République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université M'Hamed Bougara de Boumerdès

Faculté des Sciences de l'Ingénieur

Département de Maintenance Industrielle

MEMOIRE DE MAGISTER

Spécialité **: Maintenance Industrielle**

Option **: Maintenance des Systèmes Mécaniques**

THEME

Optimisation des Problèmes d'Ordonnancement Multi-Objectifs dans les Systèmes de Production

Présenté par : **BENIKHLEF Taha**.

Soutenu le 18- 12-.2005 devant le jury :

MM :

▪ **A. BENFDILA**	Professeur	U. Tizi. Ouzou	**Président**
▪ **S. ADJERID**	MC	U. Boumerdès	**Examinateur**
▪ **K. MOHAMMEDI**	MC	U. Boumerdès	**Examinateur**
▪ **K. KHALFI**	CC	U. Boumerdès	**Examinateur**
▪ **D. BENAZZOUZ**	Professeur	U. Boumerdès	**Rapporteur**

Dédicaces

À la mémoire de mon cher père, qui m'a donné le goût du savoir.

À la mémoire de mes grands- parents.

À ma chère mère qu'aucun mot ne peut décrire ma gratitude envers elle.

À mes soeurs dont leur soutien ne ternie jamais.

À mon frère qui aura à me supporter de nouveau.

À Biba pour sa patience et sa complicité.

À l'équipe des petits pour la joie qu'ils apportent.

Remerciements

J'exprime ma vive reconnaissance à Monsieur Djamel BENAZZOUZ, Professeur à la faculté de Sciences de l'Ingénieur, Université de Boumerdès, pour avoir encadré mes travaux de mémoire et pour le soutien qu'il m'a accordé.

Je tiens à remercier également :

- Monsieur BENFDILA Arazki, Professeur à l'université de Tizi-Ouzou.
- Monsieur ADJERID Smail, Maître de Conférences à l'Université de Boumerdès.
- Monsieur Kamel MOHAMMEDI, Maître de Conférences à l'Université de Boumerdès.
- Monsieur Kamel KHALFI, Chargé de Cours à l'Université de Boumerdès.

Pour l'honneur qu'ils me ferons en participant à mon Jury de mémoire.

Je remercie particulièrement, M. Jean Christophe FAUROUX, Maître de conférences à l'institut Français de la Mécanique Avancée, qui m'a décrit le charme de l'optimisation, et qui a toujours su créer une excellente ambiance de travail fertile en discussions et en réflexions.

Merci à toute l'équipe d'Autobar Packaging pour son accueil, sa bonne humeur et son ambiance. Je veux remercier El Hadj Mourad, Jean Marc et Daniel pour leurs disponibilités, pour leurs critiques et conseils, sans oublier le vieux Gérard, Samuel, Serge, les deux Gérôme, et toute l'équipe, pour l'ambiance conviviale qu'ils y mettent.

Merci aussi à tous les autres amis que j'ai côtoyé au cours de ces dernières années : Sadek, Derradji, et Bousbous sans oublier Ahmed le pakistanais.

Résumé

Le travail de recherche présenté dans ce mémoire consiste à l'étude d'une problématique industrielle concernant l'optimisation par Algorithmes Génétiques (AG) de l'ordonnancement dans un Atelier d'Extrusion Plastique.

Cet atelier se trouve actuellement face aux grands problèmes liés à la matière première de polyéthylène, matière très chère suite aux chocs pétroliers. L'objectif principal est de trouver le meilleur ordonnancement qui propose au responsable de la production un ensemble d'ordres de passage entre les commandes des clients sur les différentes machines, de telle façon à baisser le coût lié aux déchets de transition, minimiser en revanche le temps nécessaire pour finir les tâches des différentes commandes.

Les AG appartiennent aux approches d'optimisation les plus performant, cela nous a amené à exploiter leur robustesse et leur puissance pour notre étude en fixant comme critère à optimiser le temps et le coût. Ces deux critères ont été réduits considérablement lors de la phase de simulation.

Mots Clés : Algorithmes Génétiques, Ordonnancement, Optimisation, Atelier d'Extrusion Plastique.

Abstract

The research work presented in this thesis deals of the study of an industrial problems concerning Genetic Algorithms Optimization (GA) of scheduling in a plastic extrusion workshop.

This workshop is currently face to major problems involve the raw material of polyethylene, matter which is very expensive following the oil crises. The main objective of this study, is to find the best scheduling which proposes to the decision maker a whole of passage orders between the customer requests on the various machines, in such way to minimize the cost related to the transition waste, on the other hand, to minimize the time necessary to finish the tasks of the various requests.

Genetic Algorithms belong to the optimization approaches most powerful, that led us to exploit their robustness and their power for our study while fixing like criterion to optimize, times and cost. These two criteria were considerably reduced on the simulation phase.

Key Words: Genetic Algorithms, Scheduling, Optimization, Plastic Extrusion Workshop.

ملخص

إن البحث المقدم في هذه المذكرة يختص بدراسة إشكالية صناعية تتمثل في ضبط التوصيف الأرجح لورشة صناعة فيلم البلاستيك ,و هذا باستعمال الخوارزميات الجينية.

في الوقت الحالي هذه الورشة تواجه عدة مصاعب تتمثل في ارتفاع أسعار مادة البولي ايثيلين المادة الأولية .حيث أن الهدف الرئيسي من خلال هذا ,هو إيجاد احسن توصيف لانتاج مختلف الطلبيات عن طريق اقتراح أنجع الأوامر لتقنية مهام الصنع بطريقة تسمح بتخفيض المادة الانتقالية بين مختلف الطلبيات والتكلفة المتعلقة بذلك .

فالخوارزميات الجينية التي أعطت نتائج جيدة أدت بنا إلى استغلالها في هذه الدراسة بهدف تخفيض عاملي الوقت والتكلفة .حيث سمحت النتائج المحصل عليها بتخفيضات متميزة لهذين الأخيرين.

Table des Matières

Liste des Figures

Liste des Tableaux

Introduction Générale

Dans les systèmes de production, l'ordonnancement opère au niveau court terme pour la détermination des dates de lancement des opérations et des ressources qui leur sont affectées en vue d'atteindre certains nombres d'objectifs économiques, parmi lesquels on peut citer :

- La réduction des retards.
- La réduction des avances.
- La réduction des délais de fabrication.
- La réduction des coûts de fabrication, etc.

Tous ces facteurs permettent d'avoir un rendement optimal de la production.

De nos jours, les entreprises doivent être compétitives et ont besoin pour cela d'outils performants pour gérer la complexité de leur organisation. La gestion de production appartient à ces outils permettant le contrôle et la planification du processus de production. Parmi les fonctions de la gestion de production, la résolution du problème d'ordonnancement n'est pas encore complètement maîtrisée [Giar 03], [Lope 99].

Dans ce mémoire le travail présenté a été initié par une problématique industrielle présentée par la société Autobar Packaging site de France.

Cette problématique, détaillée au chapitre IV, concerne l'optimisation de la fonction ordonnancement, au niveau de l'atelier d'extrusion plastique, cet atelier se trouve actuellement face aux grands problèmes liés à la matière première de polyéthylène, matière très chère suite aux chocs pétroliers. La nature de production du plastique pénalise fortement la production. D'une part le passage d'une référence de commande à une autre crée une quantité de plastique transitoire à recycler, d'autre part, les transitions entre les différentes commandes introduisent des temps non productifs (perte de temps machine), ils peuvent coûter très chers à cause du personnel supplémentaire.

Dans cette étude nous avons proposer un outil performant basé sur les principes de l'intelligence artificielle pour la répartition de l'ensemble du travail (commandes) sur les machines, en respectant au mieux un ensemble de contraintes (de précédences, de ressources,….), et en visant un objectif (minimisation des coûts liés au changement de commandes, et de temps lié à la fin d'exécution des tâches de production '*Makespan'*).

L'objectif principal est de trouver un bon ordonnancement qui propose au responsable de la production un ensemble d'ordres de passage entre commandes sur différentes machines de telle façon à baisser le coût lié aux déchets de transition, minimiser en revanche le temps nécessaire pour finir les tâches des différentes commandes.

Ne pouvant pas matériellement calculer tous les ordonnancements possibles pour des raisons de coût et de temps, il existe des algorithmes partiellement aléatoires qui permettent d'améliorer la recherche d'optima dans des problèmes complexes [Lope 01], tels que la méthode du gradient, la méthode du recuit simulé entre autre, et aussi des algorithmes basés sur les lois de la génétique appelés Algorithmes Génétiques. Ces derniers sont basés sur le principe de l'évolution d'une population candidate. Celle-ci est composée d'un ensemble de chromosomes. Dans notre cas, il s'agit de la représentation d'un ordonnancement en un chromosome. Par application des lois de la génétique (croisement, mutation et sélection), il est possible d'augmenter les performances des chromosomes à travers les générations en sélectionnant les éléments les plus performants par rapport à une fonction objectif jusqu'à atteindre la bonne qualité d'ordonnancement qui correspond à une solution optimale.

Cette méthode peut comporter un ou plusieurs objectifs à améliorer, dans notre étude, l'entreprise souhaite minimiser les objectifs temps *(Makespan)* et coût.

Au premier chapitre, nous verrons tout d'abord les différentes définitions des problèmes d'ordonnancement, en quoi consiste l'ordonnancement en gestion de la production ainsi que les différents éléments constituants l'ordonnancement.

Puis, nous proposerons, au chapitre 2, un état de l'art sur les différentes méthodes classiques de résolution des problèmes d'ordonnancement en insistant sur le cas des machines parallèles.

Le troisième chapitre sera consacré à présenter les Algorithmes Génétiques. Ces techniques d'optimisation seront appliquées à notre problème. Nous présenterons les définitions inspirées de la génétique naturelle.

La représentation de notre problème au moyen des Algorithmes Génétiques sera l'objet du dernier chapitre. En premier lieu nous décrirons notre atelier d'une façon simple et encore une fois proche de la réalité. Nous verrons les différentes étapes pour une implémentation qui respecte le problème choisi ainsi que les descriptions des chapitres précédents. Nous expliquerons la méthode utilisée pour la résolution du problème d'ordonnancement et son optimisation avec les Algorithmes Génétiques, nous verrons aussi le fonctionnement du programme ainsi que la représentation des résultats sous diagrammes et graphes. Enfin nous commentons nos résultats tout en appréciant l'avantage mené par notre méthode d'optimisation choisie, puis nous présentons quelques perspectives de ce travail.

Chapitre I

Description des Problèmes d'Ordonnancement

Dans ce chapitre, nous présenterons les notions fondamentales concernant les problèmes d'ordonnancement. Pour cela il convient de définir ce qu'est un problème d'ordonnancement, de montrer la diversité de ces problèmes en indiquant les principes de base de ces problèmes.

1. Introduction

L'ordonnancement est une fonction essentielle en gestion de la production. C'est un problème difficile en raison du nombre de calculs à effectuer. Le problème d'ordonnancement a pour objet de définir, prévoir et coordonner l'ensemble des ressources physiques et humaines nécessaires à la production. La solution de ce problème devra permettre d'optimiser l'utilisation des ressources tout en respectant les délais de réalisation fixés. Cette solution doit donc répondre aux questions suivantes : Qui ? Quoi ? Quand ? Combien ? [Jave 97]. La réponse sera obtenue lorsqu'on saura quelle est la séquence de passage des travaux et les machines sur lesquelles ils doivent y passer, tout en optimisant un ou plusieurs critères donnés [Hent 99].

Notons aussi que nous allons évoquer la problématique de l'ordonnancement, les éléments de base des problèmes d'ordonnancement et les concepts de base intégrant notamment les principes généraux de ces problèmes.

2. Définition du problème d'Ordonnancement

Les problèmes d'ordonnancement ont été largement étudiés dans la littérature et par conséquence on trouve des notations et des définitions très diverses telles que :

- *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu des contraintes temporelles (délais, contraintes d'enchaînement,…etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches [Lope 01].*

- ***Ordonnancer** un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leur date de début [Goth 93].*

- *Mathématiquement, c'est un problème d'optimisation sous contraintes. Il s'agit de déterminer, à l'intérieur de l'ensemble des solutions réalisables (c'est-à-dire vérifiant les contraintes de problèmes), un ordre de passage des tâches qui minimise une (ou parfois plusieurs) fonction objectif donnée [Goth 04].*

- *Programmation coordonnée d'une ou plusieurs ressources pour réaliser un ensemble de tâches ou d'opérations dans un laps de temps donné [Giar 88].*

- *Détermination conjointe des dates d'exécution d'un ensemble d'opérations et de ressources mobilisées dans cette exécution [Giar 03].*

On peut dire également que l'ordonnancement est une fonction permettant de déterminer les priorités de passage des produits ou des tâches et leur affectation sur les différentes machines.

Un ordre de passage des tâches constitue une solution au problème d'ordonnancement. Il décrit l'exécution des tâches et l'allocation des ressources au cours du temps, et vise à satisfaire un ou plusieurs objectifs. Plus précisément, on parle du problème d'ordonnancement lorsqu'on doit déterminer les dates de début et les dates de fin des tâches, alors qu'on réserve le terme de problème de séquencement au cas où l'on cherche seulement à fixer un ordre relatif entre les tâches qui peuvent être en conflit pour l'utilisation des ressources. Un ordonnancement induit nécessairement un ensemble unique de relations de séquencement.

3. Ordonnancement et typologie organisationnelle

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 [Giar 03].

- **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.1.1, décrit un exemple de découpage fonctionnel de la gestion du travail dans un atelier et précise la place de l'ordonnancement. Ce schéma introduit également les fonctions *planification* et *lancement.* En amont de l'ordonnancement, la planification assure le calcul des besoins en composants et détermine un plan d'approvisionnement et un plan de charge (volumes de production par période). En aval de l'ordonnancement, le lancement permet de sélectionner parmi l'ensemble des ordres de fabrication ceux qu'il faut effectivement engager en fabrication. Pour cela, les ordres de fabrication sont transformés en ordres sur les pièces à usiner ou produits à fabriquer, éventuellement après regroupement et dimensionnement en lots de production. La quantité de travail qui est fixée par le lancement dépend de la complexité des nomenclatures (liste des pièces constitutives d'un produit) et des gammes (suites d'opérations à effectuer pour réaliser un produit).

La coordination au sein des structures hiérarchisées s'exerce à priori par un contrôle vertical : afin d'assurer la cohérence - ou consistance - du niveau inférieur vis-à-vis du niveau supérieur, celui-ci doit considérer les décisions du niveau supérieur comme des contraintes. En contrepartie, chaque niveau dispose d'une certaine autonomie grâce à laquelle il lui est possible d'absorber des aléas. L'impossibilité de respecter ces contraintes se traduit par une remise en cause et l'émission de nouvelles directives par le niveau supérieur. Des remises en cause incessantes traduisent un manque de robustesse des décisions du niveau supérieur. Au niveau de l'ordonnancement, l'impossibilité d'obtenir un plan assurant les volumes demandés de production, peut avoir comme origine une estimation trop grossière des capacités réelles de l'atelier ou une sous-estimation de sa charge réelle lors de la planification.

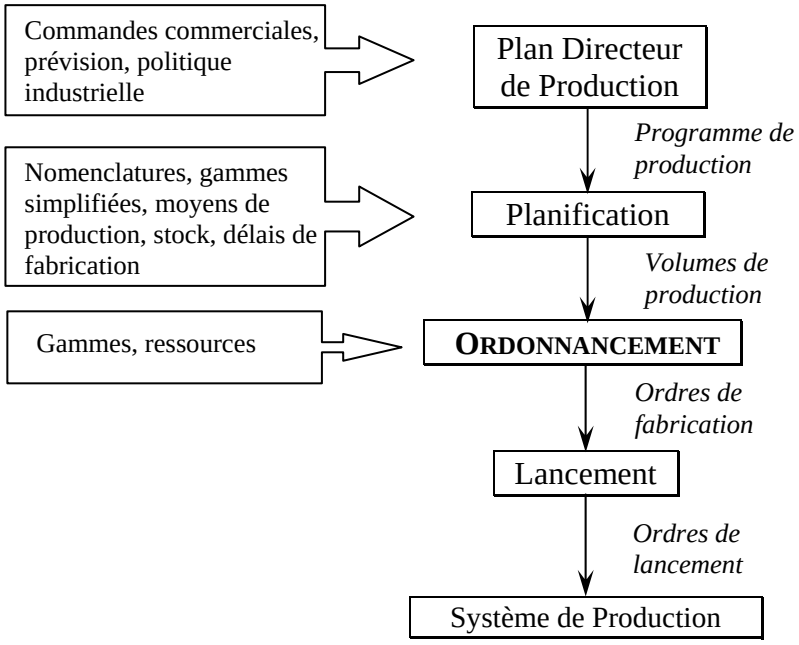

Figure.1.1. Organisation Hiérarchique Fonctionnelle

4. Problématique de l'Ordonnancement

L'environnement actuel des entreprises est caractérisé par des marchés soumis à une forte concurrence et sur lesquels les exigences et les attentes des clients deviennent de plus en plus fortes en termes **de qualité, de coût** et **de délais** de mise à disposition. Cette évolution se renforce par le développement rapide des nouvelles technologies de l'information et de la communication qui permettent une relation directe entre les entreprises et entre les entreprises et leurs clients.

