in A}\left(f_i(x)\right)\right| + 0.5}{\prod_{i=1}^{2}\left|\max_{x\in B}\left(f_i(x)\right) - \min_{x\in B}\left(f_i(x)\right)\right| + 0.5}$$

Si *OS (A, B) > 1*, le front A est dit préférable au front B par rapport à la dispersion globale.

Analyse des résultats

Dans les tableaux 5.6 - 5.9, nous comparons l'efficacité de l'algorithme MOIA1 par rapport à l'algorithme génétique NSGAII pour les instances faciles et difficiles, selon le nombre de solutions obtenues, la métrique *C*, la métrique *D1R* et la métrique *OS* respectivement. Comme on peut le voir, MOIA1 offre de bonnes solutions en termes de qualité. En effet, les solutions retournées par MOIA1 sont meilleures que celles obtenues par NSGAII de point de vue distribution et convergence.

La table 5.6 donne les meilleurs, les moyennes et les pires valeurs du critère *nombre de solutions obtenu*, sur les 20 exécutions réalisées par NSGAII et MOIA1. Comme on peut le voir dans cette table, les deux méthodes obtiennent de bonnes valeurs pour ce critère, NSGAII surpasse significativement MOIA1 en moyenne.

Les résultats de la métrique *C* présentés dans la Table 5.7 montrent que MOIA1 fournit de meilleurs résultats, en termes de convergence selon cette métrique, que l'algorithme NSGAII, pour la pluparts des instances (facile et difficile). MOIA1 surpasse clairement NSGAII en moyenne et il arrive à dominer NSGAII au moins une fois pour la plupart des problèmes de tests.

La Table 5.8 montre les résultats de la mesure *D1R*. A partir de cette table on peut constater que MOIA1 obtient des résultats meilleurs (petites valeurs) pour cette métrique que NSGAII dans tous les cas (meilleurs, moyens et pires résultats). Cela veut dire que les ensembles de solutions obtenus par MOIA1 sont plus distribués et approximent mieux le front Pareto que NSGAII.

La Table 5.9 rapporte des statistiques produites par (MOIA1 et NSGA II) pour la métrique *OS*. Cette comparaison montre qu'on peut arrondir à 1 les valeurs trouvées pour cette métrique. A partir de cette analyse, on peut noter que la dispersion des solutions des deux algorithmes est quasiment similaire pour toutes les instances.

problème	**NSGAII**			**MOIA1**		
	Meill.	**Moy.**	**Mauv.**	**Meill.**	**Moy.**	**Mauv.**
j10c5a	15	8.9	3	**15**	**7.4**	**2**
j10c10a	**18**	**9.2**	**3**	**15**	**8.4**	**1**
j15c5a	**15**	**9.8**	**6**	**14**	**7.2**	**2**
j15c10a	**19**	**10.65**	**6**	**19**	**8.05**	**2**
j10c5b	**14**	**6.95**	**2**	**9**	**5.9**	**1**
j10c10b	**16**	**11.45**	**6**	**16**	**10.3**	**3**
j15c5b	**11**	**7.3**	**4**	**6**	**4.15**	**2**
j15c10b	**20**	**13.4**	**8**	**19**	**10.02**	**4**
j10c5c	**12**	**8.7**	**6**	**13**	**7.15**	**3**
j10c10c	**15**	**9.85**	**6**	**12**	**7.8**	**5**
j15c5c	**15**	**11.25**	**6**	**16**	**9.1**	**5**
j10c5d	**13**	**8.6**	**5**	**12**	**7.1**	**4**
j15c5d	**15**	**10.5**	**6**	**14**	**10.15**	**5**

Table 5.6. Meilleures, Moyennes et Pire valeurs du *Nombre de solutions obtenues* sur les 20 exécutions réalisées par NSGAII et MOIA1.

problème	**NSGAII**			**MOIA1**		
	Meill.	**Moy.**	**Mauv.**	**Meill.**	**Moy.**	**Mauv.**
j10c5a	**1**	**0.38**	**0**	**1**	**0.44**	**0**
j10c10a	**1**	**0.4**	**0**	**1**	**0.45**	**0**
j15c5a	**0.86**	**0.4**	**0**	**1**	**0.47**	**0**
j15c10a	**1**	**0.34**	**0**	**1**	**0.49**	**0**
j10c5b	**1**	**0.38**	**0**	**1**	**0.49**	**0**
j10c10b	**0.8**	**0.32**	**0**	**0.87**	**0.49**	**0**
j15c5b	**1**	**0.57**	**0**	**0.86**	**0.28**	**0**
j15c10b	**0.75**	**0.22**	**0**	**1**	**0.65**	**0.11**
j10c5c	**1**	**0.46**	**0**	**0.89**	**0.42**	**0**
j10c10c	**0.87**	**0.33**	**0**	**0.87**	**0.50**	**0**
j15c5c	**0.85**	**0.39**	**0**	**1**	**0.49**	**0.07**
j10c5d	**1**	**0.46**	**0**	**0.9**	**0.41**	**0**
j15c5d	**0.83**	**0.29**	**0**	**1**	**0.58**	**0**

Table 5.7. Meilleures, Moyennes et Pire valeurs de *la métrique C* sur les 20 exécutions réalisées par NSGAII et MOIA1.

problème	NSGAII			MOIA1		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	**0**	**1.13**	**6**	**0**	**0.4**	**2**
j10c10a	**0**	**1.07**	**13.75**	**0**	**0.37**	**4.17**
j15c5a	**0**	**2.14**	**21.41**	**0**	**0.61**	**3.83**
j15c10a	**0**	**0.67**	**2.64**	**0**	**0.34**	**3.17**
j10c5b	**0**	**2.81**	**12.75**	**0**	**0.96**	**13.34**
j10c10b	**0**	**0.55**	**2.86**	**0**	**0.39**	**2.39**
j15c5b	**0**	**0.44**	**3.20**	**0**	**0.57**	**3.25**
j15c10b	**4.03e-005**	**1.54**	**6.01**	**0**	**0.3**	**3.07**
j10c5c	**0**	**0.63**	**3.63**	**0**	**0.23**	**1**
j10c10c	**0**	**0.78**	**3.22**	**0**	**0.22**	**2**
j15c5c	**2.10e-007**	**1.63**	**15.50**	**0**	**0.48**	**2.54**
j10c5d	**0**	**0.71**	**4**	**0**	**0.16**	**1.20**
j15c5d	**0**	**0.73**	**4.83**	**0**	**0.22**	**1.22**

Table 5.8. Meilleures, Moyennes et Pire valeurs de *la métrique D1R* sur les 20 exécutions réalisées par NSGAII et MOIA1.

problème	MOIA1		
	Meill.	Moy.	Mauv.
j10c5a	**1.14**	**0.98**	**0.73**
j10c10a	**1.13**	**0.86**	**0.008**
j15c5a	**1.12**	**0.97**	**0.79**
j15c10a	**1.17**	**0.97**	**0.79**
j10c5b	**1.18**	**0.91**	**0.01**
j10c10b	**1.12**	**0.96**	**0.82**
j15c5b	**1.07**	**0.96**	**0.86**
j15c10b	**1.06**	**0.96**	**0.85**
j10c5c	**1.09**	**0.99**	**0.92**
j10c10c	**1.22**	**0.99**	**0.76**
j15c5c	**1.23**	**0.96**	**0.79**
j10c5d	**1.16**	**0.94**	**0.62**
j15c5d	**1.23**	**0.93**	**0.8**

Table 5.9. Meilleures, Moyennes et Pire valeurs de *la métrique OS* sur les 20 exécutions réalisées par NSGAII et MOIA1.

Représentation graphique des ensembles de solutions générés

En plus de l'analyse numérique de différentes métriques, nous avons représenté graphiquement les ensembles Pareto générés par les deux algorithmes de la même manière que dans [ZDT00]. Tous les ensembles Pareto générés par chaque algorithme *i* dans les vingt exécutions réalisées vont être fusionnés dans un seul ensemble Pareto $\overline{P}_i$ en supprimant les solutions dominées. Notons $\overline{P}_i^j$ l'ensemble non dominé obtenus par l'algorithme *i* à la $j^{ième}$ exécution, P_i est l'union de tous les ensembles de solutions obtenus par l'algorithme *i* sur les 20 exécutions ($P_i = \overline{P}_i^1 \cup \overline{P}_i^2 \cup ... \cup \overline{P}_i^{20}$) et $\overline{P}_i$ l'ensemble de toutes les solutions non dominées dans P_i. La figure 5.3 montre les ensembles $\bar{P}_i$ pour deux problèmes de test. Nous avons choisi une instance facile (J10C10C) et une autre difficile (J10C5D) pour visualiser la

performance de l'algorithme MOIA1 en comparant avec l'algorithme NSGAII. En effet, les solutions retournées par MOIA1 sont meilleures que celles obtenues par NSGAII de point de vue distribution et convergence.

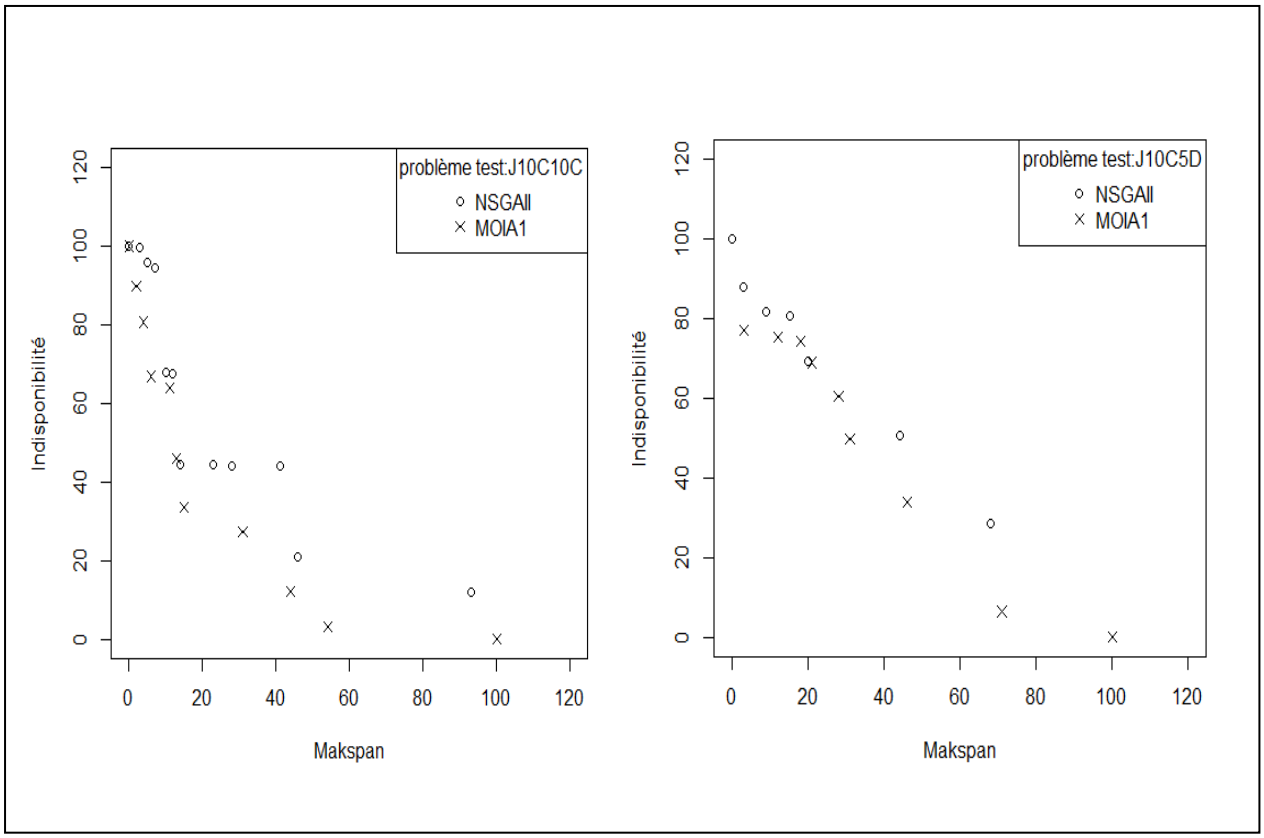

Figure 5.3. Représentation graphique des ensembles de solutions non dominées générés par chaque algorithme pour deux problèmes tests.

Dans la table 5.10, nous comparons les deux algorithmes en termes de temps de calcul. Notons que la procédure de *Ranking* a été utilisée par les deux algorithmes pour extraire le front non dominé. Cette table montre qu'en termes de temps de calcul, NSGA-II est plus rapide que MOIA1. La fonction de calcul de l'affinité basée sur les deux procédures *Ranking* et *Crowding* semble influencer la vitesse d'exécution de l'algorithme MOIA1. En effet, elle est appelée à chaque génération $\boldsymbol{N}$ fois (N est la taille de la population de clonage P) pour calculer les affinités des anticorps de la population P, puis $\boldsymbol{N_c}$ fois pour calculer l'affinité des anticorps clonés (N_c étant la taille de l'ensemble de clones).

problème	NSGAII	MOIA1
j10c5a	4.93 (0.09)	24.94 (2.004)
j10c10a	7.85 (0.61)	32.32 (3.13)
j15c5a	6.15 (0.43)	27.29 (1.79)
j15c10a	10.24 (0.64)	39.31 (3.34)
j10c5b	5.21 (0.86)	25.61 (2.25)
j10c10b	9.05 (0.51)	36.75 (2.94)
j15c5b	6.84 (0.27)	32.123 (2.303223)
j15c10b	11.12 (1.20)	39.96 (4.98)
j10c5c	5.99 (0.18)	30.42925 (2.482981)
j10c10c	7.84 (0.14)	32.67 (1.76)
j15c5c	6.79 (0.64)	28.20 (3.98)
j10c5d	5.13 (0.10)	24.55 (1.27)
j15c5d	6.23 (0.08)	26.60 (1.77)

Table 5.10. Comparaison des deux algorithmes en utilisant le temps de calcul (en seconds). Les valeurs moyennes sur les 20 exécutions avec l'écart-type entre parenthèses.

5.4 OPTIMISATION DE L'ALGORITHME IMMUNITAIRE MOIA1 PAR UN CONTROLEUR FLOU

5.4.1 La logique floue

De nos jours, la logique floue (en anglais «fuzzy logic») est un axe de recherche important sur lequel se focalisent de nombreux scientifiques, c'est une extension de la logique booléenne, elle a pour but de raisonner à partir de connaissances imparfaites qui opposent résistance à la logique classique. Pour cela la logique floue se propose de remplacer les variables booléennes par des variables floues. Elle a été créée par Lotfi Zadeh en 1965 [ZAD65] en se basant sur sa théorie mathématique des ensembles flous.

La structure d'une commande floue, présentée dans la figure 5.5, peut être décomposée en trois grands modules, Fuzzification, Moteur d'inference, Défuzzification.

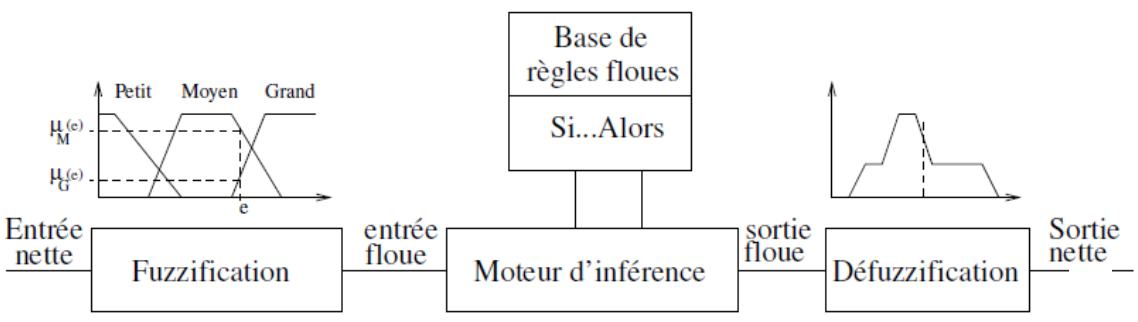

Figure 5.5. Structure générale d'une commande floue

Le premier de ces modules traite les entrées du système : c'est la fuzzification. Il permet d'associer à chacune des entrées réelles, par le biais de fonctions d'appartenances, un degré d'appartenance pour chacun des sous-ensembles flous définis sur l'ensemble du discours. Le deuxième module est constitué du moteur d'inférence et de la base de règles. Celle-ci est constituée de règles de type : "Si..., Alors..." et va permettre de passer des degrés d'appartenance des grandeurs d'entrées aux degrés d'appartenance aux sous-ensembles flous de la grandeur de commande. Le moteur d'inférence, lui, va permettre de générer une conclusion à partir des entrées et des règles actives. Il calcule alors les degrés d'appartenance aux sous-ensembles flous correspondant à la commande du système. Deux grandes classes de modèles flous sont répertoriées selon la nature des conclusions de leurs règles : Les modèles flous de Mamdani qui utilisent une conclusion symbolique (une valeur linguistique) et les modèles flous de Takagi-Sugeno où les conclusions sont numériques ou des équations mathématiques.