Dans un tel contexte, la performance de l'entreprise se construit selon deux dimensions [Lope 01] :

- **Une dimension technologique** qui vise à développer les performances intrinsèques des produits mis sur le marché afin de satisfaire aux exigences de qualité et de réduction du coût de possession des produits, ici L'innovation technologique joue un rôle important et peut constituer un élément différenciateur déterminant pour la pénétration et le développement d'un marché. Il faut noter qu' à ce niveau l'évolution technologique rapide des produits et les exigences de personnalisation de ces produits qu'attendent les marchés mènent les entreprises à abandonner souvent les productions de masse pour s'orienter vers des productions de petites ou moyennes séries, sinon à la commande. Ceci leur impose donc d'avoir des systèmes de production flexibles et évolutifs, capables de s'adapter aux exigences et aux besoins des marchés d'une façon rapide et efficace.

- **Une dimension organisationnelle** qui vise à développer la performance en termes de durée des cycles de fabrication, respect des dates de livraison prévues, gestion des stocks et des en-cours, adaptation et réactivité face aux variations du cahier commercial… Cette dimension joue un rôle de plus en plus important dans la mesure où les marchés sont de plus en plus versatiles et évolutifs et exigent des temps de réponse des entreprises de plus en plus courts. Il faut donc disposer des méthodes et des outils de plus en plus performants pour **l'organisation** et **la conduite** de la production.

Cette organisation de la production doit être vue non seulement au niveau de l'entreprise elle-même, mais aussi au niveau de sa position au sein d'une chaîne logistique dont elle constitue l'un des maillons, conduisant ainsi à une entreprise globale « virtuelle » qui doit être orientée vers la satisfaction du besoin des clients aux meilleures conditions.

Pour atteindre ces objectifs, l'organisation repose en général sur la mise en œuvre d'un certain nombre de fonctions (*la planification, la programmation, l'ordonnancement, la conduite et la commande*) dont la fonction ordonnancement joue un rôle essentiel.

Cette fonction vise en effet à organiser l'utilisation des ressources technologiques et humaines présentes dans les ateliers de l'entreprise pour satisfaire soit directement

les demandes des clients, soit les demandes issues d'un plan de production préparé par la fonction de planification de l'entreprise. Compte tenu de l'évolution des marchés et de leurs exigences, cette fonction doit organiser l'exécution simultanée de multiples travaux à l'aide de ressources polyvalentes disponibles en quantités limitées, ce qui constitue un problème complexe à résoudre. De plus, c'est cette fonction qui contribue à la réalisation des produits. Son efficacité et ses défaillances conditionnent donc fortement la relation de l'entreprise avec ses clients. Bien sûr, au sein des entreprises, cette fonction a toujours existé, mais elle est aujourd'hui confrontée à des problèmes de plus en plus complexes à résoudre, à cause du grand nombre de travaux qu'elle doit simultanément réaliser, avec pour chacun d'eux des délais de réalisation de plus en plus courts.

Apporter des solutions efficaces et performantes aux problèmes d'ordonnancement ainsi définis constitue sûrement un enjeu économique important. Malgré la simplicité de la formulation d'un tel problème, il faut bien constater qu'à ce jour, on ne dispose pas de « La » méthode susceptible de résoudre tous les cas de figures. Il existe un certain nombre de problèmes génériques qui se différencient par les caractéristiques des travaux à réaliser ou des ressources disponibles pour les réaliser.

Des méthodes spécifiques peuvent alors être associées à la résolution de chacun de ces problèmes génériques, ces méthodes spécifiques étant soit une interprétation particulière pour le problème considéré d'une méthode générale, soit une méthode particulière, dédiée au problème considéré. La résolution d'un problème concret commence alors par l'identification du problème générique auquel on peut l'associer, suivi de la sélection de la « ou des » méthode(s) adaptée(s) à la résolution de ce problème.

Il faut également remarquer que le problème de décision associé au problème d'ordonnancement appartient à la classe des problèmes combinatoires [Lope 01].

Par suite, la résolution par des méthodes exactes est peu réaliste pour des problèmes de grande dimension, ce qui justifie le recours à des méthodes heuristiques performantes (on note que les méthodes approchées ou heuristiques sont des méthodes pour lesquelles l'enjeu consiste à trouver des solutions de plus en plus proches de la solution optimale). Tout ceci explique, que la recherche sur les problèmes d'ordonnancement reste toujours d'actualité et suscite de nombreux travaux.

5. Les Différents éléments d'un Problème d'Ordonnancement

5.1. Les Tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début t_i ou de fin c_i, dont la réalisation est caractérisée par une *durée de traitement* $p_i = c_i - t_i$, et par *l'intensité* a_{ik} avec laquelle elle consomme certains moyens k, ou ressources.

Pour simplifier, on supposera que pour chaque ressource requise, cette intensité est constante durant l'exécution de la tâche [Lope 01], [Esqu & Lope 99].

Une **tâche** est un ensemble d'opérations mobilisant des ressources et réalisant un progrès significatif de l'état d'avancement du projet compte tenu du niveau de détail retenu dans l'analyse du problème [Giar 88].

Généralement, les notations utilisées [Carl & Chré 88] pour caractériser une tâche, qu'on note '*i*', sont les suivantes :

- Une date de disponibilité r_i : l'exécution de la tâche *i* ne peut pas débuter avant cette date.
- Une date échue notée d_i : la tâche *i* doit être achevée avant cette date.
- La durée opératoire –*de traitement*- notée p_i.

On note [r_i, *di*], l'intervalle temporel dans lequel la tâche *i* devrait s'exécuter.

Dans certains problèmes, les tâches peuvent être exécutées par **préemption**, qui désigne la possibilité d'interrompre une tâche pour en passer à une autre avant de la reprendre plus tard. Dans d'autres problèmes, on ne peut pas interrompre une tâche une fois commencée. On parle alors respectivement de problèmes ***préemptifs*** et ***non préemptifs*** [Giar 03], [Bruc 99], [Bruc & Krav 99], [Rinn 76], [Bapt 00].

Ce qui différencie une tâche d'une autre, en plus de ses caractéristiques technologiques, est sa disponibilité dans le temps. On parlera d'un problème d'ordonnancement statique (*ou déterministe*) [Esqu & Lope 99], [Cram & Al 97] dans le cas où toutes les tâches sont connues à l'avance avec leurs dates de disponibilité (en outre, on *dit qu'un problème est statique lorsqu'on peut déterminer les dates de réalisation de différentes opérations de toutes les tâches sur les ressources données, compte tenu des contraintes à la fois temporelles et l'utilisation de ressources* [Lope 01]).

Un problème d'ordonnancement est *statique* si l'ensemble des informations nécessaires à sa réalisation est fixé à priori et n'est pas remis en cause durant la résolution [Esqu & Lope 99]).

Dans le cas contraire,on parle de problèmes *d'ordonnancement dynamiques* [Mich & Dani 97] (*on dit que l' ordonnancement est dynamique –temps réel- quand les décisions d'ordonnancement des opérations sur une machine sont prises une par une chaque fois qu'une machine se libère ou qu'une opération devient disponible* [Lope 01]).

Certains problèmes d'ordonnancement sont dits stochastiques lorsque des caractéristiques techniques sont sujettes à des lois de probabilité. Ces caractéristiques sont alors estimées [Carl & Chré 88].

Dans notre cas d'étude, on prend en considération le premier type d'ordonnancement (statique) où toutes les tâches sont connues d'avance, on parlera d'ordonnancement déterministe.

5.2. Les Opérations

En général, la plupart des tâches donnent lieu à l'utilisation de différentes ressources et se décomposent donc en plusieurs opérations.

Dans le cas d'un problème d'ordonnancement sur une machine unique ou atelier unique, on parle de tâches mono-opération. L'opération est caractérisée par son temps d'exécution, appelé aussi *durée opératoire*. Parfois, une opération peut également être caractérisée par son temps de préparation sur la machine, ou encore les temps de montage et de démontage de la pièce (*matérialisant l'opération ou l'outillage*), les temps de transport et d'attente.

La séquence des opérations est décrite dans ce qu'on appelle la *gamme* d'une tâche [Hent 99] (voir définition ci-dessous) et dans notre cas, il existe une seule opération par tâche.

5.3. Les Gammes

La gamme est un document décrivant en détail la séquence d'opérations de fabrication, d'assemblage, d'inspection ou de transport d'un composant ou d'un produit fini [Giar 88].

Les gammes de fabrication décrivent la succession des opérations à réaliser pour passer d'une matière au composant ou du produit semi-fini au produit fini ; c'est le processus d'élaboration. On trouve dans la littérature non spécialisée, l'appellation de routage de la tâche et dans la littérature anglophone le terme de *routing*, il existe plusieurs sortes de gammes [Hent 99], parmi lesquelles on distingue :

- *Les gammes techniques* dues essentiellement au choix prédéterminé de certaines tâches à passer sur des machines spécifiques.

- *Les gammes libres* qui sont l'une des principales caractéristiques des problèmes d'ordonnancement de type *Open Shop***.** En effet, quand l'exécution des opérations est indépendante de l'ordre, il n'existe aucune contrainte de succession.

- *Les gammes linéaires* où l'ordre d'exécution des opérations entièrement imposé et prédéterminé. Ce genre de gammes est présent dans le cas du *Job Shop* (*chaque tâche a sa propre route à suivre autrement dit toutes les tâches ont des gammes prédéterminées mais non identiques*) et celui de *Flow Shop (toutes les tâches ont le même chemin (route ou gamme) à suivre).*

- *Les gammes mixtes ou semi-linéaires* où l'ordre d'exécution des opérations est partiellement déterminé, on trouve des tâches qui ont des gammes prédéterminées (*on sait à priori la route à suivre par ces tâches*), et d'autres non (*on ne sait pas, à priori, la route à suivre par ces tâches*).

5.4. Les Ressources

Une ressource *k* est un moyen technique et/ou humain, requis pour une réalisation d'une tâche et disponible en quantité limitée, sa capacité a_k (*supposée constante*) [Lope 01], [Esqu & Lope 99]. Plusieurs types de ressources sont à distinguer :

- *Ressources renouvelables* qui après avoir être utilisées par une ou plusieurs tâches, sont à nouveau disponibles en même quantités (*les hommes, les machines, l'espace, l'équipement en général, …etc*) ; les quantités des ressources utilisables à chaque instant sont limitées.

- *Ressource consommable* qui devient non disponible après avoir été utilisée par une ou plusieurs tâches, (*matière première, budget,…etc*) ; la consommation globale (*ou cumul*) au cours du temps est limitée.

- *Ressource doublement contraintes* lorsque son utilisation instantanée et sa consommation globale sont toutes les deux limitées (*source d'énergie, financement,… etc.*).

- *Ressources disjonctives (ou non partageables)*, principalement dans le cas de ressources renouvelables, sont des ressources qui ne peuvent exécuter qu'une tâche à la fois (machine-outil, robot manipulateur, … etc) [Esqu & Lope 99].

- *Ressources cumulatives (ou partageables)* qui peuvent être utilisées par plusieurs tâches simultanément (*équipe d'ouvriers, poste de travail*) [Esqu & Lope 99].

La nature de ressources prises en considération permet de dresser une typologie des problèmes d'ordonnancement (Figure.1.2).

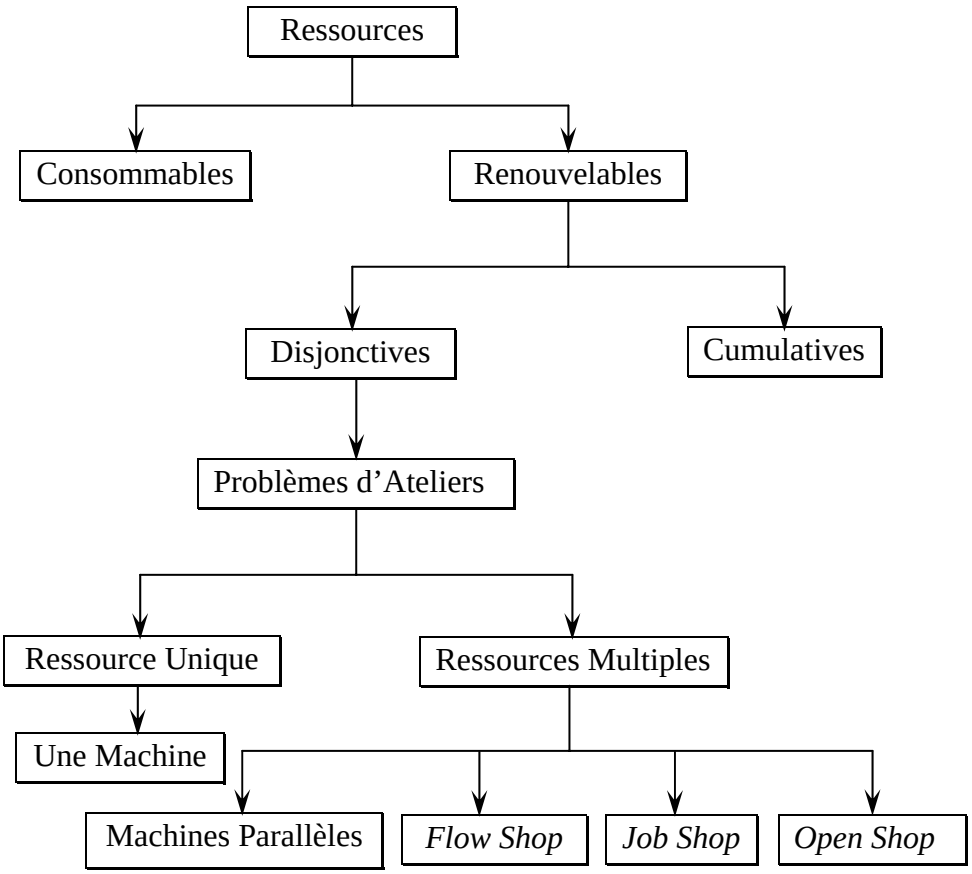

Figure.1.2. Typologie par les ressources des problèmes d'Ordonnancement

Pour les problèmes d'ordonnancement, on confondra souvent les ressources avec les machines. Généralement, les machines sont groupées par ateliers. Les exemplaires de machines d'un atelier diffèrent souvent uniquement par la rapidité d'exécution des opérations qu'elles réalisent.

On distingue alors plusieurs types d'exemplaires de machines [Hent 95] :

- *Les exemplaires de machines identiques :* une même opération passant sur l'une ou l'autre des machines aura la même durée opératoire. Une tâche peut indifféremment s'exécuter sur l'une ou l'autre des machines (*exemple d'une batterie de fraiseuses*).

On dit que les machines sont identiques, si la durée de traitement est la même sur toutes les machines [Lope 01]. Notre problème traite le cas de ces machines.

- *Les exemplaires de machines uniformes* : chaque machine a une vitesse de travail qui peut être différente des autres. La durée d'exécution d'une tâche est alors inversement proportionnelle à la vitesse v_k de la machine k ($p_k = p/v_k$) (exemple de deux imprimantes ayant des vitesses

d'impression de quatre pages par minute et de six pages par minute) [Hent 99].

On note que les machines sont dites uniformes, si la durée de traitement varie uniformément en fonction de la performance de ces machines [Lope 01].

- *Les exemplaires de machines différentes* : une opération peut s'exécuter sur plusieurs machines, mais le temps opératoire diffère d'une machine à l'autre. On note alors p_{ijk} le temps d'exécution de la tâche '*i*' dans l'atelier '*j*' sur la machine '*k*', exemple de machines avec technologies différentes (*tour à commande numérique et tour manuel*). Certains auteurs parlent également de machines indépendantes [Hent 99].

On dit que les machines sont indépendantes (*différentes*), lorsque la durée de traitement est totalement variable entre les différentes machines [Lope 01].

6. Les Organisations d'ateliers

Vu l'étendue des systèmes de production et la multitude de produits de consommation et de service, on dénombre une grande diversification des organisations de production que l'on rencontre dans la littérature.

6.1. Les Différents types d'organisations

On peut regrouper les organisations en trois grandes classes détaillées ci-après [Hent 99].

6.1.1. Les Organisations à ressource unique

Les organisations à ressource unique sont un cas que l'on peut rencontrer rarement dans les entreprises industrielles, par contre, ces organisations sont plus fréquentes dans les systèmes informatiques partagés où on dénote souvent la présence d'une seule imprimante, d'un seul microprocesseur et plusieurs usagers (*programmeurs compilant leurs codes, des impressions de document, …*).

Parfois, la présence d'une machine goulot dans une chaîne de fabrication peut pousser certains chercheurs à aborder le problème comme une seule machine à étudier afin d'utiliser des méthodes heuristiques appropriées aux systèmes à une seule machine.