Enfin, le dernier module, l'interface de défuzzification, va permettre de transformer les degrés d'appartenance des sous-ensembles flous de commande en grandeur numérique. C'est la transformation inverse du module de fuzzification.

Pour plus de renseignements concernant la logique floue, on peut revenir à la référence [KOS92]. La section suivante a pour but de présenter plus en détails la structure retenue pour le contrôleur floue FLC utilisé dans notre étude pour le réglage des paramètres de l'algorithme MOIA1.

5.4.2 Réglage des paramètres de MOIA1 par un contrôleur flou

L'algorithme immunitaire MOIA1 développé précédemment est basé sur le principe de la maturation d'affinité pour l'évolution de la population d'anticorps. Les deux mécanismes de base de ce principe sont l'hyper-mutation et la génération des récepteurs.

Le processus de mutation est réalisé selon une loi de Bernoulli de probabilité $\boldsymbol{P_m}$ pour la partie maintenance. Pour la partie production, une procédure de mutation en deux phases est utilisée où chaque clone généré subit une mutation simple dans la première phase et une mutation large dans la deuxième phase avec un nombre de mutations $\boldsymbol{T_{mut}}$. Pour le mécanisme de génération de récepteurs, un pourcentage ($\boldsymbol{B}\%$) d'anticorps de la population sont éliminés et remplacés par d'autres anticorps sélectionnés aléatoirement à partir de la population secondaire. Pour améliorer le paramétrage ($\boldsymbol{P_m, T_{mut}}$ $\boldsymbol{et}$ $\boldsymbol{B}\%$) de notre algorithme

immunitaire MOIA1, on propose de l'hybrider avec un contrôleur de logique floue (**FLC**) et on note le nouvel algorithme développé FLC-MOIA1. Le réglage dynamique des paramètres de l'algorithme FLC_MOIA1 (annexe A.4) est effectué à chaque période de dix générations consécutives [LCTH09] pour offrir un temps suffisant et nécessaire à ce dernier pour répondre aux changements. Dans la présente section, nous présentons la structure du contrôleur FLC ainsi que son fonctionnement. La figure ci –après montre cette structure.

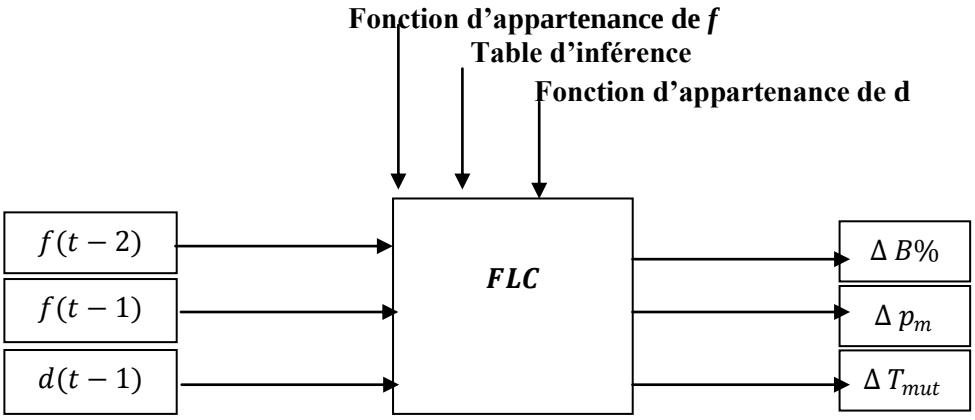

Figure 5.4 .La structure de FLC

Les entrées de FLC : La valeur moyenne des fonctions objectifs *F* des solutions du front non-dominé ainsi que leur diversité sont prises comme entrées de FLC. L'optimisation considérée dans notre étude est bi objectif minimisant le Makspan et l'indisponibilité des machines alors $f(t-1), f(t-2)$ sont les valeurs moyennes du Makspan et l'indisponibilité de la population à l'itération *t-1* et *t-2* respectivement. $d(t-1)$ représente le degré de diversité de la population à l'itération *t-1*.

Les sorties de FLC: la variation de nombre de fois de mutations (large) ΔT_{mut} , la variation de probabilité de mutation Δp_m et la variation de pourcentage d'élimination $\Delta B\%$ sont déterminées comme sorties du contrôleur flou.

Le nombre de mutations T_{mut} est calculé par : $T_{mut}(t) = T_{mut}(t-1) + \Delta T_{mut}$ où $T_{mut}(t)$, $T_{mut}(t-1)$ sont les nombres de mutations à l'itération *t* et *t-1* respectivement.

La probabilité de mutation P_m est calculé par : $P_m(t) = P_m(t-1) + \Delta P_m$ où $P_m(t)$ et $P_m(t-1)$ sont les probabilités de mutation à l'itération *t* et *t-1* respectivement.

Le pourcentage d'élimination $B\%$ est calculé par: $B\%(t) = B\%(t-1) + \Delta B\%$ où $B\%(t)$, $B\%(t-1)$ sont les pourcentages d'élimination à l'itération *t, t-1* respectivement.

5.4.3 Le contrôleur flou FLC

Le fonctionnement de notre contrôleur flou FLC passe par les étapes suivantes :

- Choix de la stratégie de fuzzification.
- Etablissement de la base des règles.
- Choix de la méthode d'inférence.
- Choix de la stratégie de défuzzification.

La description de chacune de ces étapes est donnée ci-dessous :

– **La fuzzification :**

Durant cette phase, les deux entrées sont tout d'abord traitées:

• $\boldsymbol{f_a(t)} = f(t-1) - f(t-2)$ qui est la déférence entre la valeur moyenne des fonctions objectifs de la population à l'itération *t-1* et l'itération *t-2*.

• $\boldsymbol{d(t-1)}$ est la somme de la distance de *Hamming* entre tous les individus de la population à l'itération *t-1* (voir l'article de Lau et al. [LCTH09] pour plus de détails).

Ensuite, les données traitées sont transformées en qualifications linguistiques (ensembles flous). Il est aussi indispensable de fuzzifier les variables de sortie du contrôleur. On a besoin de ces ensembles flous au niveau de la formulation des inférences et lors de la défuzzification. La fuzzification des variables est une phase délicate du processus mis en œuvre par la logique floue. Elle est souvent réalisée de manière itérative et requiert de l'expérience. On peut choisir plusieurs formes de fonction d'appartenance, mais les plus usuelles sont triangulaires, trapézoïdales ou gaussiennes, nous avons opté pour la forme triangulaire symétrique pour représenter les deux variables d'entrée $f_a(t)$ et $d(t)$ ainsi que pour représenter les deux variables de sortie $\Delta\, p_m$ et $\Delta\, B\%$ et une fonction d'appartenance singleton pour représenter la variable de sortie $\Delta\, T_{mut}$. L'univers de discours et la fonction d'appartenance de chaque variable sont montrés dans la figure 5.6. La signification de chaque terme linguistique est présentée dans la table 5.11.

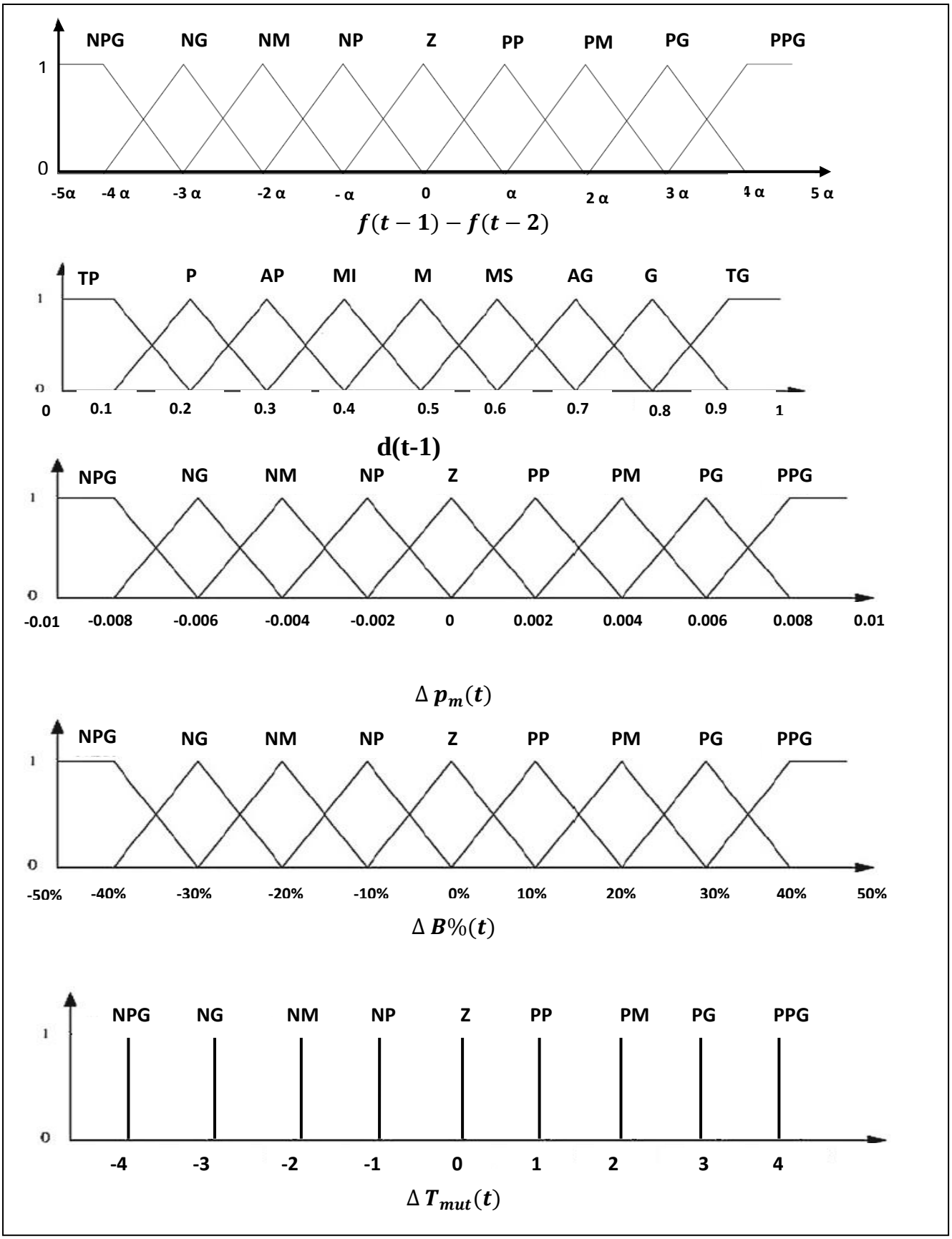

Figure 5.6.les fonctions d'appartenance

Les termes linguistiques pour $f_a(t)$, $\Delta T_{mut}(t)$, $\Delta B\%(t)$, $\Delta P_m(t)$	**Signification**	**Les termes linguistiques pour *d(t-1)***	**Signification**
NPG	Négatif Plus Grand	TP	Très Petit
NG	Négatif Grand	P	Petit
NM	Négatif Moyen	AP	Assez Petit
NP	Négatif Petit	MI	Moyen Inferieur
Z	Zéro	M	Moyen
PP	Positif petit	MS	Moyen Supérieur
PM	Positif Moyen	AG	Assez Grand
PG	Positif Grand	G	Grand
PPG	Positif Plus Grand	TG	Très Grand

Table 5.11. Les termes linguistiques

– **La base de règles :** une base de règles floues est une collection de règles qui permet de lier les variables floues d'entrée et de sortie à l'aide de différents opérateurs (*et, ou, non*). Elles sont définies par le concepteur de système de réglage en fonction de son expérience et mémorisées dans l'organe de commande. La description de la commande se fait par l'intermédiaire de ces règles qui ont la forme suivante: $\boldsymbol{Si\ x_1\ est\ A_1\ et\ x_2\ est\ A_2\ Alors\ y\ est\ B}$ Où x_1 , x_2 sont les deux entrées du contrôleur et *y* est la variable de sortie. A_1, A_2 et *B* sont les termes linguistiques. Il existe déférentes possibilités d'exprimer les inférences, à savoir par description linguistique et symbolique, ainsi que par matrices et tableaux d'inférence. Nous avons choisi la matrice d'inférence de Lau et al. [LCTH09] pour régler les trois paramètres de sortie (Δp_m, ΔB et ΔT_{mut}) de notre contrôleur. Un exemple des règles présentées dans la table 5.12 est montré comme suit : *Si* $f_a(t)$ *est PPG et* $d(t-1)$ *est TG Alors* Δp_m *est Z.*

d(t-1)\f$_a$(t)	**NPG**	**NG**	**NM**	**NP**	**Z**	**PP**	**PM**	**PG**	**PPG**
TG	NPG	NPG	NG	NG	NM	NM	NP	NP	Z
G	NPG	NG	NG	NM	NM	NP	NP	Z	PP
AG	NG	NG	NM	NM	NP	NP	Z	PP	PP
MS	NG	NM	NM	NP	NP	Z	PP	PP	PM
M	NM	NM	NP	NP	Z	PP	PP	PM	PM
MI	NM	NP	NP	Z	PP	PP	PM	PM	PG
AP	NP	NP	Z	PP	PP	PM	PM	PG	PG
P	NP	Z	PP	PP	PM	PM	PG	PG	PPG
TP	Z	PP	PP	PM	PM	PG	PG	PPG	PPG

Table 5.12. Matrice d'inférence pour$\Delta\ \boldsymbol{p_m}$, $\Delta\ \boldsymbol{B\%}$ $\boldsymbol{et}$ $\Delta\ \boldsymbol{T_{mut}}$

– **Méthode d'inférence floue :** dans les inférences par logique floue interviennent les operateurs *ET, OU*. L'operateur *ET* s'applique aux variables à l'intérieur d'une règle, tandis que l'operateur *OU* lie les différentes règles. A cause de l'empiétement des fonctions d'appartenance, en général deux ou plusieurs règles sont activées en même temps (une règle est activée lorsque le facteur d'appartenance lié à la condition de cette règle est non nul). Ce fait doit être pris en considération lors de la réalisation de l'operateur *OU*.

Il existe plusieurs possibilités pour réaliser ces operateurs à savoir, la méthode d'inférences max-min, la méthode d'inférence max-prod et la méthode d'inférence somme-prod. Nous avons opté pour la méthode d'inférence somme-prod pour tenir compte des valeurs floues des deux variables d'entrée. Cette méthode réalise, l'opérateur *OU* qui lie les différentes règles par la formation de la somme, plus précisément par la valeur moyenne, tandis que l'opérateur *ET* est réalisé par la formation du produit.

– **La défuzzification :**

Le but de cette étape est de calculer les valeurs de variation de probabilité de mutation Δp_m et de pourcentage d'élimination $\Delta B\%$, la variation du nombre de fois de mutation ΔT_{mut} en utilisant les ensembles flous des variables de sortie résultants de l'agrégation des règles ainsi que leurs fonctions d'appartenance. Dans la littérature, il existe plusieurs stratégies pour réaliser cette opération telle que la moyenne des maxima, le centre des aires, le centre des maxima. La méthode de défuzzification par le centre de gravité est la méthode la plus utilisée en commande floue du fait qu'elle fournit intuitivement la valeur la plus représentative de l'ensemble flou issu de l'agrégation des règles. Le calcul de centre de gravité dans le cas simple de fonctions d'appartenance singleton ou symétrique correspond au calcul d'une somme pondérée. Dans notre étude, Les fonctions d'appartenance des variables de sortie sont de type triangulaire symétrique ou singleton alors nous avons adopté la méthode des hauteurs pondérées pour défuzzifier les variables de sortie comme suit [MER09]:

$$y_{cg} = \left(\sum_{i=1}^{m} \mu_{R_i} * y_i \right) \Big/ \sum_{i=1}^{m} \mu_{R_i}$$

Où y_i et μ_{R_i} sont respectivement le centre et le degré d'appartenance de la *i-ème* règle.

5.4.4 Comparaison de l'algorithme FLC-MOIA1 et l'algorithme MOIA1

Dans cette section, nous allons tester la performance de l'algorithme immunitaire FLC-MOIA1 proposé en le comparant avec l'algorithme MOIA1 sur le même jeu d'instances que dans le chapitre précédent. La structure de la FLC-MOIA1 est donnée à l'annexe A4.

Pour nos expériences, la taille de la population et le nombre de générations sont fixé à 100.

L'algorithme FLC-MOIA1 a été exécuté avec les paramètres suivants : la probabilité de mutation initiale est de 0.1, le nombre de mutations T_{mut} initial est de 6, le pourcentage d'élimination initial est de 60%. Nous limitons la valeur du taux de mutation à l'ensemble des valeurs $T_{mut} \in \{2, \dots, 7\}$ et la valeur de la probabilité de mutation à l'ensemble des valeurs $P_m \in \{0, \dots, 0.1\}$, la valeur de pourcentage d'élimination à l'ensemble des valeurs $B\% \in \{10\%, \dots, 70\%\}$. La valeur α est déterminée empiriquement pour chaque instance après plusieurs expériences. Ce coefficient est utilisé pour ajuster la différence *f (t-1)-f (t-2)* dans la phase de fuzzification de la variable d'entrée $f_a(t)$.