6.1.2. Les Organisations à ressources multiples

Dans ce type d'organisations on distingue plusieurs cas selon l'ordre d'exécution des opérations des tâches :

- Le cheminement d'exécution des tâches se fait suivant un cheminement propre à la tâche (*chaque tâche passe sur toutes les machines dans un ordre fixé* [Esqu & Lope 99]), cette organisation est dite de type *Job Shop* (Figure.1.3). Si pour au moins un des postes de travail, on dispose de plus

d'un exemplaire pour effectuer l'opération, on dit que c'est une organisation de type *Job Shop Hybride*, autrement dit, on dispose d'au moins un atelier qui dispose plus d'une machine (Figure.1.4) sinon l'organisation est dite *Job Shop simple.*

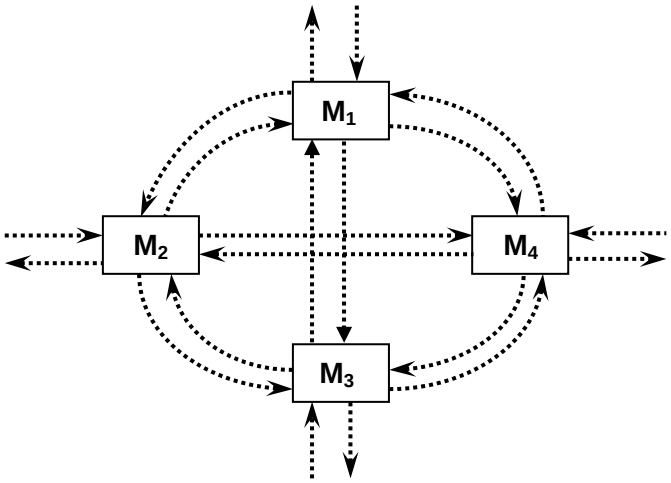

Figure.1.3. Exemple d'organisation *'Job Shop Simple'* avec 4 Machines

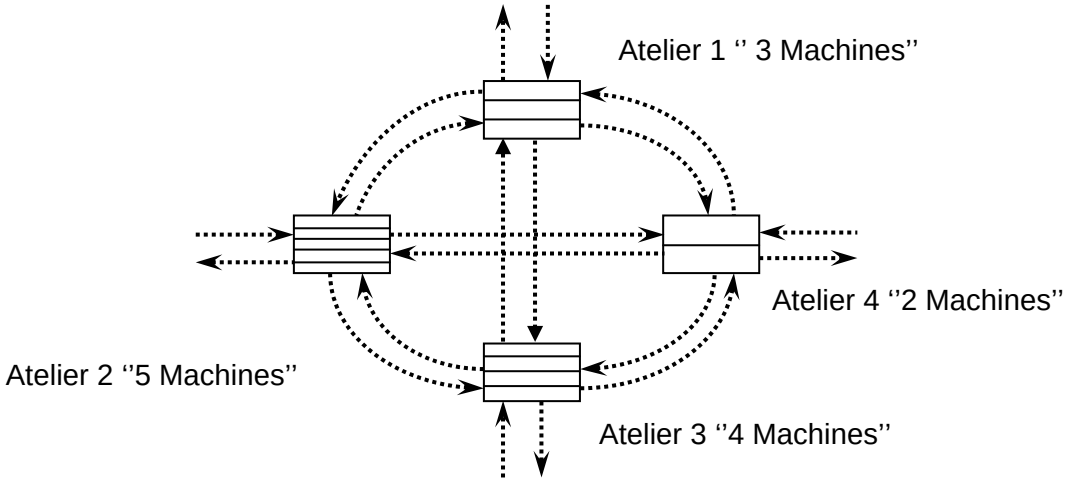

Figure.1.4. Exemple d'organisation *'Job Shop Hybride'* avec 4 Ateliers

- Dans le cas des organisations de type *Flow Shop Simple* [Hent & Guin 96], les tâches ont une gamme identique (*un même cheminement*) ; une tâche est ainsi constituée d'opérations visitant différentes machines et enchaînées de manière linéaire suivant une chaîne [Esqu & Lope 99], par conséquent, l'ordre de passage sur les postes de travail est le même (Figure.1.5). Autrement dit, on a *'m'* machines en série et chaque tâche doit être exécutée sur chacune des machines. Toutes les tâches doivent s'exécuter sur la machine 1, puis sur la machine 2,…, et ainsi de suite. Après la fin d'exécution sur une machine, la tâche joint la file d'attente de la machine suivante, et ne passe qu'une seule fois sur la machine [Pine 95]. Si pour au moins un des

postes de travail, on dispose de plus d'un exemplaire pour effectuer l'opération, on dit que c'est une organisation de type *Flow Shop Hybride* (*autrement dit on a 'w' ateliers en série et au moins l'un de ces ateliers dispose de plus d'une machine*) (Figure.1.6) sinon l'organisation est dite *Flow Shop Simple.*

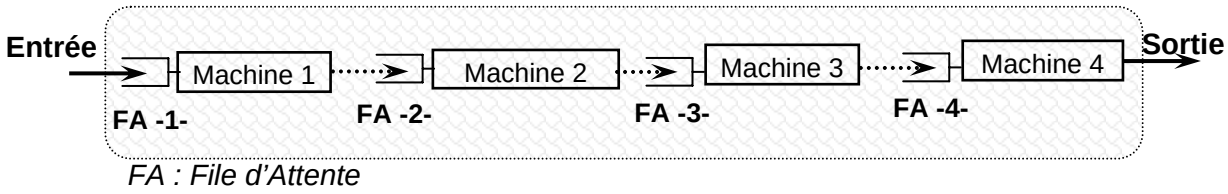

Figure.1.5. Exemple d'organisation *Flow Shop Simple* avec 4 Machines

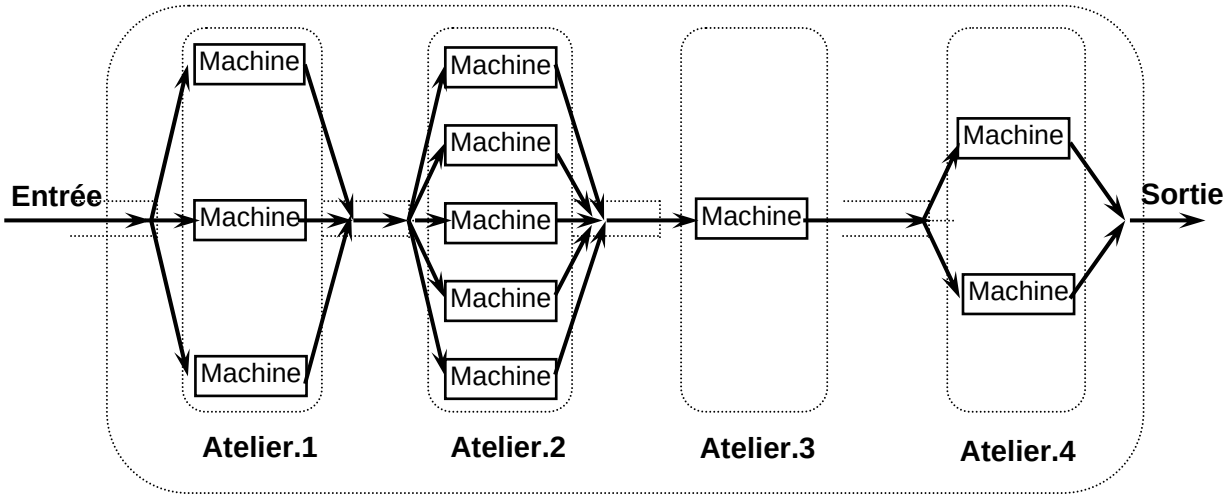

Figure.1.6. Exemple d'organisation *Flow Shop Hybride* avec 4 Ateliers

- Dans les organisations de type *Open Shop Simple* ou *Hybride***,** les tâches ont une gamme libre (*non précisée*). Dans les problèmes d'ordonnancement de production dit à cheminement libres (*ou sans contraintes d'enchaînement*), chaque produit à fabriquer doit subir diverses opérations sur des machines, mais dans un ordre totalement libre.

Les organisations à ressources multiples concernent les organisations de type machines parallèles (*le problème à machines parallèles se caractérise par le fait que plusieurs machines sont possibles pour l'exécution d'un travail qui n'en nécessite qu'une seule* [Esqu & Lope 99]) (Figure.1.7), *Flow Shop simple, Flow Shop hybride, job Shop, job Shop hybride*, et les organisations *Open Shop simple* et *hybride*. Très peu de résultats ont été obtenus au niveau de la résolution optimale de ces problèmes [Lope 01].

Les études théoriques ont surtout porté sur la recherche de résultats approchés avec une certaine garantie de performance en temps et en qualité.

La plupart des tentatives de résolution et de recherche d'algorithmes approchés sont basées sur l'algorithme polynomial de Johnson [Hent 99] qui résout, de manière optimale, le problème *Flow Shop* avec deux machines. En effet, la plupart des problèmes sont NP-Complets ou NP-Difficiles, (Les spécialistes qualifient de NP-Complets ou NP-Difficiles les problèmes pour lesquels on n'a pas trouvé d'algorithmes de résolution en un temps limité supérieurement par une fonction de taille N du problème, on qualifie de faciles les problèmes pour lesquels une telle fonction polynomiale existe (par exemple N^2) et de difficiles pour lesquels le temps de résolution n'est limité supérieurement que par une fonction exponentielle de N (par exemple 2^N)), [Hent 99], [Giar 03] .

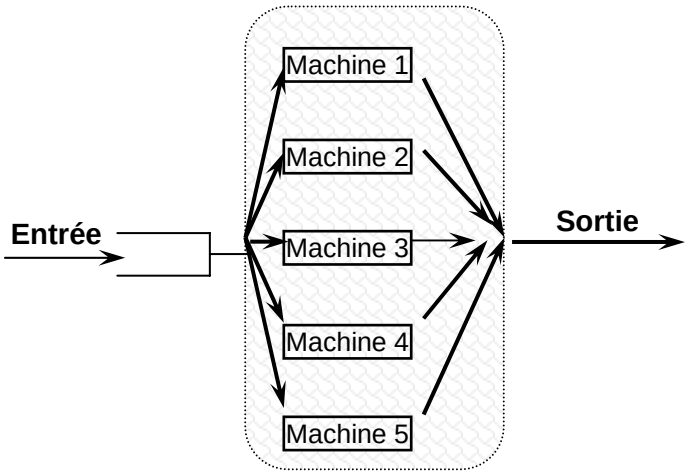

Figure.1.7. Exemple d'organisation machines parallèles avec 5 machines

6.1.3. Les organisations multi-lignes parallèles

Les systèmes à organisations Multi-lignes sont des systèmes hybrides avec des ressources dédiées.

A chaque ligne de production est assignée une certaine gamme de produits [Arti 97]. Un exemple d'organisation Multi-lignes parallèles est représenté sur la Figure.1.8

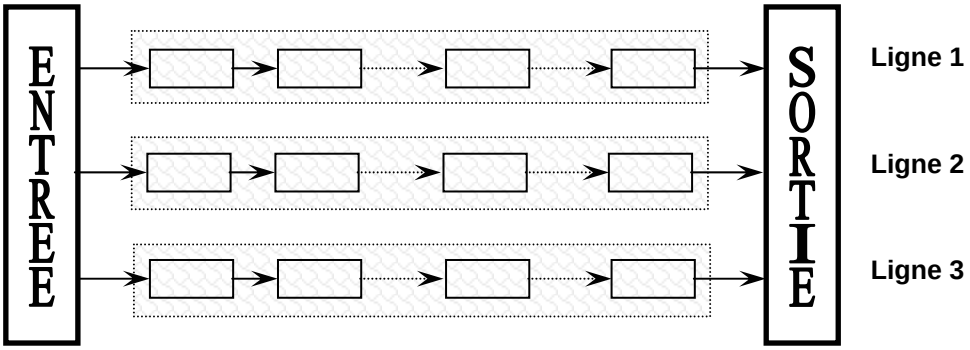

Figure.1.8. Schéma d'une organisation multi-lignes parallèles

Dans certains cas, il arrive que les lignes soient des organisations hybrides. Ces problèmes ont deux aspects, une organisation linéaire (*Flow Shop Simple, le cas où, on a une seule ligne de production*) et l'autre de machines en parallèle (*le cas où on a plus d'une ligne et chaque ligne dispose d'une seule machine*). La plupart de ces problèmes sont généralement résolus en associant des heuristiques et l'expérience humaine [Arti 97].

6.2. Relations entre organisations

La figure ci-dessous (Figure.1.9) illustre schématiquement les différences et les ressemblances qui existent entre les différentes organisations de systèmes de production. Un arc d'une organisation O_1 à une autre organisation O_2 peut être interprété comme suit : O_2 étant un cas particulier de la première organisation O_1. Ces relations entre organisations illustrent bien les similitudes des systèmes de production.

Dans certains cas, ce schéma peut être lu en terme de domaine de solutions réalisables, par exemple, l'ensemble des solutions réalisables du problème machine simple appartient aussi à l'ensemble des solutions réalisables du problème *Flow Shop Simple*. D'une façon plus générale, la figure ci dessous illustre les liens entre les différents systèmes de production [Hent 99].

Les relations, ainsi montrées, peuvent nous faire considérer la résolution du problème d'ordonnancement de l'organisation O_1 à travers une organisation O_2.

Nous pouvons lire sur la Figure.1.9, que le *Flow Shop Simple* est un cas particulier du *Flow Shop Hybride,* (dans le cas du *Flow Shop Hybride* où tous les ateliers disposent d'une seule machine on dit, dans ce cas, qu'on a une organisation de type *Flow Shop Simple*) et que le *Job Shop Simple* est un cas particulier du *Job Shop Hybride,* (dans le cas du *Job Shop Hybride* où tous les ateliers disposent d'une seule machine on dit, dans ce cas, qu'on a une organisation de type *Job Shop Simple),* on peut dire, aussi, que l'organisation de machines parallèles est un cas particulier du *Flow Shop Hybride,* c'est le cas où l'organisation de type *Flow Shop Hybride* dispose d'un seul atelier et ce dernier à son tour dispose d'au plus d'une machine, …etc.

On pourrait rajouter sur la Figure.1.9 ci-dessus le *Flow Shop* de permutation mais on le prendra en compte dans la définition du problème par la contrainte de non dépassement de tâches. Le *Flow Shop* de permutation est caractérisé, par rapport au *Flow Shop Simple,* par le même ordre de passage des tâches sur chaque machine.

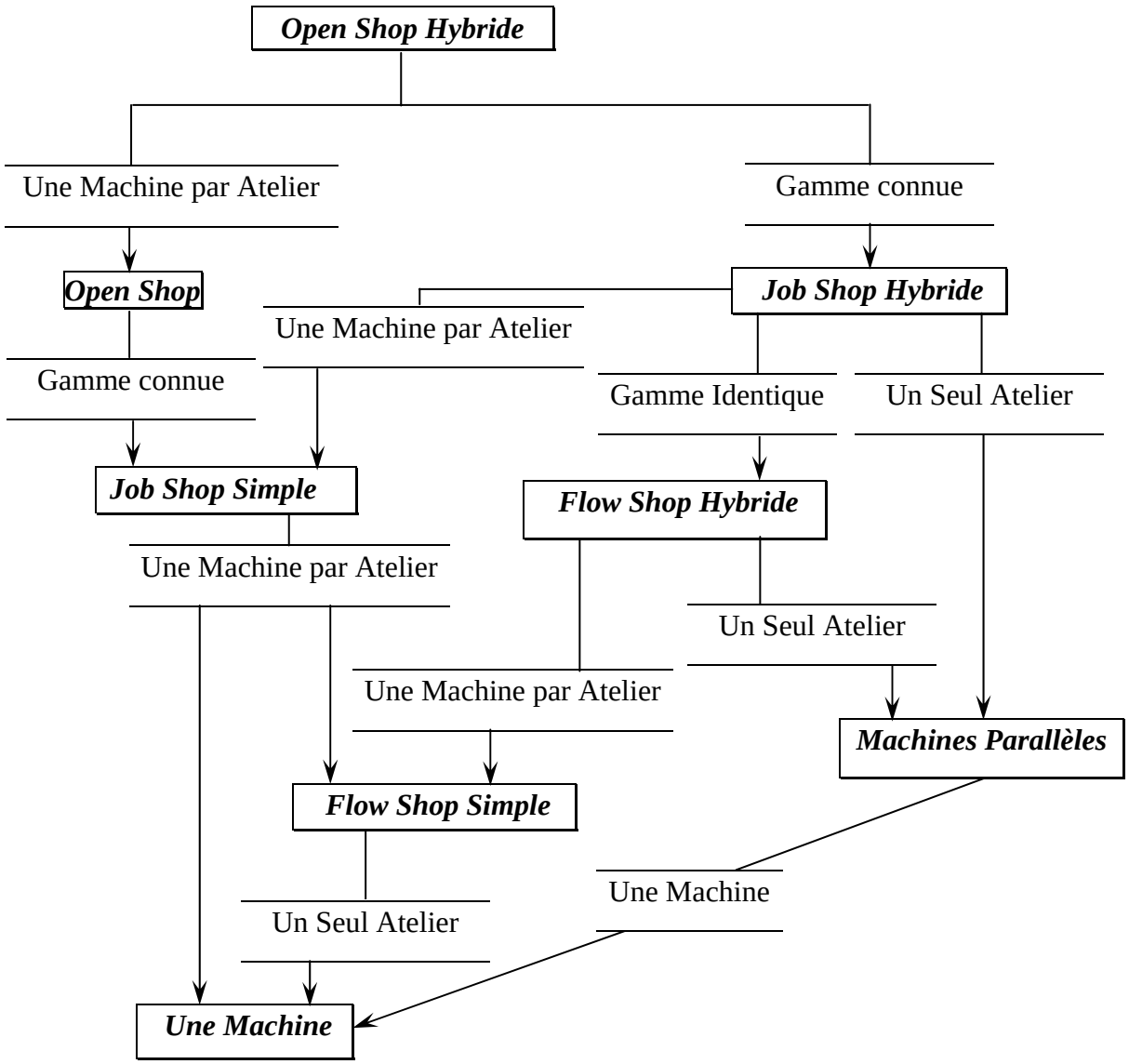

Figure.1.9. Schéma montrant les liens entre diverses organisations d'ateliers

7. Les Critères de performances

Selon le type de fabrication suivi dans chaque système de production, les critères de performance (*fonction objectif, critères d'optimisation*) se regroupent dans les deux principales approches de gestion de la production [Rinn 76] :

- Gestion sur stock,
- Gestion à la commande.