L'algorithme MOIA1 a été exécuté avec les paramètres suivants : la probabilité de mutation est de 0.01, le taux de mutation est de 6, le pourcentage d'élimination est de 10.

Chacun des deux algorithmes est appliqué à chaque problème de test 20 fois. Pour chaque exécution, une nouvelle instance des temps de traitements des tâches est générée.

Nous avons utilisé deux métriques pour évaluer la convergence et la diversité des fronts non dominées des deux algorithmes: la métrique d'hyper volume *H* et la métrique *C* définies dans le chapitre précédent.

Analyse des résultats

Dans cette section nous comparons les deux algorithmes MOIA1 et FLC-MOIA1 suivant la métrique *C* et la métrique d'hyper-volume *H*. Les tableaux 5.13 et 5.14 représentent les résultats de ces deux métriques respectivement. Les valeurs moyennes de la métrique *C* obtenues par FLC-MOIA1sont toujours supérieures à celles trouvées par MOIA1.

En comparant les valeurs de l'hyper-volume des deux algorithmes (Table 5.14), nous remarquons que FLC-MOIA1 obtient globalement les valeurs les plus petites (donc les meilleures) de l'hyper-volume.

problème	MOIA1			FLC-MOIA1		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	1	0.36	0	1	0.54	0
j10c10a	1	0.30	0	1	0.44	0
j15c5a	0.8	0.32	0	1	0.46	0
j15c10a	1	0.35	0	1	0.54	0
j10c5b	1	0.39	0	1	0.52	0
j10c10b	1	0.35	0	1	0.50	0
j15c5b	1	0.41	0	1	0.43	0
j15c10b	1	0.43	0	1	0.45	0
j10c5c	0.92	0.35	0	1	0.51	0
j10c10c	1	0.35	0	1	0.46	0
j15c5c	1	0.41	0	1	0.48	0
j10c5d	1	0.38	0	1	0.54	0
j15c5d	1	0.39	0	0.9	0.46	0

Table 5.13. Meilleures, Moyennes et Pire valeurs de la métrique *C* sur les 20 exécutions réalisées par MOIA1 et FLC-MOIA1.

problème	MOIA1			FLC-MOIA1		
	Meill.	Moy.	Mauv.	Meill.	Moy.	Mauv.
j10c5a	33.12	34.18	37.1	34.33	36.80	39.46
j10c10a	54.3	58.8	62.3	54.78	56.93	59.03
j15c5a	51.24	54.54	57.13	48.73	51.13	55.61
j15c10a	97.76	104.37	113.83	93.57	99.02	105.22
j10c5b	41.47	42.87	44.26	39.32	42.27	45.22
j10c10b	54.83	65.63	75.24	55.83	62.57	71.53
j15c5b	47.77	56.68	62.19	47.16	54.76	59.99
j15c10b	89.9	91.2	92.55	88.11	92.42	96.73
j10c5c	3.45	3.82	4.32	3.27	3.74	4.22
j10c10c	6.81	6.84	6.86	6.36	6.86	7.36
j15c5c	4.71	5.4	6.2	4.84	5.31	5.69
j10c5d	1.01	1.13	1.26	1.05	1.28	1.52
j15c5d	2.03	2.23	2.4	1.74	2.08	2.39

Table 5.14. Meilleures, Moyennes et Pire valeurs de *la métrique H* sur les 20 exécutions réalisées par MOIA1 et FLC-MOIA1.

Conclusion

Dans ce chapitre, nous avons proposé deux algorithmes immunitaires multi objectifs s'inspirant du principe de la sélection clonale et du principe de maturation d'affinité du système immunitaire naturel pour la résolution du problème d'ordonnancement conjoint production /maintenance dans un environnement de type flow-shop hybride. Le premier algorithme proposé MOIA1 est basé sur la non dominance pour la sélection et le clonage des anticorps alors que le deuxième MOIA2 est basé sur la somme pondérée des critères. Une série de tests a été menée sur le même jeu d'instances et dans le même cadre d'expérimentations que dans le chapitre précédent pour évaluer les méthodes de résolution proposées. Nous avons présenté les résultats obtenus en deux volets : le premier concerne la comparaison entre les deux algorithmes immunitaires. Il en résulte que la meilleure méthode est celle basé sur le concept de dominance de Pareto pour la sélection et le clonage *MOIA1* quelque soit le type d'instance (difficile ou facile). Dans le second volet, nous avons testé les

performances de l'algorithme immunitaire MOIA1 en le comparant avec l'algorithme génétique NSGAII. Nous avons trouvé que l'algorithme MOIA1 offre de bonnes solutions en termes de qualité et de quantité. En effet, les solutions retournées par MOIA1 sont meilleures que celles obtenues par NSGAII de point de vue distribution et convergence. Nous avons combiné l'algorithme MOIA1 avec un contrôleur de la logique floue FLC, pour améliorer son paramétrage, afin d'optimiser la qualité des solutions retournées par l'algorithme de point de vue convergence et diversité. Nous avons testé et comparé les deux algorithmes MOIA1 et FLC-MOIA1 sur les mêmes instances que dans le chapitre précédant. L'étude comparative des deux algorithmes suivant les deux mesures de performance *H* et *C* montre que le nouvel algorithme FLC-MOIA1 trouve globalement les meilleurs résultats.

CONCLUSION GENERALE ET PERSPECTIVES

L'objectif de ce mémoire est d'étudier la problématique de l'ordonnancement conjoint production et maintenance dans un environnement de type Flow Shop Hybride et d'en fournir une solution. Le travail que nous venons de présenter nous a permis de tester un ensemble de méta-heuristiques à base d'optimisation par une approche génétique et une autre immunitaire pour la résolution de ce problème. La recherche bibliographique réalisée a montré : l'importance de ce type d'atelier dans le monde industriel et un nombre significatif d'articles réalisés dans le domaine d'ordonnancement avec contraintes de disponibilité des machines (approche déterministe) pour le cas d'un atelier flow shop hybride. Par contre, il y a peu de travaux qui s'intéressent à l'approche stochastique où l'ordonnancement de la production et la maintenance sont considérés conjointement. De plus, les ateliers de type Flow Shop Hybride n'a fait l'objet d'aucune étude prenant en considération la production et la maintenance simultanément.

Notre contribution s'est focalisée sur l'étude d'un modèle bi objectif intégré dans un atelier de production de type Flow Shop Hybride. Le modèle optimise simultanément deux critères de performance. L'un est lié à l'ordonnancement de la production, le Makespan (largement utilisé) et l'autre est lié à la maintenance. Pour l'aspect maintenance, des modèles de fiabilité ont été utilisés pour tenir compte des critères de performance du système conjointement aux critères de production, sans privilégier *a priori* les uns par rapport aux autres. Ce problème étant NP-difficile, pour le résoudre, nous nous sommes intéressés aux méta-heuristiques qui ont prouvé leurs aptitudes de résolution pour un nombre important de problèmes d'optimisation. Deux méthodes de résolution ont été considérées : les algorithmes génétiques et les systèmes immunitaires artificiels. L'optimisation considérée est bi-objectif de type Pareto afin de trouver un compromis pour le service d'ordonnancement et le service de maintenance. Les variables de décision du problème sont l'ordre de tâches de production sur les machines et les dates de début des tâches de maintenance préventive de chaque machine.

Dans ce mémoire, nous avons étudié les deux approches et comparé leurs résultats.

Les algorithmes génétiques sont très bien adaptés au traitement d'un problème D'optimisation multi objectifs. En témoigne le nombre important d'articles qui ont été publiés sur ce sujet. Deux algorithmes génétiques multi objectifs élitistes sont adaptés à notre modèle bi objectif. Ils ont été comparés sur plusieurs problèmes de test en utilisant des mesures appropriées pour tenir compte de la qualité de solutions. Les deux AGs utilisent la dominance de Pareto pour déterminer le front de Pareto. Ils traitent les deux objectifs de production et de maintenance d'une manière

égale. Les résultats obtenus indiquent que les AGs proposés peuvent être utilisés efficacement pour résoudre le problème étudié. Les perspectives de recherche incluent des investigations sur l'hybridation avec des heuristiques de type recherche locale pour augmenter les performances des AGs.