La gestion sur stock concerne les produits peu coûteux et d'usage général et standard. Ces produits sont fabriqués en grande quantité, souvent par lot de fabrication de taille importante. Ainsi le produit est disponible instantanément à la

vente et son prix est déjà catalogué. La gestion sur stock est inévitable lorsque le processus de fabrication est plus long que les délais de livraison [Hent 99].

La gestion à la commande s'applique à des produits plus spécifiques. Ils sont produits en petites séries. Généralement, ce mode de gestion est adopté pour éviter des stocks importants, par conséquent des en-cours considérables surtout pour des produits onéreux.

Ces deux stratégies de fabrication sont les plus utilisées dans les entreprises de production. On peut, néanmoins, trouver une troisième stratégie qu'on appellera stratégie mixte et qui utilise les deux politiques. Elle est employée notamment lorsque le délai de livraison est inférieur au délai de fabrication.

7.1. Classification des critères de performance

On peut regrouper les critères de performance suivant les quatre points ci-dessous [Hent 99] :

- Les dates de fin, notées C_i,
- Les délais de livraison, notés d_i,
- Le volume des en-cours ($\overline{C}$),
- L'utilisation des ressources (C_{max}).

Soit C_i la date de fin de la tâche '*i*' dans le dernier atelier, d_i le délais de livraison de la tâche '*i*' et r_i le délais de disponibilité de la tâche '*i*'. p_{ij} est la durée de l'opération '*j*' de la tâche '*i*'.

7.1.1. Les Critères basés sur les Dates d'Achèvement

Il s'agit des critères suivants :

- $C_{max} = Max\ C_i$: est le plus grand délais d'achèvement, ou bien la durée totale, le temps nécessaire pour finir toutes les tâches en terme anglo-saxon on l'appel 'Makespan'.
- $F_i = C_i - r_i$: est le délai de séjour.
- $F_{max} = Max\ r_i$: est le maximum des temps de séjour, r_i étant le délai de disponibilité de la tâche *i*.

Le temps d'attente est égal à : $W_i = C_i - r_i - \Sigma p_{ij}$

- $W_{max} = Max\ \{W_i / \forall i \in [1,n]\}$: est le maximum des temps d'attente.
- ΣC_i : est la somme des délais de fin.

- $\Sigma w_i C_i$: est la somme des délais de fin pondérés ; qui est le critère utilisé dans notre étude.

Les critères ci-dessus sont basés également sur les dates de fin des tâches : $\overline{C}$, $\overline{F}$, $\overline{W}$ sont les moyennes arithmétiques, respectivement, des dates de fin, des temps de séjours, et des temps d'attentes.

Dans certains cas, on trouve ces critères associés à une pondération des tâches. Dans ce cas, la pondération est notée *w*.

7.1.2. Les Critères basés sur les dates de livraison

Il s'agit des critères :

- $L_i = C_i - d_i$ est le retard algébrique,
- $L_{max} = Max\ L_i$: est le plus grand retard.
- $T_i = Max\ (0, L_i)$ est le retard absolu.
- $E_i = - Min\ (0, L_i)$ est l'avance absolue.

On trouve aussi des critères sous forme de moyennes arithmétiques, notés $\overline{L}$, $\overline{T}$, $\overline{E}$, ou bien encore pondérés par une pénalité. Représentant le nombre de tâches en retard, le critère N_t est également l'un des plus utilisés.

7.1.3. Les Critères basés sur les volumes des en-cours

Le critère $\overline{C}$ est des plus utilisés pour la minimisation du volume des en-cours (*l'en-cours : est le plus souvent caractérisé par le temps moyen des travaux* F_{moy} [Giar 88]). C'est la moyenne arithmétique de l'ensemble des dates de fin.

Une opération est toujours caractérisée par le numéro de la tâche et l'atelier sur lequel elle passe. Soit s_o la date de début de l'opération *o* et p_o sa durée d'exécution. A chaque instant '*t*' on peut calculer les mesures suivantes :

$N_p(t) = \sum_{i=1}^{n} e_i(t)$ est le nombre de tâches en cours d'exécution.

Avec $e_i(t) = \begin{cases} 1 & si(s_o \le t < s_o + p_o) \\ 0 & Sinon \end{cases}$

La maximisation de ce critère permettra d'exploiter de façon quasi optimale les ressources. Plus la valeur de ce critère est grande, plus le nombre de machines en attente est faible.

$N_w(t) = \sum_{i=1}^{n} e_i(t)$ est le nombre de tâches en attente d'exécution.

Avec $e_i(t) = \begin{cases} 1 & si(s_o + p_o \le t < s_{o+1}) \\ 0 & Sinon \end{cases}$

Minimiser ce critère permet de diminuer le nombre de tâches en attente dans les stocks intermédiaires entre les machines et, par conséquent de réduire la capacité intrinsèque des stocks.

$N_f(t) = \sum_{i=1}^{n} e_i(t)$ est le nombre de tâches terminées sur la dernière machine.

Avec $e_i(t) = \begin{cases} 1 & si(C_i \le t) \\ 0 & Sinon \end{cases}$

La maximisation de ce critère permet de répondre au plus vite à la demande des clients.

On peut observer que $N_p(t) + N_w(t) + N_f(t) = N$ (N : est le nombre de tâches à ordonnancer pour un horizon fixé) et, par conséquent, les objectifs de production dans ce cas peuvent être complètement antagonistes [Hent 99], [Rinn 76].

7.1.4. Les Critères basés sur l'utilisation des ressources

L'exploitation des ressources peut également être un souci majeur du gestionnaire. En plus, le critère de la minimisation de la plus grande date de fin C_{max} (*Makespan*), les critères suivants sont parmi les mieux appropriés. Soit M_v le nombre de machines disponibles dans l'atelier 'v' et C_{max} la plus grande date de fin de l'ordonnancement.

$\overline{U} = \left(\sum_{i=1}^{n} \sum_{j=1}^{v} p_{i,j} \right) \Big/ \left(\sum_{j=1}^{v} M_j * C_{\max} \right)$ Représente l'utilisation moyenne des ressources.

Elle permet de signaler éventuellement la sous-utilisation des ressources de fabrication de l'entreprise. Elle indique par conséquent les périodes pleines et les périodes creuses de la chaîne de fabrication.

$I_v = (C_{max} - \sum_{i=1}^{n} p_{i,v}) * M_v$ est le temps d'inactivité de l'atelier v (*Idle time*).

Cet indicateur donne le temps d'inactivité de l'atelier correspondant et sa minimisation permettra une plus grande exploitation des ressources.

7.2. Propriétés de ces critères

Un critère est dit *régulier* quand sa valeur décroît, si et seulement si l'un au moins des C_i décroît (les autres restants fixes) [Oura 01].

Un critère en fonction des variables C_i est dit *régulier* si l'on peut le dégrader en avançant l'exécution d'une tâche, c'est le cas pour la minimisation du temps total d'exécution, du plus grand retard, …etc [Lope 01].

Un sous ensemble de solutions est dit *dominant* pour l'optimisation d'un critère donné, s'il contient au moins un optimum pour ce critère [Esqu & Lope 99].

7.3. Relations entre ces Critères

Il existe plusieurs relations d'équivalence et de similitude entre les critères d'optimisation. En effet, une solution optimale pour l'un ou l'autre des problèmes pour un critère donné peut s'avérer optimale pour un autre critère.

La figure ci-dessous donne les liens entre certains critères, et on lit, au début qu'une solution optimale pour le critère L_{max} est aussi optimale pour le critère C_{max} (*avec* d_i =0), autrement dit, une méthode qui optimise L_{max} optimise C_{max}, la méthode qui optimise le critère $\Sigma w_i C_i$ optimise aussi le critère ΣC_i , …etc.

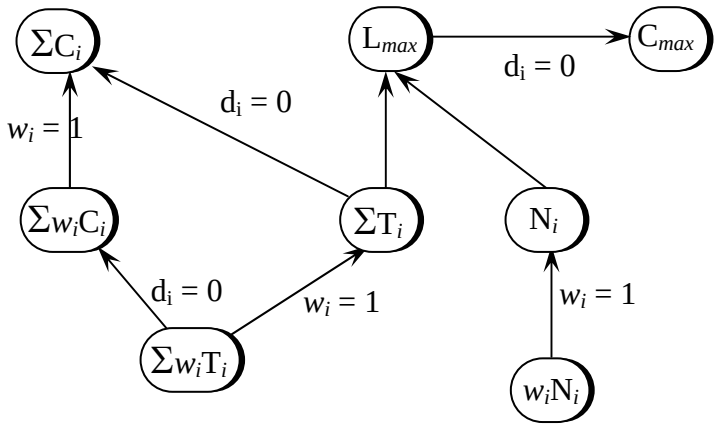

Figure.1.10. Les Relations entre les critères d'optimisation

8. Représentation des Résultats

Le graphique de Gantt, encore appelé diagramme de Gantt, est une technique de visualisation de l'utilisation de moyens productifs et/ou de l'avancement de l'exécution de tâche popularisée par Gantt en 1917, (mais dont retrouve des utilisations chez les prêtes égyptiens de l'antiquité) et est classiquement utilisé en ordonnancement en atelier [Giar 03], [Esqu & Lope 99]. Une tâche y est représentée sur un axe, habituellement horizontal, par un segment dont la longueur est, en principe, proportionnelle au temps d'exécution. Lorsque l'on étudie l'évolution de l'utilisation de plusieurs facteurs productifs (par exemple des machines), l'utilisation de chaque facteur productif est portée sur un axe différent ; on a alors un faisceau de droites parallèles sur un même document (tableau mural par exemple).

La même échelle temporelle est utilisée pour tous les axes, ce qui fait que les intersections de ces droites avec une même perpendiculaire repèrent le même instant. Pour faciliter ce repérage, un papier quadrillé est utilisé et l'échelle des temps est explicitée en haut du document. Un curseur vertical permet de visualiser un instant précis du temps, ce qui facilite l'utilisation de ce document pour le suivi d'un atelier. Au dessus de chaque segment on porte le code d'identification de la tâche (ou ordre de fabrication), ou des quantités s'il s'agit de la production d'un article connu sans ambiguïté.

Ce document est utilisé pour le lancement des travaux en atelier. Conventionnellement une tâche est programmée comme le montre la Figure.1.11 où la longueur du segment est proportionnelle à la durée d'exécution et *x* est le numéro de la tâche, ou la quantité à produire. Notons que le diagramme de Gantt est le meilleur moyen pour visualiser la solution d'ordonnancement et calculer le temps de chaque tâche ainsi que les différents temps d'achèvement de ces tâches 'Makespan C_{max}'.

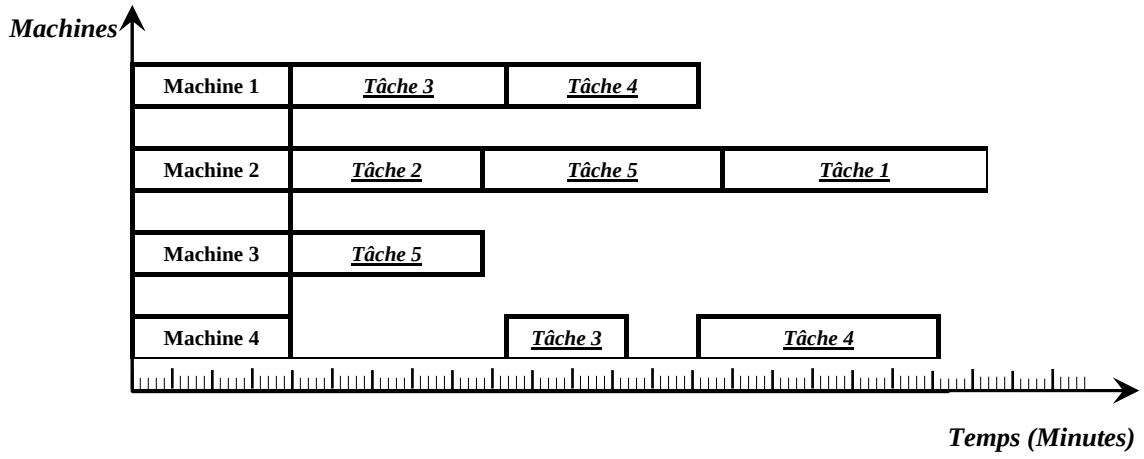

Figure.1.11. Exemple de visualisation d'un Ordonnancement sur un Diagramme de Gantt

9. Conclusion

Les problèmes d'ordonnancement se retrouve dans toutes les formes d'organisation de la production. Dans les systèmes de production ils sont considérés comme un problème *NP-Difficile.* La qualité des solutions proposées conditionne largement la performance, voire la survie, de l'organisation au sein de ces systèmes de production. La complexité de ces problèmes n'est pas en terme de formalisation réaliste car les techniques de modélisation permettent une formalisation satisfaisante de ces problèmes, mais en terme de résolution en raison du nombre élevé de variables de commande, de la nature discrète de ces variables et de la prise en compte des contraintes.

La suite de notre travail sera consacrée à présenter les différentes approches existantes pour la résolution de ces problèmes d'ordonnancement.

Chapitre II

Approches de Résolution

Dans ce chapitre, nous présenterons les différentes méthodes pour la recherche d'une bonne solution aux problèmes d'ordonnancement. Nous présenterons le principe de chaque méthode d'optimisation appliquée en ordonnancement. Nous essayerons de donner une graduation dans les méthodes en partant de la technique la moins performante à des techniques performantes sur le point de vue du temps de calcul et efficacité [Plip 98], [Giar 03].

1. Introduction

L'optimisation est le processus d'aller vers le meilleur. L'ingénieur ou le scientifique crée une nouvelle idée, l'optimisation lui améliore cette idée. Suivant la nature du problème à traiter, l'optimisation consiste à ajuster les entrées ou les caractéristiques de l'idée pour trouver le minimum ou le maximum des sorties ou des résultats. En ordonnancement l'optimisation consiste à résoudre mathématiquement un ordre de passage entre tâches sous contraintes d'une telle façon à minimiser ou maximiser une fonction (ou parfois plusieurs) objectif donnée.

L'ordonnancement est le plus répondu des modes d'organisation des systèmes de production. C'est sans doute aussi celui qui pose les problèmes les plus compliqués, non pas en termes de formalisation réaliste, car les techniques de modélisation permettent de le décrire d'une façon satisfaisante, mais en terme de résolution en raison de termes élevés de variables de commande, de la nature discrète de ces variables et de la prise en compte des contraintes de disponibilité de ressources, qui conduit à la résolution de problème de grande taille, et implique l'utilisation des méthodes approchées (heuristiques).

Les apports de la recherche opérationnelle sont loin d'être négligeables comme nous allons le voir en examinant les différentes heuristiques de résolution des problèmes d'ordonnancement. Ça sera l'occasion de présenter un formalisme permettant de distinguer les problèmes d'ordonnancement entre eux et de les classer, cela donne avantage à modéliser le problème d'ordonnancement, le résoudre ensuite avec une méthode appropriée.

2. Classification des Problèmes d'ordonnancement

Etant donné la diversité des problèmes d'ordonnancement, nous utilisons couramment un formalisme de classification [Lope 01],[Plip 98], permettant de distinguer les problèmes d'ordonnancement entre eux et de les classer. Ce formalisme, comporte trois champs : α, β, γ **,** permettant de décrire les différentes entités d'un problème d'ordonnancement.

2.1. Champ α : Organisation des ressources

Le champ α est composé de deux sous champs α_1, α_2. Les ressources peuvent être parallèles (chaque machine peut exécuter chaque tâche) ou dédiées (spécialisées à l'exécution d'une ou plusieurs tâches). Dans le cas de machines parallèles, elles peuvent être soit identiques et avoir la même vitesse pour toutes les tâches, soit uniformes : chaque machine a alors sa propre vitesse qui ne dépend pas de la tâche exécutée, soit non liées (cas plus général) où la vitesse d'exécution dépend de la machine et de la tâche exécutée.

Le paramètre $\alpha_1 \in \{\phi, P, Q, R, F, O, J\}$, représente le type de ressources principales (machines) utilisées :

- $\alpha_1 = \phi$: une seule machine. C'est le cas le plus simple qui peut être vu comme un cas particulier des autres configurations.
- $\alpha_1 = P$: machines parallèles identiques. Les ressources sont composées de machines de même cadence, disposées en parallèle pouvant exécuter toutes les tâches.
- $\alpha_1 = Q$: machines parallèles uniformes. Les cadences des machines sont différentes deux à deux, mais restent indépendantes des tâches.
- $\alpha_1 = R$: machines parallèles non liées. Les cadences des machines sont différentes deux à deux et dépendent des tâches exécutées.
- $\alpha_1 = F$: fow-shop. Les tâches sont décomposées en plusieurs opérations qui doivent être exécutées sur l'ensemble des machines, disposées en série, selon un même routage.
- $\alpha_1 = O$: open-shop. Les opérations des tâches doivent être exécutées sur certaines machines sans restriction sur le routage des tâches.
- $\alpha_1 = J$: job-shop. Les opérations des tâches doivent être exécutées sur l'ensemble des machines disposées en série, mais peuvent avoir des routages différents.

Le paramètre $\alpha_2 \in \{\phi, m\}$, dénote le nombre de machines composant le système.

Lorsque $\alpha_1 = \phi$, le nombre de machines est variable, tandis que lorsque $\alpha_1 = m$ le nombre de machines est égal à *m* (*m* > 0).

2.2. Champ β : contraintes et caractéristiques du système

Ce champ $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8$ décrit les caractéristiques des ressources et des taches. Pour répondre à des problèmes industriels des plus divers, la variété de ces caractéristiques est très importante.

Tout d'abord, différents modes d'exécution sont possibles : soit avec préemption (l'exécution d'une opération peut être interrompue puis reprise sur la même machine ou sur une autre), soit sans préemption (lorsqu'une opération est commencée, elle doit être terminée avant de pouvoir exécuter une autre opération sur la même machine).

- Le paramètre $\beta_1 \in \{\phi, pmtn\}$, indique cette possibilité de préemption $\beta_1 = (pmtn)$. Si $\beta_1 = \phi$, la préemption n'est pas autorisée.

- Le paramètre $\beta_2 \in \{\phi, res\}$, caractérise les ressources supplémentaires nécessaires à l'exécution d'une tâche (outils, ressources de transport). $\beta_2 = \phi$, indique qu'il n'y a pas de ressource complémentaire. $\beta_2 = res$, spécifie les ressources complémentaires.

- Le paramètre $\beta_3 \in \{\phi, prec, tree, chain\}$, reflète les contraintes de précédence entre tâches spécifiant qu'une tâche doit être exécutée avant une autre. Les valeurs; *prec*, *tree*, *chain*, représentent respectivement des tâches indépendantes, des relations de précédence générales, des relations de précédence particulières sous forme d'arbres ou de chaînes.

- Le paramètre $\beta_4 \in \{\phi, r_j\}$, décrit les dates d'arrivée (ou dates de disponibilité, ou dates au plus tôt) des différentes tâches dans le système. Ces dates peuvent être identiques et égales à zéro pour toutes les tâches $(\beta_4 = \phi)$ ou différentes suivant les tâches $(\beta_4 = r_j)$

- Le paramètre $\beta_5 \in \left\{\phi, p_j = p, \underline{p} \le p \le \bar{p}\right\}$, détaille les temps d'exécution des différentes tâches. Ces temps peuvent être ou ne pas être fonction de la machine. Différentes restrictions peuvent également être considérées pour simplifier certains problèmes. Si :

  - $\beta_5 = \phi$: Les tâches ont des temps d'exécution arbitraires.

  - $\beta_5 = p_j = p$: Toutes les taches ont un temps d'exécution égal à *p*.

- $\beta_5 = \underline{p} \le p \le \bar{p}$: Les temps d'exécution des taches sont compris entre $\underline{p}$ *et* $\bar{p}$.

- Le paramètre $\beta_6 \in \left\{\phi, d_j\ \bar{d}_j\right\}$, indique les éventuelles dates d'échéance (ou dates au plus tard) des tâches, dates pour lesquelles les tâches doivent être terminées.Si :

- $\beta_6 = \phi$: Les taches n'ont pas de date d'échéance.

- $\beta_6 = \bar{d}_j$: Chaque tache a une date d'échéance souhaitée. Généralement, un non respect de ces dates entraîne une pénalisation.

- $\beta_6 = \bar{d}_j$: chaque tâche a une date d'échéance impérative (date limite), qu'il faut absolument respecter.

- Le paramètre $\beta_7 \in \{\phi, s, s_i, s_{ij}, nwt\}$, permet de spécifier des contraintes sur les enchaînements de tâches très souvent introduites pour représenter au mieux les problèmes réels. Il est parfois nécessaire de considérer un temps de changement entre l'exécution de deux tâches différentes sur une même machine pour représenter les changements et réglages d'outils. Ces temps de changement peuvent être constants, fonction de la nouvelle tâche (s_i) ou bien fonction de l'enchaînement des deux tâches (s_{ij}). Egalement, pour le cas des industries ou l'on manipule de la matière en fusion, il convient de pouvoir imposer que toutes les opérations d'une tâche soient exécutées sans temps d'attente $(\beta_7 = nwt(no\ wait))$.

  - Le paramètre $\beta_8 \in \{\phi, M_j\}$ indique dans le cas de machines parallèles, des restrictions sur la polyvalence des machines. L'ensemble M_j représente l'ensemble des machines capables de réaliser la tache *j*. Lorsque β_8 est vide, toutes les machines sont capables d'exécuter toutes les tâches.

2.3. Champ γ : critères d'optimisation

Le troisième champ, le champ γ, concerne les critères d'évaluation d'un ordonnancement. Les objectifs visés sont liés à une bonne utilisation des ressources, une minimisation du délai global ou encore le respect d'un maximum de contraintes. Nous donnons ici les critères les plus couramment utilisés :

$\gamma = C_{Max}$: *Makespan*. Ce critère vise à minimiser la date de sortie du système de la dernière tâche ($\gamma = C_{Max} = Max_j C_j$, C_j représente la date fin d'exécution de la tâche *j*)

$\gamma = w_j C_j$: Somme pondérée des dates de fin d'exécution. Ce critère représente la somme des encours des tâches dans le système.

$\gamma = L_{Max}$: Décalage temporel maximal. Ce critère mesure la plus grande violation des dates d'échéances souhaitées $\left(L_{Max} = Max_j L_j = Max_j \left(C_j - d_j\right)\right)$.

$\gamma = \sum w_j T_j$: Somme pondérée des retards ($\left(T_j = Max_j \left(C_j - d_j, 0\right)\right)$, qui permet d'évaluer les pénalités dues aux retards (w_j : poids de la tache *j*).

$\gamma = \sum w_j U_j$: Somme pondérée du nombre de tâches en retard (U_j =1 si la tâche *j* est en retard, 0 sinon).

3. Complexité

L'expérience montre que certains problèmes sont plus faciles que d'autres à résoudre. Une théorie de la complexité a été développée et permet mathématiquement de classer les problèmes faciles et difficiles en deux classes : les classes P et NP [Lope 01]. Nous exposerons dans la partie qui suit, les grands principes de la théorie de la complexité des problèmes d'ordonnancement.

4. 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 [Plip 98]. 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) [Giar 03], [Plip 98]. 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 [Plip 98]. 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 [Giar 03]. Les problèmes d'optimisation combinatoire dont le problème de décision associé est *NP*-complet sont qualifiés de *NP*-difficiles.

5. Méthodes de résolution

Nous avons vu précédemment qu'il existe des problèmes d'ordonnancement de complexité différente. Les problèmes appartenant à la classe *P* ont des algorithmes polynomiaux permettant de les résoudre. Ces algorithmes sont spécifiques au problème à résoudre et nous ne les citerons pas ici. Pour les problèmes appartenant à la classe *NP*, l'existence d'algorithmes polynomiaux semble peu réaliste. Ainsi, différentes méthodes de résolution (méthodes exactes ou heuristiques), sont largement utilisées pour appréhender les problèmes *NP*-difficiles. Nous exposons dans cette partie, les grandes lignes des méthodes les plus connues classées en trois catégories : les méthodes exactes, les heuristiques constructives et les méthodes amélioratices. Le lecteur intéressé par de plus amples détails pourra se reporter aux références

5.1. Les méthodes exactes

Dans ce paragraphe, il est question de trois types de méthodes exactes : la programmation dynamique, la méthode de séparation et évaluation et la résolution à partir d'une modélisation analytique. Comme nous le verrons, le temps de calcul de ces méthodes est exponentiel ce qui explique qu'elles ne sont utilisables que sur des problèmes de petites tailles.

5.1.1 Programmation dynamique

Introduite par Bellman dans les années 50 [Plip 98], la programmation dynamique décompose un problème de dimension n en n problèmes de dimension 1. Le système est alors constitué de n étapes que l'on résout séquentiellement, le

passage d'une étape à une autre se faisant à partir des lois d'évolution du système et d'une décision.

Le principe d'optimalité de Bellman est basé sur l'existence d'une équation récursive permettant de décrire la valeur optimale du critère à une étape en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille k -1 et porte le nombre de sous-ensembles considérés à 2^n (où n est le nombre d'éléments considérés dans le problème). La programmation dynamique est donc de complexité exponentielle. Pourtant, pour les problèmes *NP*-difficiles au sens faible, c'est-à-dire pour lesquels il existe des algorithmes de résolution pseudo-polynomiaux, il est souvent possible de construire un algorithme de programmation dynamique pseudo-polynomial, pouvant alors être utilisé pour des problèmes de tailles raisonnables.

5.1.2 Méthode de séparation et évaluation

La méthode de séparation et évaluation est basée sur une énumération implicite et intelligente de l'ensemble des solutions réalisables. Pour cela, la séparation consiste à décomposer le problème initial en plusieurs sous problèmes qui sont à leur tour décomposables. Ce processus peut se visualiser sous forme d'un arbre d'énumération. Pour chaque sous problème (noeud de l'arbre), la procédure d'évaluation calcule (dans le cas d'un problème de minimisation) une borne inférieure de la solution obtenue à partir de ce sous problème. Au préalable, une borne supérieure de la solution optimale a été calculée et est utilisée pour éviter l'exploration de noeuds dont la valeur de la borne inférieure est supérieure à la valeur de la borne supérieure. Cette borne supérieure est réactualisée lorsqu'une solution réalisable de valeur inférieure est atteinte.

Ainsi, l'exploration de certaines branches de l'arbre est coupée, ce qui permet de ne pas énumérer réellement toutes les solutions. Il faut donc remarquer que l'efficacité d'un algorithme de séparation et d'évaluation est déterminée par la qualité des bornes utilisées et par de bonnes conditions de branchement.

5.1.3 Modélisation analytique et résolution

La modélisation analytique d'un problème permet, non seulement de mettre en évidence l'objectif et les différentes contraintes du problème, mais également, parfois de le résoudre. L'idéal est d'obtenir un programme linéaire dont les variables sont réelles. Dans ce cas, il existe un bon nombre de solveurs pouvant le résoudre. Dès que le problème comporte des coûts fixes, ou des décisions nécessitent l'utilisation de variables entières, les modèles deviennent plus difficiles à résoudre. Il en est de même lorsque le modèle n'est pas linéaire, mais quadratique par exemple.

Néanmoins, il est parfois surprenant de voir qu'un problème particulier de taille intéressante peut être appréhende et résolu par la programmation mathématique. Il est donc justifié de commencer à étudier un problème en proposant une ou plusieurs modélisations analytiques. De plus, cette démarche a été simplifiée car il existe

aujourd'hui plusieurs langages de modélisation permettant d'écrire les programmes linéaires de façon formelle, proche de l'écriture mathématique. Ces langages de modélisation peuvent soit générer des fichiers lisibles par des solveurs, soit être directement couplés à ces solveurs. Malheureusement, tous les problèmes ne peuvent être résolus par cette approche car la résolution de programmes linéaires mixtes, par exemple, demande souvent beaucoup trop de temps de calcul.

5.2. Méthodes heuristiques constructives

Pour les problèmes de grande taille, les méthodes exactes ne sont pas envisageables de par leur temps de calcul. Il est dans ce cas possible d'utiliser des méthodes approximatives qui donnent des solutions certes sous-optimales, mais obtenues rapidement. Ces solutions peuvent ensuite servir de solution initiale pour les méthodes amélioratrices. La performance de tels algorithmes est généralement calculée par le rapport entre la valeur de la solution calculée par l'heuristique et la valeur de la solution optimale : $R_A(I) = A(I) \big/ OPT(I)$ pour le pire des cas. D'autres analyses de performances peuvent être menées. Par exemple, l'analyse en moyenne regarde le comportement moyen de l'heuristique en calculant $R_A(I)$, non pas seulement pour le pire des cas, mais également pour la moyenne de plusieurs instances. De plus, lorsque la solution optimale n'est pas calculable (parce que les problèmes sont de trop grande taille, par exemple), il est également possible d'étudier expérimentalement le comportement de l'heuristique en comparant ses performances soit avec les performances d'autres heuristiques, soit avec des bornes inférieures de la solution optimale.

Les méthodes par construction progressive sont des méthodes itératives où à chaque itération, une solution partielle est complétée. La plupart de ces méthodes sont des algorithmes gloutons car elle considèrent les éléments (processeurs ou tâches, suivant les cas) dans un certain ordre sans jamais remettre en question un choix, une fois qu'il à été effectué. De principe très simple, ces algorithmes permettent de trouver une solution très rapidement. Parmi ces méthodes nous en exposons deux types.

Regardons tout d'abord, les algorithmes de liste. Ces algorithmes consistent, dans une première phase, à calculer une liste qui donnera l'ordre de prise en compte des éléments. Cette liste construite à priori à partir d'un critère bien défini, n'est pas remise en cause au cours de l'ordonnancement. La deuxième phase de l'algorithme se réduit à considérer les éléments (processeurs ou tâches) dans l'ordre de la liste pour construire l'ordonnancement.

Un deuxième type de méthode par construction progressive, consiste à choisir, au cours de la construction, l'affectation d'une tâche sur une machine en utilisant des règles de priorité. Ces méthodes peuvent également être vues comme des méthodes sérielles dynamiques où les listes sont reconstruites à chaque étape, selon un critère qui peut évoluer dans le temps (le nombre de tâches disponibles, par exemple). Notons que si le choix de la tâche à ajouter est guidé par l'application d'un théorème

de dominance, qui permet d'établir des résultats concernant l'existence de solutions optimales si certaines hypothèses sont vérifiées [Goth 88], la méthode peut donner la solution optimale.

5.3. Méthodes amélioratrices

Nous exposons, dans ce paragraphe, les méthodes d'amélioration locales les plus connues. Ces méthodes sont initialisées par une solution réalisable, calculée soit aléatoirement soit à l'aide d'une des heuristiques constructives exposées précédemment, et recherchent à chaque itération une amélioration de la solution courante par des modifications locales. Cet examen se poursuit jusqu'à ce qu'un critère d'arrêt soit satisfait. L'utilisation de ces heuristiques itératives suppose que l'on puisse définir pour toute solution *S*, un voisinage de solution, *N(S)*, contenant les solutions voisines (proches dans un certain sens). En général, le voisinage d'une solution est généré en appliquant, plusieurs fois et de façon différente, à cette solution, une petite transformation (échanges, par exemple). A chaque solution S, nous associons le coût de cette solution, *c(S)*.

5.3.1 Méthode de descente

C'est l'une des heuristiques de recherche locale les plus simples. Elle consiste à rechercher dans le voisinage de la solution courante, une solution de coût plus faible. Elle procède ainsi jusqu'à arriver à un optimum local. Cette méthode a l'avantage d'être rapide, mais s'arrête dès qu'un optimum local est atteint, même si celui-ci n'est pas de bonne qualité.

Parmi ces méthodes, nous décrivons ici les méthodes d'échanges de type *r-opt*. Ces méthodes d'optimisation ont été initialement proposées pour résoudre le Problème du Voyageur de Commerce (PVC), mais elles s'appliquent également à tout problème combinatoire dont la solution consiste en une permutation de *r* composantes parmi *n*.

Le terme *r-opt* indique qu'une solution ne peut plus être améliorée en échangeant *r* éléments. La méthode consiste donc à sélectionner *r* composantes et à regarder si en interchangeant ces composantes, on obtient une meilleure solution.

Nous remarquons donc qu'une solution *n-optimale* est une solution optimale (dans le cas d'un problème de taille n). Ainsi, plus *r* augmente, plus on se rapproche de la solution optimale, mais plus les calculs sont difficiles. En effet, il y a (r^n) façons de choisir *r* composantes et r! façons de les échanger. En pratique, on se limite à *r = 2 ou 3.*

5.3.2 Recuit Simulé

Cette classe de méthodes d'optimisation est due aux physiciens Kirkpatrick, Gelatt et Vecchi [Lope 01]. Elle s'inspire des méthodes de simulation de Métropolies (année 50) en mécanique statistique.

L'analogie historique s'inspire du recuit des métaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global.

La méthode du recuit simulé, appliquée aux problèmes d'optimisation, considère une solution initiale et recherche dans son voisinage une autre solution pouvant devenir solution courante. L'originalité de cette méthode est qu'il est possible de se diriger vers une solution voisine de moins bonne qualité avec une probabilité non nulle. Ceci permet d'échapper aux optima locaux. Au début de l'algorithme, un paramètre *T*, apparenté à la température, est déterminé et décroît tout au long de l'algorithme pour tendre vers 0. De la valeur de ce paramètre va dépendre les probabilités d'acceptation des solutions détériorantes. La performance du recuit simule dépend de la règle de refroidissement (c'est-à-dire la décroissance du paramètre T) que l'on utilise. Un refroidissement trop rapide mènerait vers un optimum local pouvant être de très mauvaise qualité. Un refroidissement trop lent serait très coûteux en temps de calcul. Le réglage des différents paramètres (température initiale, nombre d'itérations par palier de température, décroissance de la température, ...) peut donc être long et difficile.