Un système immunitaire artificiel (SIA) est une catégorie d'algorithmes inspirée par les principes et le fonctionnement du système immunitaire naturel (SIN) des vertébrés. Les caractéristiques intéressantes des systèmes immunitaires ont encouragé leur adaptation au domaine informatique pour la résolution d'une large gamme de problèmes notamment les problèmes d'optimisation multi objectifs. La principale difficulté d'utilisation des SIA est de trouver le modèle biologique à partir duquel il faut s'inspirer pour résoudre un problème ainsi que la représentation de ce problème sous forme biologique. Dans ce mémoire nous avons développé deux algorithmes multi objectifs s'inspirant du principe de la sélection clonale et du principe de maturation d'affinité du système immunitaire naturel. Le premier (MOIA1) est basé sur la non-dominance pour la sélection et le clonage des anticorps alors que le deuxième(MOIA2) est basé sur la somme pondérée des critères. Nous avons testé les performances de l'algorithme immunitaire MOIA1 en le comparant d'une part avec l'algorithme immunitaire MOIA2 et d'autre part avec l'algorithme génétique NSGAII. Nous avons trouvé que l'algorithme MOIA1 offre de bonnes solutions en termes de qualité et de quantité. Nous avons amélioré la qualité des solutions retournées par ce dernier en intégrant un contrôleur de la logique floue (FLC) pour le réglage dynamique des paramètres de l'algorithme MOIA1.

Comme perspective de notre travail, nous envisageons :

– d'hybrider notre méthode immunitaire avec d'autres méthodes bio-inspirées. Cette hybridation nous permettra de tirer profit des points forts de différentes méthodes de résolutions, et d'élaborer plusieurs schémas d'hybridation et de coopération entre ces méthodes.

– L'optimisation du contrôleur flou conçu par un algorithme génétique.

– Etudier d'autres types de maintenance régissant le fonctionnement du système.

– D'autres contraintes ou critères, liés soit à la maintenance soit à la production peuvent être intégrés au modèle pour viser une performance globale du système.

– Le développement de méthodes exactes et la recherche de bornes inférieures pour les critères liés à cette problématique.

REFERENCES

[AA06] Allaoui H, Artiba A. Scheduling two-stage hybrid flow shop with availability constraints. Computers and Operations Research, 33, 5, pp.1399-1419, 2006.

[ACE07] Amodeo L, Chen H & El Hadji A. Multi-objective Supply Chain Optimization: An Industrial Case Study. Springer-Verlag, EvoWorkshops, LNCS 4448, pp.732-741, 2007.

[AFNOR] AFNOR**.** Recueil de normes françaises X 06, X 50, X 60, AFNOR.

[AGG02] Aggoune R. Ordonnancement d'ateliers sous contraintes de disponibilité des machines. Thèse de doctorat, Université de Metz, France, 2002.

[AMO99] Amodeo L. Contribution à la simplification et à la commande des réseaux de Pétri stochastiques. Application aux systèmes de production, Thèse de doctorat en Productique soutenue à l'INPG, Grenoble (France), 1999.

[BAR03] Barichard V. Approches Hybrides pour les Problèmes Multiobjectifs. Thèse de Doctorat, Université d'Angers, pp. (9, 11,37, 40), 2003.

[BAV02] Bavier G. Les techniques d'ordonnancement, disponible sur http://www.peda.acmartinique. fr/ecogest/ressources/cours/ordo/Techordo.pdf.

[BAYCM08] Berrichi A, Amodeo L, Yalaoui F, Châtelet E and Mezghiche M. Une approche évolutionnaire bi-objectif pour l'ordonnancement conjoint production- maintenance : cas de machines parallèles.7e Conférence Internationale de MOdélisation et SIMulation « Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.» ,MOSIM, ,Paris(France), 2008.

[BAYCM09] Berrichi A, Amodeo L, Yalaoui F, Châtelet E & Mezghiche M. Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem. Journal of Intelligent Manufacturing, 20 (4), pp. 389-400, 2009.

[BBGVZ03] Benbouzid F, Bessadi Y, Guebli S, Varnier C, Zerhouni N. Résolution du problème de l'ordonnancement Conjoint Maintenance/Production par la stratégie séquentielle. 4ème Conférence Francophone de Modélisation et de SIMulation ''Organisation et Conduite d'Activités dans l'Industrie et les Services'' MOSIM'03, Toulouse (France), 2003.

[BEM02] Bembla M. Ordonnancement conjoint production et maintenance : Critère et heuristique de résolution. Mémoire de DEA, U.F.R des Sciences et Techniques de l'Université de Franche Comte, 2002.

[BEN05] Benbouzid F. Contribution à l'étude de la performance et de la robustesse des ordonnancements conjoints production/maintenance –cas du flow shop. Thèse de doctorat en Automatique et Informatique de l'Université de Franche-Comté, France, 2005.

[BEN08] Benyamina A. Ordonnancement Hiérarchique Multi-Objectif d'Application Embarquees Intensives. Thèse pour l'obtention du Diplôme De Doctorat D'Etat, 2008.

[BEPSW96] Blazewicz J, Ecker K.H, Pesch E, Schmidt G, Weglarz J. *Scheduling Computer and Manufacturing Processes*. Springer, Berlin, 1996.

[BEPSW07] Blazewicz J, Ecker K.H, Pesch E, Schmidt G, Weglarz J .*Handbook on scheduling from Theory to Applications* . International Handbooks on Information, 2007.

[BER01] Berrou A. Optimisation multi objectif et stratégies d'évolution en environnement dynamique. Thèse de doctorat de l'Université des Sciences Sociales, Toulouse I, France, 2001.

[BER09] Berrichi A. La gestion à deux niveaux avec optimisation de la production et de la maintenance sous diverses contraintes : cas mono et multicritère. Thèse de doctorat de l'Université M'hamed Bouguara de Boumerdès (UMBB), Algérie, 2009.

[BGBVZ11] Benbouzid Sitayeb F, Guebli S, Bessadi Y , Varnier C, Zerhouni N. Joint scheduling of jobs and preventive maintenance operations in the flowshop sequencing problem: A resolution with sequential and integrated strategies. International Journal of Manufacturing Research 6, 1,pp.30-48, 2011.

[BTL10] Besbes W, Teghem J, Loukil T. Scheduling hybrid flow shop problem with non-fixed availability constraints. European J. of Industrial Engineering, Vol.4, No.4, pp.413 - 433 ,2010

[BTM07] Benbouzid F,Tirchi M, Mahloul A. An Integrated Resolution of Joint Production and Maintenance Scheduling Problem in Hybrid Flowshop. IWINAC , Part I, LNCS 4527, pp. 518–527, 2007.

[BVZ06] Benbouzid F, Varnier C, Zerhouni N. Résolution du problème de l'ordonnancement conjoint production/maintenance par colonies de fourmis. 6e Conférence Francophone de MOdélisation et SIMulation « Modélisation, Optimisation et Simulation des Systèmes : Défis et Opportunités » MOSIM'06, Rabat (Maroc), 2006.

[BYAM10] Berrichi A, Yalaoui F, Amodeo L, Mezghiche M .Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. Computers & Operations Research ,37 ,pp.1584–1596, 2010.

[CC88] Carlier J, Chretienne P. *Les Problèmes d'ordonnancement*. Masson, Paris, France, 1988.

[CC98] Coello Coello C A. Using a min-max method to solve multiobjective optimization problems with genetic algorithms. Dans IBERAMIA'98, Springer-Verlag, Lecture notes in computer science, pp.303-314, Lisbon, Portugal, 1998.

[CC01] Coello Coello C A. A Short Tutorial on Evolutionary Multiobjective Optimization. First International Conference on Evolutionary Multi-Criterion Optimization, Springer-Verlag, Lecture Notes in Computer Science No 1993, pp. 21-40, 2001.

[CC02] Coello Coello C A, Van Veldhuizen D A, et Lamont G B. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York, 576 p, 2002.

[CCF13] Costa A, Cappadonna F A, Fichera S. A dual encoding-based meta-heuristic algorithm for solving a constrained hybrid flow shop scheduling problem. Computers & Industrial Engineering,64, pp. 937–958,2013.

[CK03] Cassady C R & Kutanoglu E. Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. IIE Transactions, 35: pp.503–513, 2003.

[CLJ10] Chen J, Lin Q, Ji Z. A hybrid immune multiobjective optimization algorithm. European Journal of Operational Research 204, 294–302, 2010.

[CMM90] Carraway R L, Morin T L, and Moskowitz H. Generalized dynamic programming for multicriteria optimization. European Journal of Operational Research, 44 ,pp.95–104, 1990.

[CN00] Carlier J & Néron E. An exact method for solving the multi-processor flow-shop. *RAIRO-RO*, 34(1), pp.1-25, 2000.

[DAV91] Davis L. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, 1991.

[DEB99] Deb K. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering design. Evolutionary algorithms in engineering and computer science, pp. 135-161, 1999.

[DEJ75] De Jong K A. An analysis of the Behaviour of a Class of Genetic Adaptive Systems. Thèse de Doctorat. Université de Michigan,1975.

[DEN08] DENIS P. Guide de la maintenance industrielle [Livre]. - [s.l.] : DELAGRAVE, 2008.

[DFV02] De Castro L N & Fernando J, Von Zuben F J .Learning and Optimization Using The Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, vol. 6, n. 3, pp.239- 251, 2002.