L'intérêt des cette classe de méthode est qu'il existe une preuve de la convergence, ainsi, lorsque certaines conditions sont vérifiées, on a la garantie d'obtenir la solution optimale.

5.3.3 Recherche Tabou

Ces méthodes dont l'origine peut remonter à 1977 [Plip 98], ont été formalisées plus tard, en 1986, par Glover [Lope 01]. Elles n'ont aucun caractère stochastique et utilise la notion de mémoire pour éviter de tomber dans un optimum local.

Le principe de l'algorithme est le suivant : à chaque itération le voisinage (complet ou un sous-ensemble du voisinage) de la solution courante est examiné et la solution minimisant l'augmentation de coût est sélectionnée. Pour éviter les phénomènes de cyclage, la méthode a l'interdiction de revisiter une solution récemment visitée. Pour cela, une liste taboue contenant les dernières solutions visitées est tenue à jour. Chaque nouvelle solution considérée chasse de cette liste la solution la plus anciennement visitée. La longueur de la liste, paramètre à définir, détermine donc le nombre d'itérations pendant lesquelles une solution ayant été visitée ne pourra être reconsidérée. Ainsi, la recherche de la solution courante suivante se fait dans le voisinage de la solution courante actuelle sans considérer les solutions appartenant à la liste taboue. Tout au long de l'algorithme, la meilleure solution doit être conservée car l'arrêt se fait rarement sur la meilleure solution. En effet, l'arrêt de cette méthode peut se faire soit après un certain nombre d'itérations, soit lorsqu'il n'y a pas eu d'amélioration de la meilleure solution depuis un certain nombre d'itérations. Des versions plus élaborées permettant des recherches plus efficaces, ont été proposées par la suite.

Cette méthode donne de très bons résultats pratiques, malgré l'inexistence de résultats théoriques garantissant la convergence de l'algorithme vers la solution optimale.

5.3.4 Algorithmes Génétiques

Cette classe de méthodes est basée sur une imitation des phénomènes d'adaptation des êtres vivants. L'application de ces méthodes aux problèmes d'optimisation a été formalisée par Goldberg en 1989 [Gold 89].

Les algorithmes génétiques fonctionnent sur une analogie avec la reproduction des êtres vivants. On part d'une population (ensemble de solutions) initiale sur laquelle des opérations de reproduction, de croisement ou de mutation vont être réalisées dans l'objectif d'exploiter au mieux les caractéristiques et propriétés de cette population. Ces opérations doivent mener à une amélioration (en terme de qualité des solutions) de l'ensemble de la population puisque les bonnes solutions sont encouragées à échanger par croisement leurs caractéristiques et à engendrer des solutions encore meilleures.

Les difficultés pour appliquer les algorithmes génétiques résident dans le besoin de coder les solutions, et dans les nombreux paramètres à fixer (taille de la population, coefficients de reproduction, probabilité de mutation, ...). Cette approche d’optimisation sera l’objet d’une étude détaillée au chapitre suivant.

6. Conclusion

Nous avons vu, dans ce chapitre, différentes classifications d'un problème d'ordonnancement, nous l'avons sélectionné comme un problème *NP-difficile*. Nous avons exposé plusieurs méthodes de résolution heuristique.

Dans ce type de résolution heuristique, l'inconvénient c'est qu'on ne peut pas garantir théoriquement la bonne qualité des résultats (minimisation des coûts, de temps) pour cas d'étude bien défini. Nous sommes donc invités à tester une méthode de résolution et dans le cas échéant faire une étude comparative des différentes méthodes, cela s'avère lent et pénible numériquement.

Les Algorithmes Génétiques ont prouvé leur robustesse et leur grande efficacité pour résoudre des problèmes compliqués de grande taille tels que les problèmes d'ordonnancement [Vach 00], plusieurs études ont sélectionné les AG comme l'outil le plus performant pour résoudre des problèmes d'ordonnancement de différents types [Goth 04], [Park 03], [Giar 03], [Plip 98]. Aux chapitres suivants nous allons justifier quant à nous ce choix tout en commentant les résultats obtenus lors de la simulation.

Chapitre III

Concepts de base des Algorithmes Génétiques

Les Algorithmes Génétiques sont des algorithmes d'exploration basés sur les mécanismes de la sélection naturelle et de la génétique. Ils utilisent d'une part les principes de la survie des structures les mieux adaptées, et les échanges d'informations pseudo aléatoires, pour former un algorithme d'exploration qui possède certaines caractéristiques de l'exploration humaine [Hop79]. A chaque génération, un nouvel ensemble de créatures artificielles (des chaînes de caractères) est crée en utilisant des parties des meilleurs éléments de la génération précédente, ainsi que des parties innovatrices, à l'occasion. Bien qu'utilisant le hasard, les Algorithmes Génétiques ne sont pas purement aléatoires. Ils exploitent de manière efficace l'information obtenue précédemment pour spéculer sur la position de nouveaux points à explorer, avec l'espoir d'améliorer la performance.

1. Introduction

Les Algorithmes Génétiques (AG) sont des procédures robustes d'exploration d'espaces complexes basées sur la théorie Darwinienne. L'ouvrage originel sur ce thème est celui de John Holland '*Adaptation in Natural and Artificial Systems*' [Hol75]. Un grand nombre d'articles et de thèses de doctorat établissent la validité de la technique pour les applications d'optimisation de fonctions et de contrôles. Ayant été reconnus comme une approche valide des problèmes nécessitant une exploration performante et économique du point de vue du calcul, les Algorithmes Génétiques sont maintenant appliqués plus largement, aux domaines des affaires, à la recherche scientifique en général, ainsi que pour l'ingénierie. Les raisons de ce nombre grandissant d'applications sont claires. Ces algorithmes sont simples d'un point de vue de calcul, mais sont cependant très performants dans leur recherche d'amélioration. De plus, ils ne sont pas fondamentalement limités par des hypothèses contraignantes sur le domaine d'exploitation (hypothèses concernant la continuité, l'existence de dérivées, etc.).

Un AG peut être considéré à la base comme un processus aléatoire. Le travail de recherche commence en plusieurs points dans l'espace des solutions, c'est-à-dire sur une population de points et non pas en un point singulier comme dans la plupart des techniques d'optimisation. Par les opérations de sélection, mutation et croisement, ce sont les bons membres de cette population qui survivent. Donc, ils peuvent passer à la prochaine génération.

A la fin de l'algorithme, une meilleure solution est engendrée parmi les membres suivants. Les caractéristiques des Algorithmes Génétiques peuvent se résumer par les aspects fondamentaux suivants :

- Les AG travaillent sur un codage de l'ensemble des paramètres et non pas avec les paramètres eux-mêmes.

- Les AG cherchent l'optimum à partir d'une population de points et non à partir d'un seul point.

- Les AG utilisent l'information de la fonction objectif et non pas les informations provenant des fonctions dérivées de quelques ordres que ce soit.

- Les AG font appel à des règles de transition probabilistes et non pas à des règles déterministes.

- Les AG possèdent une très bonne exploration des résultats précédents et une exploitation moyenne dans le domaine de recherche.

A partir de ce constat, nous allons décrire l'inspiration de la génétique naturelle, puis nous décrirons le principe utilisé par les AG dans les différentes étapes de ceux-ci. Enfin nous donnerons les différents paramètres à respecter lors de leurs implémentations pour résoudre un problème d'optimisation.

2. La Génétique naturelle

L'évolution dans la nature survient quand des entités ont la capacité de se reproduire, qu'il existe une population de ces entités, qu'il existe une variété (diversité) à travers ces entités et que la survie des entités dépend des différences entre elles. Toute entité vivante possède un génotype et un phénotype (Figure.3.1).

- Le génotype est constitué de gènes situés sur des chromosomes stockés dans le noyau des cellules sous la forme d'une longue chaîne d'acide désoxyribonucléique (ADN) qui constitue l'ensemble des chromosomes ou le génome d'un individu.

- Le phénotype est l'ensemble des protéines et des enzymes qui peuvent être fabriqués à partir de l'ADN. En général, on compte une protéine (un enzyme) par gène. Ce sont les protéines et les enzymes qui dictent la structure et le comportement des cellules qui permettent à un individu de réaliser des tâches dans son environnement, de survivre et de se reproduire à des taux différents.

La reproduction se traduit par la transmission du génome aux individus de la progéniture ce qui permet de préserver les gènes menant à des performances supérieures. Occasionnellement, des processus naturels tels que la mutation et le croisement génétique, introduisent une variation dans les chromosomes. Or les individus les mieux adaptés, c'est.à.dire capables de mieux effectuer les tâches nécessaires à leur survie, se reproduisent à des taux les plus élevés, alors que les individus les moins adaptés se reproduisent à des taux plus faibles. Ce sont les principes de survie et de reproduction. Il s'avère alors qu'une population ayant une grande variété va, de génération en génération, contenir des individus dont le génotype se traduit par une meilleure adaptation, et ceci à cause de la contrainte de la sélection naturelle.

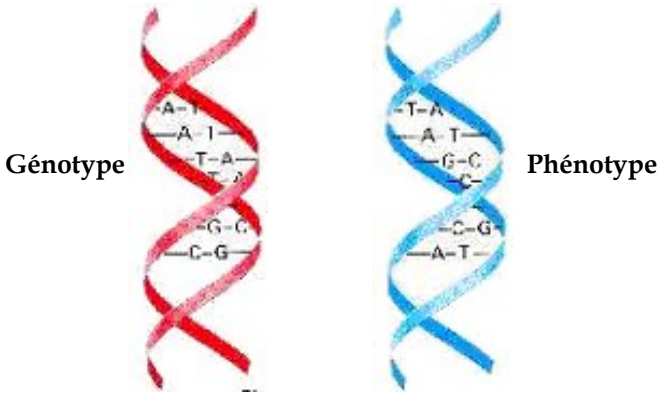

Figure.3.1. Eléments de la génétique naturelle.

3. Exemple d'analogie entre l'évolution Biologique et Binaire

La Figure.3.2 montre un exemple d'analogie entre l'évolution biologique d'un animal choisi (ici le chien) et l'évolution binaire d'un Algorithme Génétique, toutes les deux commencent par une population qui contient des membres aléatoires. Des chaînes binaires (chromosomes) représentent les caractéristiques sélectionnées de chaque chien choisi.

L'aboi fait partie de ces caractéristiques codées en binaire. L'objectif est de créer une nouvelle race de chiens caractérisée par un très haut aboi. Pour cet exemple quatre chiens sont sélectionnés d'une façon aléatoire pour former une population initiale. Cette population correspond à une population de quatre chromosomes codés représentants les caractéristiques de chaque chien respectivement.

Les chiens constituants la population initiale sont alors classés en évaluant leur puissance d'aboi. Deux chiens seront sélectionnés d'une façon aléatoire pour créer deux chiots (croisement en terme d'évolution d'AG). Les chiens qui ont un très haut aboi ont plus de chance d'être sélectionnés (chiens d'abois 7 et 5 par exemple). Les chiots issus de leurs parents possèdent un haut aboi parce que leurs chromosomes contiennent une portion binaire des deux parents. Pour mieux améliorer les caractéristiques d'aboi des deux chiots, on fait introduire des mutations sur leurs chaînes binaires, la mutation qui est une opération aléatoire, consiste à remplacer un bit (gène) en vue d'améliorer les performances des chiots. Ces chiots ainsi crées seront les membres d'une nouvelle génération. On reprend le même processus d'évolution sur plusieurs itérations jusqu'à un critère d'arrêt qui correspond à une population finale possédant des chiens de très haut aboi. La satisfaction de cet objectif est une solution optimale.

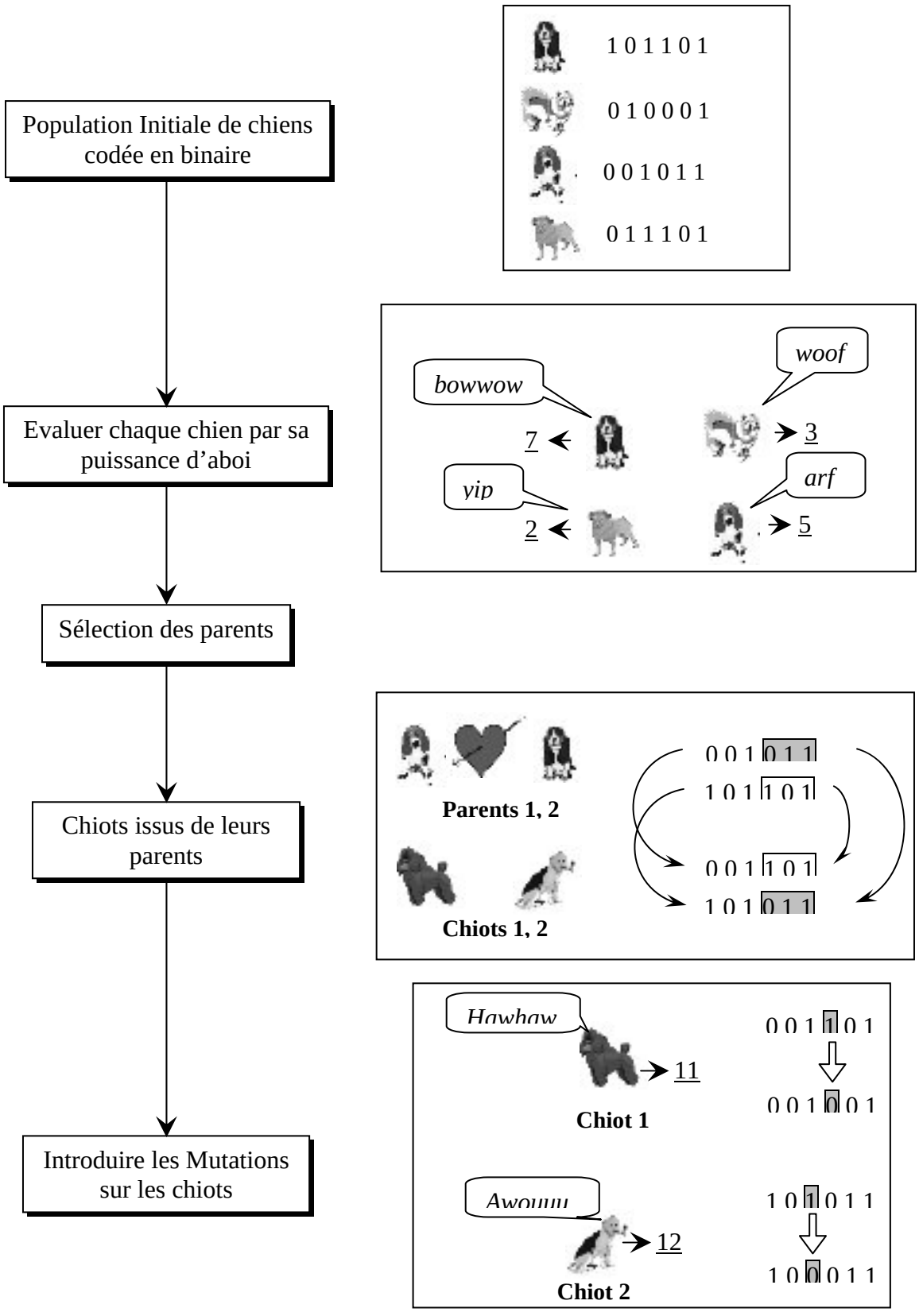

Figure.3.2. Exemple d'analogie entre les évolutions Génétique et Binaire d'un AG

4. Cycle des Algorithmes Génétiques

Un Algorithme Génétique est une procédure itérative basée sur une évolution des populations de solutions candidates [Mela 99]. Un Algorithme Génétique, qui engendre de bons résultats sur des problèmes pratiques, comporte les opérations fondamentales suivantes [Houc 04], [Zegh 03] :

- Création d'une population initiale et codage des variables.
- Evaluation de la population initiale.
  - Calculer l'aptitude de chaque individu de la population initiale.
- Sélection des individus les mieux adaptés, seuls jugés aptes à survivre :
  - Evaluer le degré d'adaptation de chaque individu à son environnement.
  - Sélectionner des paires d'individus (parents) en fonction de la valeur de l'adaptation.
- Engendrer des descendants correspondants à leurs parents.
- Calculer l'aptitude de chacun des descendants.
- Mettre ces nouveaux individus dans la nouvelle génération.
- Répéter, jusqu'à un critère d'arrêt fixé par l'utilisateur.

Pour commencer le cycle de l'Algorithme Génétique on doit avoir les éléments suivants [Lakh 98] :

- Le codage des variables.
- Le choix des opérateurs génétiques et leurs probabilités stochastiques.
- La taille de la population ainsi que les conditions d'arrêt.

5. Description des opérations génétiques

5.1. Codage des variables

L'utilisation d'un Algorithme Génétique nécessite le choix préalable d'un code génétique représentant le problème à traiter [O'Re 05], [Penn 99]. Le choix de ce codage est essentiel car il va déterminer les performances de l'algorithme. Le code est représenté sous forme d'une chaîne de bits ou de caractères, chaîne analogue à un chromosome. Pour l'Algorithme Génétique chaque paramètre dans l'espace des solutions est une chaîne de caractères appelée 'gènes', qui représente une solution

possible, un chromosome est constitué d'un ensemble de gènes et la totalité de chromosomes constitue un individu (génotype).

Suivant la nature du problème d'optimisation rencontré, on peut définir deux types de codage : le codage binaire, et le codage non binaire.

a. Codage binaire

Soit la fonction $F(x)$ représentée par la Figure.3.3, la fonction quantifiée sur la même figure permet de discrétiser la fonction en échantillons quantifiés de niveaux horizontaux égaux. Chaque variable $F(x_i) = y_i$ a une coordonnée quantifiée sur la fonction quantifiée qui correspond à une valeur élevée, moyenne, ou basse. Pour cette exemple, on prend 4 variables de $F(x)$: 0.55, 0.11, 0.95, et 0.63, leurs coordonnées quantifiées sur la fonction quantifiée sont enregistrées sur la Figure.3.4. En général les meilleures valeurs de ces cordonnées correspondent aux valeurs quantifiées moyennes [Haup 04]. Le chromosome codé en binaire indique l'intervalle d'échantillonnage, la variable 0.55 se trouve entre 0.625 et 0.5 donc on lui affecte le codage 100.

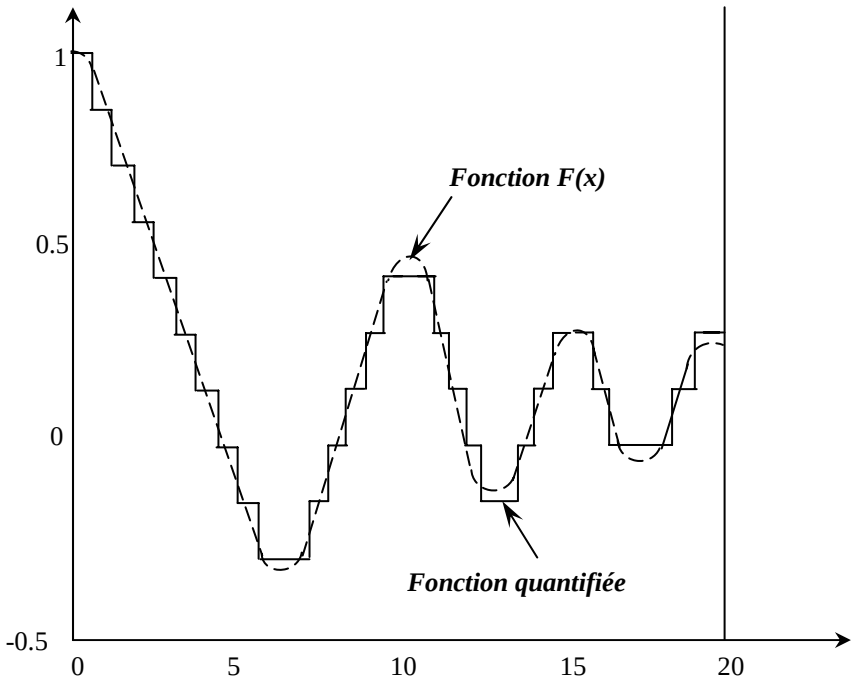

Figure.3.3. Fonction F(x) et sa fonction quantifiée

		Valeurs des Variables				
		0.55	0.11	0.95	0.63	
1.000	111			*		0.9375
0.875	110					0.8125
0.750	101				*	0.6875
0.625	100	*				0.5625
0.500	011					0.4375
0.375	010					0.3125
0.250	001					0.1875
0.125	000		*			0.0625
0.000	Valeurs quantifiées élevées	0.625	0.125	1.000	0.750	
	Valeurs quantifiées basses	0.500	0.000	0.857	0.625	
	Valeurs quantifiées moyennes	0.5625	0.0625	0.9375	0.6875	
		100	000	111	101	

Figure.3.4. Exemple de la représentation binaire des valeurs quantifiées

Les formules mathématiques de codage et de décodage de la variable p_n sont données par :

- **Codage**

$$p_{nom} = \frac{p_n - p_{lo}}{p_{hi} - p_{lo}} \tag{3.1}$$

$$gene[m] = round\left\{ p_{nom} - 2^{-m} - \sum_{p=1}^{m-1} gene[2p]^{-p} \right\} \tag{3.2}$$

- **Décodage**

$$p_{quan} = \sum_{p=1}^{m-1} gene[m] 2^{-m} + 2^{-(m+1)} \tag{3.3}$$

$$q_n = p_{quant}(p_{hi} - p_{lo}) + p_{lo} \tag{3.4}$$

Pour chaque cas :

- p_{norm} : Variable normalisée, $0 \le p_{nom} \le 1$

- p_{lo} : La plus petite valeur
- p_{hi} : La plus grande valeur
- $gene[m]$: Version binaire de p_n
- $round\{\ \}$: Arrondissage vers un entier proche
- p_{quant} : Version quantifiée de p_{norm}
- q_n : Version quantifiée de p_n

Ces principes de codage et de décodage sont applicables lorsque chaque variable peut être représentée par un seul bit (gène ou allèle), mais lorsque les variables sont codées par plusieurs bits (gène multi allèles), chacun d'eux est codé avec une longueur de chaîne propre, ensuite on les concatène de façon à former une chaîne unique. La représentation ci-dessous est un exemple de codage binaire en chromosomes (ou individus) de *N* variables, chacun est codé avec $N_{gène} = 10$ bits,

$$Chromosome = \left[\underbrace{1111001001}_{gène1} \underbrace{0011011111}_{gène2} ..... \underbrace{0000101001}_{géneN_{\text{var}}} \right]$$

Ce chromosome a le total de $N_{bits} = N_{gène} \times N_{\text{var}} = 10 \times N_{\text{var}}$ bits.

b. Codage non binaire

Avec les développements récents, d'autres types de représentation plus sophistiquée sont apparus [Sart 02]. Ainsi, un chromosome peut être une liste des villes à parcourir pour le problème de voyageurs de commerce [Caux 95], un ordre de passage entre différentes tâches sur différentes machines dans un système de production. Ce type de codage sera détaillé pour le cas d'ordonnancement d'un atelier au chapitre IV.

5.2. La Population initiale

La population initiale peut être obtenue en générant aléatoirement les chaînes de l'espace de recherche. Cette méthode a l'avantage de commencer la recherche à partir de diverses solutions de l'espace de recherche, cela donne un point fort de plus aux Algorithmes Génétiques par rapport à d'autres méthodes d'optimisation.

Une population qui contient N_{pop} chromosomes est une matrice de taille $N_{pop} \times N_{bits}$, elle peut être généré en utilisant la fonction aléatoire rand sous Matlab [Math 04].

pop=round (rand (N_{pop} , N_{bits})) (3.5)

Les membres aléatoires de la matrice ainsi générée sont des valeurs comprises entre 0 et 1. La fonction round permet d'arrondir ces valeurs réelles en valeurs entières proches.

5.3. La Fonction d'adaptation :

La fonction *fitness* ''d'adaptation'' *F(x),* mesure la puissance de chaque chromosome à s'adapter, elle décrit l'aptitude d'un chromosome de la population, à optimiser l'objectif. La plus grande valeur de *F(x)* correspond à la meilleure solution.

Cette fonction est choisie en fonction de la valeur de la fonction objectif à traiter. Généralement, si la prédiction de la valeur de l'extremum est inférieure à 1 on prend directement la valeur de la fonction objectif [Rend 95] :

$$fitness = F(x) = F(Pop_i) \tag{3.6}$$

Sinon la fonction est prise pour des raisons opérationnelles en choisissant une certaine échelle telle que la valeur de la fonction objectif sera inférieure à 1 :

$$fitness[pop_i] = \left(\frac{1}{F(x)}\right).100 \qquad \text{D'où} \quad 0 < fitness[pop_i] < 1 \tag{3.7}$$

D'autres calculent l'indice d'adaptation comme un pourcentage entre la valeur du point considéré et la somme de toutes les valeurs de la population [Park 03] :

$$fitness[pop_i] = \frac{F[pop_i]}{\sum_{i=1}^{n} F[pop_i]}. \tag{3.8}$$

Ce pourcentage peut être pris comme un indice de sélection (méthode roulette).

5.4. La Sélection :

La sélection permet de choisir quels chromosomes vont se reproduire afin de faire évoluer les populations. Cette sélection est faite à partir d'une fonction qui évalue les chromosomes. Cette fonction donne un indice d'évaluation à chaque chromosome suivant son aptitude à résoudre le problème posé. Un nombre identique de chromosomes sera ainsi sélectionné aléatoirement avec une probabilité plus ou moins forte suivant leur indice. La sélection peut se faire soit par roulette ou bien par classement [Mar et al 99.a] :

a. La méthode roulette :

C'est une méthode classique, elle est utilisée dans la majorité des applications. Les chromosomes "parents" sont sélectionnés suivant leurs indices. Plus leurs indices sont élevés, plus ils ont la chance d'être sélectionnés. Cette méthode passe par 6 étapes essentielles [Math 04] :

- **Etape 1 :** Calculer la fonction fitness $F(pop_i)$ de chaque chromosome de la population.

- **Etape 2 :** Calculer la valeur moyenne de $F(x)$ pour chaque population.

$$F(pop_i)_{moy} = \frac{\sum_{i=1}^{n} F(pop_i)}{taille\,de\,la\ population = N_{pop}} \tag{3.9}$$

- **Etape 3 :** Déterminer le nombre prévu de copies de chaque chromosome dans la population intermédiaire (mating pool ou bassin de reproduction).

$$Nombre\_de\_Copies(pop_i) = \frac{F(pop_i)}{F(pop_i)_{moy}} \tag{3.10}$$

- **Etape 4 :** Déterminer la probabilité de sélection de chaque chromosome.

$$\Pr obabilité\_de\_sélection(pop_i) = \frac{Nombre\_de\_Copies(pop_i)}{taille\,de\,la\ population = N_{pop}} \tag{3.11}$$

- **Etape 5 :** Déterminez la probabilité cumulée pour chaque chromosome par l'addition de la probabilité cumulée de chromosome précédent avec la probabilité de sélection de chromosome actuel. Cette probabilité a une valeur entre 0 et 1.

$$\begin{aligned}\Pr obabilité\_cumulée(pop_i) = {} & \Pr obabilité\_de\_sélection(pop_i) \\ & + \Pr obabilité\_cumulée(pop_i - 1)\end{aligned} \tag{3.12}$$

Etape 6 : Choisir un nombre aléatoire entre 0 et 1. Le chromosome d'une probabilité cumulée supérieure à ce nombre aléatoire doit être sélectionné dans la population intermédiaire (mating pool).

b. La méthode de classement

Les chromosomes d'une population sont classés dans une liste selon l'ordre croissant de leur fitness, ensuite la sélection est proportionnelle à leur rang dans la liste de la population. Cette méthode est utilisée pour une fonction de fitness mal définie, ou quand les valeurs de fitness sont très proches, elle est rarement utilisée dans les applications d'optimisation [Zegh 03].

5.5. Modèle de reproduction

Pendant la phase de reproduction, les chromosomes sont sélectionnés et leurs structures sont modifiées pour générer de nouveaux individus qui vont former la génération suivante. Différentes stratégies existent pour maintenir la population constante.

Les parents sélectionnés par roulette sont introduits dans le bassin de reproduction (*mating pool* qui est considéré comme une population intermédiaire) où ils seront de nouveaux choisis aléatoirement pour voir leurs chromosomes subir des transformations par les opérateurs génétiques. Les deux principaux opérateurs de base sont le croisement et la mutation. Le croisement réalise une opération binaire (ou sexuée), et nécessite deux parents. La mutation est une opération unaire (ou asexuée), utilisée pour introduire une faible variation dans la solution ou changer la direction de recherche.

5.5.1. Le Croisement

Le croisement ou l'hybridation permet de créer une nouvelle population à partir des chromosomes choisis par l'opération de sélection. C'est un échange de par blocs d'éléments entre deux chromosomes parents pour générer deux autres chromosomes enfants. Il existe deux types de croisement [Lakh 98] :

a. Croisement simple (par point)

Un site de croisement est choisi au hasard avec une probabilité p_{cross} (taux de croisement) sur la longueur de chaque chromosome parent et une coupe du chromosome est réalisée (appelée site de croisement). Cette coupe produit deux morceaux que l'on permute. Les chaînes enfants résultantes, contiennent chacune un morceau hérité de chacun des parents (Figure.3.5).

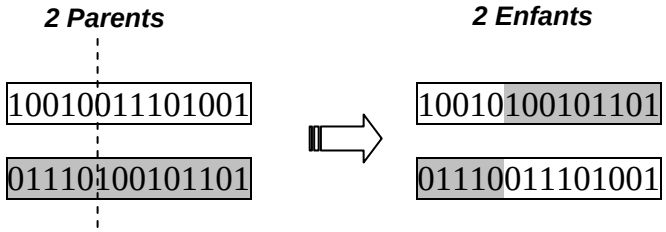

Figure.3.5. Croisement par un point de coupure.

b. Croisement multiple (uniforme)

Le croisement simple peut être généralisé en croisement en *k* points de coupe, générant *k+1* sous chaînes pour chaque chromosome. Les deux chromosomes fils sont obtenus par concaténation de ces sous chaînes en alternant les parties venant de chaque parent (Figure.3.6).

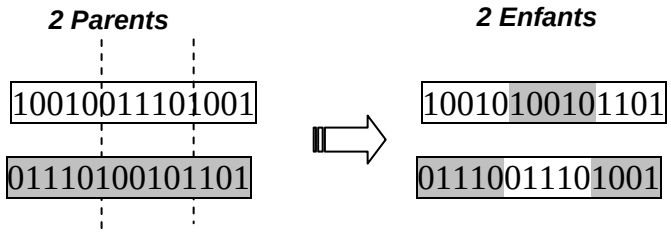

Figure.3.6. Croisement par deux points de coupure.

5.5.2. La Mutation

La mutation est une forme de l'adaptation génétique de l'individu à son environnement. Elle permet de bien faire une recherche de la solution dans tout l'espace de recherche. La mutation a pour but de modifier un chromosome de la population afin que la population puisse se diversifier (Figure.3 5). Elle se manifeste par une modification brutale d'un gène. Le phénomène se produit rarement (de 0.1% à 5% de la population).

L'opérateur de mutation est un opérateur unaire générant un seul individu à partir d'un seul individu. Il consiste à modifier la valeur d'un élément de la chaîne. Pour le codage binaire il suffit de changer un bit choisi aléatoirement de 0 à 1 ou vice versa avec une probabilité p_{mut} (taux de mutation) (Figure.3.7).

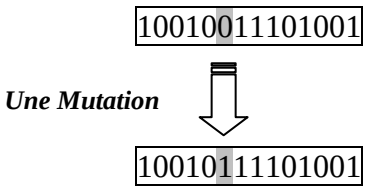

Figure.3.7. Représentation schématique d'une Mutation dans un Chromosome.

5.5.3. L'élitisme :

L'élitisme est une opération spéciale [Sart 02], il peut intervenir en parallèle de la sélection et de croisement. Il permet de garder le meilleur chromosome lors de la sélection et à l'introduire dans la nouvelle population lors de l'hybridation. Cela permet de ne pas perdre le meilleur chromosome qui aurait pu avoir des chromosomes fils moins bons à la génération suivante. Cet élément peut améliorer considérablement les performances sur certains problèmes. Il est conseiller de l'utiliser si la taille de la population est élevée. Le principal inconvénient de l'élitisme est que l'algorithme converge plus rapidement vers une solution unique [Zegh 03].

6. Choix des valeurs des paramètres

Les valeurs des paramètres d'un Algorithme Génétique doivent être choisies avec les considérations suivantes [Sart 02]:

6.1. Taille de la population

Elle doit être judicieusement choisie en fonction de la taille du problème :

- *Trop faible* : l'AG n'a pas assez d'échantillons de l'espace de recherche.

- *Elevée* : une taille prévient contre la convergence prématurée.
- *Trop élevée* : le nombre élevé d'évaluations de la fonction *fitness* par génération ralenti la convergence.

6.2. Taux de croisement

Plus le taux croisement p_{cross} est élevé, plus il y aura de nouvelles structures qui apparaissent dans la population.

- *Trop élevé* : les bonnes structures risquent d'être cassées trop vite par rapport à l'amélioration que peut apporter la sélection.
- *Trop faible* : la recherche risque de stagner à cause du faible taux d'exploration.

Le taux habituel de croisement est choisi entre 60% et 80% [Sart 02].

6.3. Taille de mutation

La mutation est un opérateur secondaire pour introduire la diversité dans la population. Son taux d'application p_{mut} est choisi entre 0.1% et 5%

- *Trop élevé* : rend la recherche trop aléatoire.
- *Trop faible* : la recherche risque de stagner à cause du faible taux d'exploration.

Si la taille de la population est faible, un taux de croisement faible doit être combiné avec un taux de mutation élevé. Les observations sont valables pour les fonctions continues sans contraintes avec codage binaire.