[DPAM02] Deb K, Pratap A, Agarwal S & Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), pp.182–97, 2002.

[EBE97] Ebeling C. E. An introduction to reliability and maintainability engineering. USA: McGraw Hill, 1997.

[EL99] Esquirol P, Lopez P. Lordonnancement. Economica, Paris, France, 1999.

[EMH01] Erickson M, Mayer A & Horn J. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems.First International Conference on Evolutionary MultiCriterion Optimization, Springer-Verlag.Lecture Notes in Computer Science, No 1993, pp. 681-695, 2001.

[FLI98] Flipo-Dhaenens C. Optimisation d'un réseau de production et de distribution. Thèse de doctorat, INPG de Grenoble , p 197,1998.

[GIA88] Giard V. Gestion de production, Paris, Ed. Economica, 1988.

[**GIA03**] Giard V. Gestion de la production et des Flux, Paris, Ed.Economica, 1229 p, 2003.

[GJ88] Gupta, Jatinder N. D. Two-stage hybrid flow shop Scheduling Problem. Journal of the Operational Research, 39, pp.359-364, 1988.

[GKLWS02] Gupta J.N.D, Krger K, Lauff V, Werner F & Sotskov Y.N. Heuristics for hybrid flow shops with controllable processing times and assignable due dates. *Computers and Operations Research*, 29, pp.1417-39, 2002.

[GM05]Gratacap A & Médan P. Management de la production. Dunod, 2005.

[GOL89] Golberg D. Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing, 1989. Paru

[GR87] Goldberg D E, Richardson J .Genetic Algorithms with Sharing for Multimodal Function Optimization. Dans Genetic Algorithms and their Applications: Proceedings of the Second International,Conference on Genetic Algorithms, pp.41-49,Lawrence Erlbaum,1987.

[HAR95] Haran G. Méthode P.E.R.T : Gestion et ordonnancement des projets par la méthode du chemin critique. EYROLLES, 1995.

[HHD07] Hadda H, Hajri-Gabouj S, Dridi N. Etude de flow shop hybride à deux étages avec machines dédiées sous contraintes d'indisponibilité. acte de la conférence scientifique conjointe en recherche opérationnelle et aide à la décision FRANCORO V/ROADEF, 2007.

[HLV96] Hoogeveen J.A, Lenstra J.K & Veltman B. Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. European Journal of Operational Research, 89,p.172-175, 1996.

[HNG94] Horn J, Nafpliotis N & Goldberg D E. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Vol. 1, pp. 82-87, Piscataway, New Jersey,1994.

[HOL75] Holland J. Adaptation in natural and artifical systems. Cambrige, Mass : MIT press, 1975.

[HOO05] Hoogeveen H. Multicriteria scheduling. European Journal of Operational Research, 167, pp.592-623, 2005.

[HY10] Hnaien F, Yalaoui F. Optimisation Biobjectif d'Ordonnancement et de Maintenance d'un Atelier Flow-Shop. 8e Conférence Internationale de MOdélisation et SIMulation « Evaluation et optimisation des systèmes innovants de production de biens et de services », MOSIM, Hammamet (Tunisie), 2010.

[IM98] Ishibuchi H, & Murata T. A multi-objective genetic local search algorithm and its application to flow shop scheduling. IEEE Transactions on Systems, Man and Cybernetics, 28, pp.392-403, 1998.

[JAM04] Jamali M A. Modélisation et validation de politiques optimales de maintenance preventive. Thèse de doctorat, Faculté des sciences et de génie, Université Laval, Québec, Canada, 2004.

[JRCW08] Jungwattanakit J, Reodecha M, Chaowalitwongse P, Werner F. Algorithms for Flexible Flow Shop Problems with Unrelated Parallel Machines, Setup Times, and Dual Criteria. International Journal of Advanced Manufacturing Technology (in press) , 2008.

[JZTO9] Jabbarizadeh F ,Zandieh M ,Talebi D.Hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints . Computers & Industrial Engineering , 57, pp. 949–957, 2009.

[KAA04] Kaabi J. contribution a l'ordonnancement des activités de maintenance dans les systèmes de production. Thèse de doctorat en Automatique et Informatique de l'université de FRANCHE-COMTE, France, 2004.

[KOS92] Bart Kosko. Neural networks and fuzzy systems. A dynamical systems approach to machine intelligence, Prentice-Hall Inc, 1992.

[KVZ02] Kaabi J, Varnier C & Zerhouni N. Heuristics for scheduling maintenance and production on a single machine. IEEE Conference on Systems, Man and Cybernetics. October 6–9 Hammamet, Tunisia, 2002.

[KVZ03] Kaabi J, Varnier C & Zerhouni N. Un algorithmes génétique pour le problème d'ordonnancement production et maintenance dans un Flow Shop. Laboratoire d'automatique de Besançon, France, 2003.

[LC00] Lee C Y and Chen Z L. Scheduling jobs and maintenance activities on parallel machines. Naval Research Logistics, 47:145–165, 2000.

[LCTH09] Lau HCW, Chan TM, Tsui WT, Ho GTS. Cost optimization of the supply chain network using genetic algorithms. IEEE Transactions on Knowledge and Data Engineering, 99(1), 2009.

[LLRS93] Lawler E L, Lenstra J K, Rinnooy Kan A H G & Shomoys D B. Sequencing and scheduling: algorithms and complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P. Zipkin,

editors, Handbooks in Operations Research and Management Science: Logistics of Production and inventory, 1993.

[LYAC12] Li X, Yalaoui F, Amodeo L, Chehade H. Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. J Intell Manuf,23,pp. 1179–1194,2012.

[MER09] MERABTI H. Etude des systèmes flous à intervalle. Mémoire de magistère, Université MENTOURI de Constantine, 2009.

[MFZ11] Moradi E, Fatemi Ghomi S M T & Zandieh M.Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem,Expert Systems with Applications, 38 (6), pp.7169–7178, 2011

[MMA12] Mokhtari H, Mozdgir A & Abadi I N K. A reliability/availability approach to joint production and maintenance scheduling with multiple preventive maintenance services. International Journal of Production Research, 50:20, pp.5906-5925, 2012.

[MZ10] Moradi E & Zandieh M.Minimizing the makespan and the system unavailability in parallel machine scheduling problem: a similarity-based genetic algorithm. International Journal of Advanced Manufacturing Technology, 51 (5–8),pp.829–840, 2010.

[PAR85] Parunak, H.V.D. Manufacturing experience with the contract net. *In Proceedings of the Fith Workshop on Distributed Artificial Intelligence*, 1985.

[PGE04] Przybylski A, Gandibleux X & Ehrgott M. Seek and cut algorithm computing minimal and maximal complete efficient solution sets for the biobjective assignment problem. In 6th Int. Multi-Objective Programming and Goal Programming conf , MOPGP'04, 2004.

[RA09] Ruiz R, Allahverdi A. Minimizing the bicriteria of makespan and maximum tardiness with an upper bound on maximum tardiness. *Computers & Operations Research*, 36(4), pp.1268-83, 2009.

[RGM07] Ruiz R, García-Diaz J C, and Maroto C. Considering scheduling and preventive maintenance in the flowshop sequencing problem. Computers & Operations Research, 34(11), pp.3314–3330, 2007.

[RIN76] Rinnooy Kan A.H.G. Machine Scheduling Problems: Classification, Complexity and Computations. Martinus Nijhoff, The Haque, 1976.

[ROB88] Roboam M. Modèles de référence et intégration des méthodes d'analyse pour la conception des systèmes de production. Thèse de doctorat soutenue à l'université de Bordeaux, France, 1988. In [BEN05]

[RUE89] Ruegg A **.** Processus stochastiques [Livre]. - 1015 Lausanne,Suisse : Press polytechniques romandes, 1989.

[RV10] Ruiz R &Vázquez-Rodríguez J A. The hybrid flow shop scheduling problem. European Journal of Operational Research, 205 :1–18.46,2 010.

[SD94] Srinivas N et Deb K. *Multiobjective Optimization using Nondominated* Sorting in Genetic Algorithms. Evolutionary Computation, Vol. 2, No 3, pp.221-248, 1994.

[SMI06] Smith D. Fiabilité, Maintenance et Risque [Livre]. - Paris : Dunod, 2006.

[**SS00]** Sloan T W and Shanthikumar J G. Combined production and maintenance scheduling for a multiple product, single machine production system. Production and Operation Management, 9(4), pp.379–399, 2000.

[SW91] Stewart B S & White C C. Multiobjective A*. Journal of the ACM, 38(4), pp.775–814, 1991.

[THSC08] Timmis J, Hone A, Stibor T, Clark E. Theoretical advances in artificial immune systems. Vol. 403, pp.11–32, 2008.

[UT95] Ulungu E L and Teghem J. The two phases method : An efficient procedure to solve biobjective combinatorial optimization problems. Foundation of computing and decision science, 20 , pp.149–156, 1995.

[VBP99] Vignier A, Billaut J.C & Proust C. Les Problèmes d'ordonnancement de type flowshop hybride: état de l'art. RAIRO-RO, 33(2), pp.117-83, 1999.

[VBPT96] Vignier A, Billaut J C, Proust C & Tkindt V. Resolution of some 2-stage hybrid flowshop scheduling problems. IEEE International Conference on Systems, Man and Cybernetics, 4, pp.2934-2941, 1996.

[VCP97] Vignier A, Commandeur C & Proust P. New lower bound for the hybrid flowshop scheduling problem. In Proceedings of IEEE sixth International Conference on Emerging Technologies and Factory Automation (ETFA 97), pp.446-451,1997.

[VIL91] Villemeur A. Reliability, availability, maintainability and safety assessment. USA:Wiley, 1991.

[WC99] Weinstein L & Chung C H. Integrating maintenance and production decisions in a hierarchical production planning environment. Computers and Operations research, 26, pp.1059‑1074, 1999.

[WJ04] Wei Wang & John L. Hunsucker. makespan distributions in flow shops with multiple processors. Journal of the Chinese Institute of Industrial Engineers, Vol. 21, No. 3, pp.242-250 , 2004.

[XSL08] Xu D, Sun K & Li H. Parallel machine scheduling with almost periodic non preemptive maintenance and jobs to minimize makespan. Computers and Operations Research, 35: pp.1344–1349, 2008.

[XW05] Xie J & Wang X. Complexity and algorithms for two-stage flexible flowshop scheduling with availability constraints. Computers and Mathematics with Applications, 50 (10–12), pp.1629–1638, 2005.

[ZAD65] Zadeh L.A. Fuzzy Sets. Information and Control, 8, pp.338–353, 1965.

[ZDT02] Zitzler E, Deb K & Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results, Evolutionary Computation 8 (2):173–195, 2000.

[ZIT99] Zitzler E. Evolutionary algorithms for multi- objective optimization: Methods and applications. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999.

[ZLT02] Zitzler E, Laumanns M & Thiele L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm, in K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou and T. Fogarty (eds.), EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, CIMNE, Barcelona, Spain, pp.95-100, 2002.

ANNEXES

ANNEXE A

PSEUDO CODES DES ALGORITHMES

A1. Pseudo code de l'algorithme NSGA-II

Pseudo code de l'algorithme NSGA-II

Etape1: créer aléatoirement les populations initiales P_0 et Q_0 de taille N.
Etape 2: **Tant que** (*le critère d'arrêt n'est pas vérifié*) **faire**
Etape 3: Création de la population $R_t = P_t \cup Q_t$.
Etape 4: Construire les différents fronts Fi de Rt avec la procédure de Ranking.
Etape 5:Mettre $P_{t+1} = \emptyset$, $i = 0$.
Etape 6: **Tant que** $|P_{t+1}| + |F_i| < N$ **faire**
Etape 7: $P_{t+1} = P_t \cup F_i$
Etape 8: $i = i + 1$
Etape 9: **Fin Tant que**
Etape 10: Inclure dans P_{t+1} les $(N - |P_{t+1}|)$ individus de F_i les mieux répartis au sens de la distance de Crowding.
Etape 11: Création d'une nouvelle population enfant (Q_{t+1}) à partir de P_{t+1} par sélection, croisement et mutation.
Etape 12: **Fin Tant que**

A2. Pseudo code de l'algorithme WSGA

Pseudo code de l'algorithme WSGA

Etape 0: Générer aléatoirement une population initiale de taille N_{pop}.
Répéter
Etape 1: Evaluer les objectifs pour chaque individu dans la population courante.
Ensuite mettre à jour la population secondaire avec les solutions non dominées
obtenues de la population courante.
Etape 2: Sélectionner $(N_{pop} - N_{elite})$ paires de parents en répétant les procédures suivantes:
a) Spécifier aléatoirement les poids α *et* β.
b) Sélectionner une paire de parents en se basant sur la fonction scalaire (de fitness)
donnée par l'Equation (4.1).
La sélection par tournoi binaire est utilisée.
Etape 3: Effectuer les opérateurs évolutionnaires (croisement et mutation) à chacune des
$(N_{pop} - N_{elite})$ paires de parents sélectionnés pour générer deux nouvelles
solutions pour chaque paire.
Etape 4: Sélectionner aléatoirement N_{elite} solutions à partir de la population secondaire.
Ensuite ajouter leurs copies aux $(N_{pop} - N_{elite})$ solutions générées à l'étape 3 pour
Construire une population de solutions.
Jusqu'à (la condition d'arrêt est vérifiée).

A3. Pseudo code de l'algorithme MOIA2

Pseudo code de l'algorithme MOIA2

Etape0 : Initialisation

– Création d'une population initiale des anticorps *P* de taille *N*

– Créer une population secondaire S initialement vide

Répéter

Etape 1: *Evaluation*

– Evaluer les objectifs pour chaque anticorps dans la population courante.

– Spécifier aléatoirement le poids α.

– Calculer pour chaque individu de la population d'anticorps ***P*** son affinité selon les équations (5.5) et (5.6).

Etape 2 : Mémorisation

– Mettre à jour la population secondaire avec les solutions non dominées obtenues de la population courante.

Etape 3 : Sélection et Clonage

- Sélection des anticorps ayant les meilleures affinités dans la population ***P.***
- Cloner chaque anticorps sélectionné proportionnellement à son affinité.
- Muter les anticorps clonés avec un degré anti proportionnel à leurs affinités.
- Placer les clones dans une nouvelle population d'anticorps.

Etape 4 : Suppression Clonale (Principe de génération des récepteurs)

- Eliminer les B % d'anticorps de plus faibles affinité de la nouvelle population
- Créer aléatoirement B% nouveaux anticorps à la place de ceux éliminés.
- Remplacer les anticorps éliminés par ceux crées.

Jusqu'à (*la condition d'arrêt est vérifiée*).

Etape5 : afficher les solutions non dominés dans la population secondaire comme approximation de l'ensemble Pareto optimale

A4. Pseudo code de l'algorithme FLC-MOIA1

Pseudo code de l'algorithme FLC-MOIA1

Entrées :

Gmax : nombre de générations
Popsize : taille de la population d'anticorps
B : pourcentage d'élimination des anticorps initial
T_{mut} : nombre de mutation initial
P_m : probabilité de mutation initiale

Sorties : ensemble Pareto optimal

Etape 1 : Initialisation

- créer aléatoirement une population ***P*** initiale des anticorps
- une population secondaire ***S*** initialement vide
- initialiser le nombre des générations gen =0

Etape 2 : **Test d'arrêt de l'Algorithme**

Si gen > Gmax aller à l'étape **10 Sinon** aller à l'étape **3**

Etape 3 : incrémenter le nombre de génération gen

Etape 4 : Evaluation

- calculer pour chaque individu de la population d'anticorps ***P*** son affinité selon la procédure du *Ranking* et *crowding*

Etape 5 : Mémorisation

- copier les individus non dominés (de rang = 1) dans la population secondaire ***S***

Etape 6 : Sélection et Clonage

- sélection des anticorps ayant les meilleures affinités dans la population ***P***
- cloner chaque anticorps sélectionné proportionnellement à son affinité
- muter les anticorps clonés avec un degré anti proportionnel à leurs affinités
- placer les clones dans une nouvelle population d'anticorps

Etape 7 : Suppression Clonale (Principe de génération des récepteurs)

- calculer pour chaque individu de la nouvelle population d'anticorps ***P*** son affinité selon la procédure du *Ranking* et *crowding*
- éliminer les B % d'anticorps de plus faibles affinité de la nouvelle population
- sélectionner aléatoirement B% d'anticorps de la population secondaire
- remplacer les anticorps éliminés par ceux sélectionnés (aléatoirement)

Etape 8 : Appel du contrôleur flou

Si *gen mod 10=0* **Alors**

- La valeur *f(t-1)-f(t-2)* et la diversité *d(t-1)* sont considérées comme entrées de contrôleur flou
- Fuzzification des variables d'entrée et de sortie
- Moteur d'inférence
- Defuzzification des variables de sortie
- Mise à jours des paramètres (P_m , T_{mut} et *B* %) de l'algorithme

END Si

Etape 9 : Retour à l'étape **2**

Etape 10 : Approximation du front Pareto optimal

- afficher les solutions non dominés dans la population secondaire comme approximation de l'ensemble Pareto optimale
- fin de l'Algorithme