7. Prise en compte des contraintes

La plupart des problèmes réels contiennent des contraintes. La formulation classique de ces problèmes est de type :

Minimiser $f(x)$, sous contraintes :

$$\begin{cases} g_i(x) \le 0 & i = 1,.....,m \\ h_j(x) \le 0 & j = 1,.....,n \end{cases} \tag{3.13}$$

La méthode généralement retenue pour introduire ces contraintes dans la formulation utilisée avec les AG est de transformer le problème contraint en un problème non contraint avec fonction de pénalité (Sart 02). Au lieu de minimiser la fonction objectif $f(x)$, on minimise la fonction pénalisée qui a la forme :

$$F(x) = f(x) + \sum_{i=1}^{m} r_i \Phi_1(g_i) + \sum_{j=1}^{n} s_j \Phi_2(h_j) \tag{3.14}$$

Où (r_i, s_j) sont des coefficients de pénalité et (Φ_1, Φ_2) des fonctions de pénalisation. Voici des fonctions traditionnellement retenues

$$\Phi_1(g_i) = \begin{cases} g_i^2 & si\ g_i > 0 \\ 0 & si\ g_i \le 0 \end{cases} \tag{3.15}$$

$$\Phi_2(h_j) = h_j^2 \tag{3.16}$$

Les coefficients (r_i, s_j) doivent être ajustés de sorte que chaque contrainte ait le poids relatif qu'on souhaite lui voir jouer. La présence de contraintes égalités ne facilite généralement pas l'obtention d'une convergence rapide et d'un optimum de bonne qualité. Théoriquement l'algorithme converge vers une solution réalisable lorsque les coefficients de pénalité tendent vers l'infini. En pratique, il n'est souhaitable de prendre une valeur très grande pour ces coefficients que lorsque le problème est faiblement contraint et que, de ce fait il existe un domaine de solutions réalisables assez vaste pour qu'il soit possible d'en trouver au moins une après un nombre d'itérations raisonnable. Lorsqu'un problème est fortement contraint, il vaut mieux utiliser des valeurs de pénalisation raisonnables au départ puis les augmenter progressivement au fur et à mesure des itérations (on parle de pénalisation dynamique). Cela permet de ne pas rejeter au départ les points faiblement pénalisés. Ayant ainsi des chances de se reproduire, ils peuvent aider à trouver un chemin conduisant progressivement vers l'intérieur du domaine de solutions réalisables.

8. Champs d'application

Les applications des Algorithmes Génétiques sont multiples : optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), traitement d'images (alignement de photos satellites, reconnaissance de suspects…..), optimisation d'emplois de temps, optimisation de formes en design, contrôle des systèmes industriels, optimisation de l'ordonnancement, résolution des problèmes de voyageur de commerce, apprentissage des réseaux de neurones, etc.

Les AG peuvent être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire…) car la population peut s'adapter à des conditions de changeantes. En particulier, ils supportent bien l'existence de bruit dans la fonction à optimiser. Ils peuvent aussi servir à déterminer la configuration d'énergie minimale d'une molécule ou à modéliser le comportement animal par exemple

9. Conclusion

Les Algorithmes Génétiques correspondent à des méthodes d'optimisation nouvelles, leur processus permet d'atteindre une meilleure solution (optimale) en se basant sur le principe de l'évolution génétique. Leur fonctionnement est très différent des algorithmes classiques d'optimisation. On retient quatre points principaux :

1. Utilisation d'un codage des paramètres, et non des paramètres eux-mêmes.
2. Ils travaillent sur une population de points, au lieu d'un point unique.
3. Utilisation des valeurs de la fonction à optimiser, et non de leur dérivée.
4. Utilisation des fonctions de transition probabilistes, non déterministes

Le calcul pour obtenir une nouvelle génération avec les AG se décompose couramment en 4 parties : la Sélection, le Croisement, la Mutation et la Reproduction.

Les opérateurs dans les Algorithmes Génétiques n'ont pas la même importance. Ainsi, l'opérateur de croisement a une place importante dans le processus d'exploration de l'espace des solutions comme le prouve la théorie. La mutation, quant à elle, ne joue qu'un rôle secondaire par rapport au croisement, mais son influence sur l'exploration de l'espace des solutions n'est pas à négliger.

Chapitre IV

Simulation et Résultats

Dans ce chapitre, nous allons donner et justifier les choix réalisés pour le problème d'ordonnancement auquel nous allons appliquer les Algorithmes Génétiques. Le problème que nous avons choisi est un atelier d'extrusion plastique. Nous verrons à chaque étape comment ces choix ont été implémentés. En premier lieu, nous montrerons comment nous avons décidé de choisir ce type de chaînes de production. Ensuite, nous décrirons notre atelier d'une façon simple, claire, générale et encore une fois proche de la réalité. Nous verrons alors les différentes étapes pour une implémentation qui respecte le problème choisi ainsi que les descriptions des chapitres précédents. Nous expliquerons la méthode utilisée pour la résolution du problème d'ordonnancement et son optimisation avec les Algorithmes Génétiques, nous verrons aussi le fonctionnement du programme ainsi que la représentation des résultats sous diagrammes et graphes. Enfin nous commentons nos résultats tout en appréciant l'avantage mené par notre méthode d'optimisation choisie.

1. Introduction

En ordonnancement de production, il est souvent utile de choisir une chaîne de production pour tester le modèle d'ordonnancement d'atelier que l'on souhaite faire. Cependant, tout choix doit faire face à d'importants problèmes d'ordonnancement et de gestion des perturbations dans la production. Dans ce chapitre on propose d'étudier un atelier de fabrication du plastique par extrusion.

Ce procédé de fabrication en chaînes continues est décomposé en plusieurs étapes. Différentes machines se succèdent, dans le cadre de cette étude, nous nous intéressons à la ligne de production en tant que telle, et non pas à chacune des machines séparément, ces lignes fonctionnent 24 heures par jour, toute l'année, sauf en cas de panne, de maintenance, de surcapacité (ce qui peut arriver exceptionnellement lors des périodes de faibles demandes).

Sur une même ligne de production il y a du passage de production d'une référence à une autre suivant les commandes et les besoins des clients, il peut y avoir un changement de format du film de plastique, de couleur, de poids spécifique, ou son décor esthétique pour chaque commande. Ces changements sont plus ou moins longs en fonction des commandes enchaînées et de personnels qualifiés pour le réglage des machines d'extrusion. Ils varient entre plusieurs minutes pour un changement de décor, quelques heures pour un changement de format. Donc il faut comprendre que d'une part, ces changements de production pénalisent fortement la production car ils introduisent des temps non productifs (perte de temps machine) et peuvent d'autre part, coûter très cher à cause du personnel supplémentaire spécialisé nécessaire ainsi que les produits rebutés suite aux différents réglages.

Chaque atelier d'extrusion propose la répartition de l'ensemble du travail sur les machines, en respectant au mieux un ensemble de contraintes (de précédences, de ressources,….) et en visant un objectif (minimisation des coûts liés au changement de commandes, et de temps lié a l'exécution des tâches ...). En informatique, on fait également régulièrement appel aux problèmes d'ordonnancement.

Nous consacrons notre travail non seulement à proposer un ordonnancement classique et le résoudre, mais aussi le rendre optimal. L'objectif est la minimisation du coût de transitions entre les commandes enchaînées sur une même ligne de production, minimiser aussi le temps total de l'ordonnancement *''le makespan''*, ces

deux paramètres ‘’*coût* ‘’et ‘’*makespan*’’ qui seront introduits dans une fonction objective *f* qui reste à déterminer. On propose pour résoudre ce problème un outil d’optimisation avec les Algorithmes Génétiques, nous justifions le choix de cet outil génétique tout en commentant les résultats et les graphes de notre implémentation.

2. Description de l'Atelier d'Extrusion

Ce procédé de mise en œuvre en continu permet de transformer les poudres et granulés de polyéthylène en films dont la longueur n'est pas limitée, ce film continu sera extrudé sur la même ligne de production sous forme de bobines suivant les commandes des clients. Chaque commande qui constitue un ensemble de bobines est définie par un grade lié à sa matière première, et un poids spécifique lié aux dimensions des bobines.

Le procédé de production implique plusieurs étapes, chaque étape aura lieu sur une machine différente. La Figure.4.1 représente la chaîne de production en extrusion.

La première étape dans le processus de fabrication est de remplir les silos avec la matière première, l'approvisionnement s'effectue avec de grandes quantités afin d'éviter toute interruption probable en matière. L'ensemble des silos 1, 2, et 3 contiennent la même matière de base et cela, pour un ensemble de commandes qui aura lieu sur cette ligne de production tel que le montre le haut de la Figure.4.1.

Le transfert de cette matière de base s'effectue à l'aide des compresseurs par l'intermédiaire de tuyaux qui relient les silos et les conteneurs, ces derniers sont de capacité de stockage inférieure à celle des Silos reçoivent la matière première par quantités discontinues. Suivant les besoins des clients et suivant l'usage du plastique on rajoute différents additifs à la matière de base dans ces conteneurs, les additifs augmentent les caractéristiques thermomécaniques du plastique, et donnent au plastique différentes couleurs et apparences avec un poids spécifique différent aussi.

L'ensemble de matière de base et additifs seront enchaînés ensuite en petites quantités à l'intérieur d'un four électrique, ce dernier permet de mener la matière de l'état solide à l'état plastique ou fondu. Un dispositif de compression sert à gonfler la matière fondue sous forme de longue bulle, un autre dispositif de refroidissement placé au niveau de la gaine déjà gonflée arrête l'étirage du film. Ce phénomène est provoqué par la solidification de la matière refroidie qui supporte alors les contraintes provenant du gonflage. Le film, en forme de bulle, est ensuite aplati par un dispositif appelé " foulard ", composé de deux panneaux convergents, vers les rouleaux pinceurs. L'étirage en direction transversale est obtenu par la suppression de l'air dans la gaine extrudée et pincée par deux rouleaux tireurs (un rouleau est recouvert de caoutchouc et l'autre est en acier) .L'étirage de film se maintient jusqu'à l'extrudeuse qui extrude le film sur des mandrins sous forme de bobines de différents formes et poids.

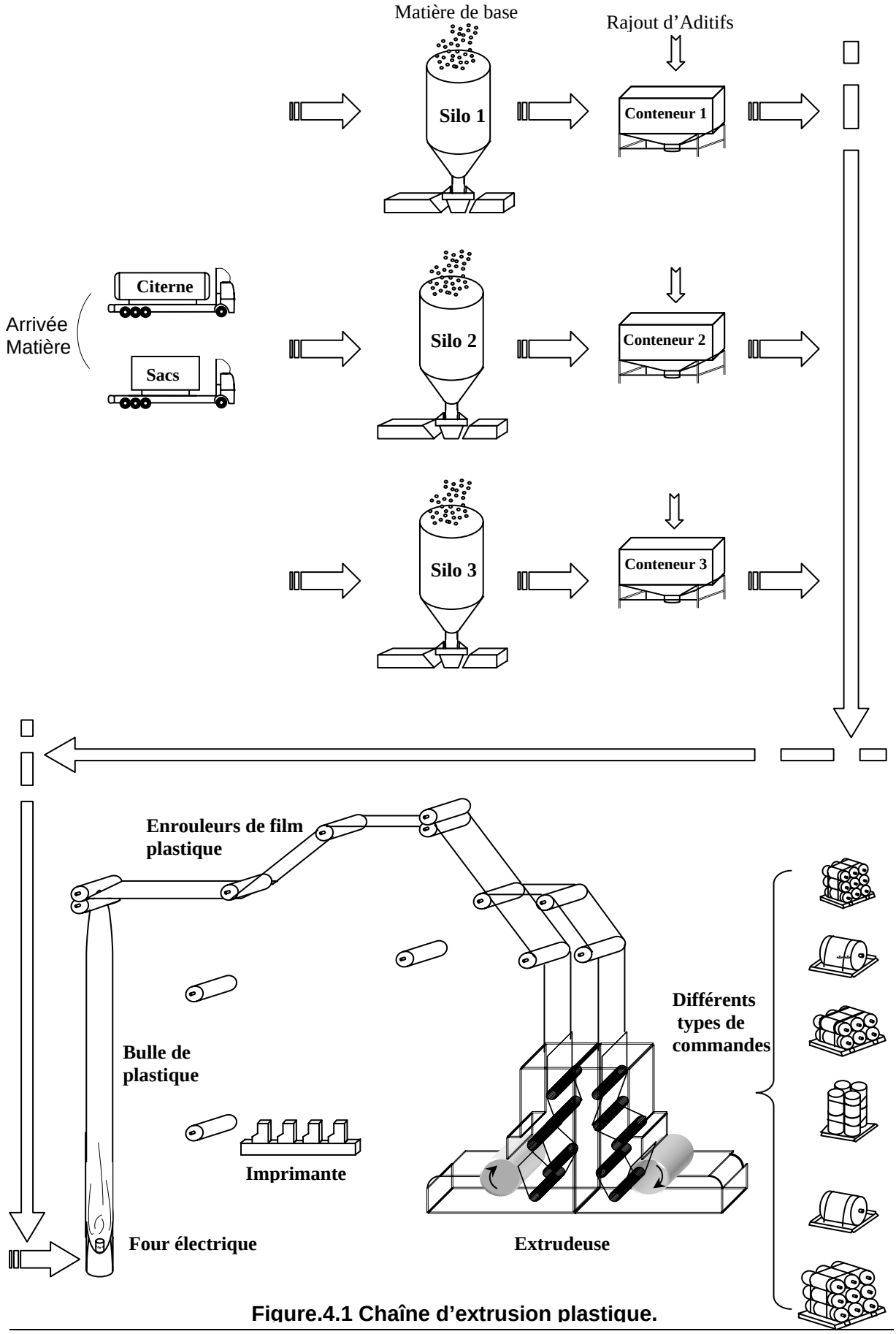

Figure.4.1 Chaîne d'extrusion plastique.

3. Définition du problème

On considère une simple ligne de production (Figure.4.1), qui a pour entrée l'un des trois silos, et qui se termine par la machine extrudeuse.

Une particularité importante, liée au type du plastique fabriqué, est la saisonnalité des demandes. En effet, en préparation de l'été, les clients passent de fortes commandes à partir des mois de mars avril, et pour éviter les stocks, diminuent fortement leurs demandes après septembre. Dans notre travail on considère qu'on va exécuter six commandes dans 36 heures, chaque commande doit avoir un poids, une forme et un grade différents, Le tableau.4.1 suivant, donne plus de détails sur les ordres de grandeurs des différentes commandes considérées.

Tableau.4.1 Ordre de grandeur de différentes commandes.

COMMANDE	GRADE	NOMBRE DE BOBINES	POIDS /BOBINE (KG)
1	A	18	134
2	B	32	341
3	C	4	503
4	D	4	605
5	E	8	466
6	F	4	623

Avant de commencer, on considère l'ensemble des silos et conteneurs comme des machines, nous rappelons ainsi la hiérarchie existant entre les différentes configurations de ces machines. La configuration la plus simple est composée de 3 machines d'entrée (les silos) parallèles, 03 machines (des conteneurs) intermédiaires parallèles et une machine extrudeuse à la sortie. Chaque machine a une capacité de stockage différente. L'ensemble de ces machines peut exécuter des tâches *i* dans une durée opératoire p_i et qui dépendent des vitesses de transfert de matière et d'enroulement du film plastique. Le tableau.4.2 donne la contenance maximale de chaque machine.

Tableau.4.2 Caractéristiques stockage des machines.

MACHINE	CONTENANCE MAXI (KG)
SILO1	**23 000**
SILO2	**20 000**
SILO 3	**24 000**
CONTENEUR 1	**2500**
CONTENEUR 2	**2500**
CONTENEUR 3	**2500**

Le tableau.4.3 donne les caractéristiques temps de chaque machine associées à chaque type de commandes. Ces valeurs dépendent de la nature d'écoulement de chaque matière et de la nature du gonflage et refroidissement de bulle de plastique. Pour simplifier notre étude on suppose que la matière se transfert avec un seul mode d'écoulement et cela pour toutes les commandes. Les valeurs enregistrées sur ce tableau représentent le cumul de temps sur chaque machine nécessaire pour transférer le film plastique d'une machine à une autre terminant par l'enrouler en bobines sur l'extrudeuse. En ordonnancement on fait appel aux temps d'exécution des tâches, c'est le moyen pour calculer la valeur du *Makespan* pour chaque ordonnancement possible.

Tableau.4.3 Temps d'exécution tâches (Min).

	COMMANDE 1	COMMANDE 2	COMMANDE 3	COMMANDE 4	COMMANDE 5	COMMANDE 6
SILO 1	200	190	200	180	180	195
SILO 2	220	180	200	190	180	190
SILO 3	200	200	190	190	180	190
CONTENEUR 1	140	120	100	95	120	130
CONTENEUR 2	140	120	110	95	120	120
CONTENEUR 3	140	140	110	95	130	120
EXTRUDEUSE	90	85	90	75	90	80

Les problèmes d'ordonnancement rencontrés dans les ateliers de plasturgie en particulier ceux d'extrusion, sont des problèmes où le temps nécessaire pour passer de l'exécution d'une tâche à une autre dépend de l'enchaînement de ces tâches. Ainsi, la modification d'une taille d'une bobine sur une ligne peut entraîner un arrêt de cette dernière, pour cause de réglage, d'une durée pouvant aller jusqu'à des heures ce qui pénalise très fortement la production. Lors du passage d'une commande d'une couleur et grade A à une autre commande d'une couleur et grade différents, une quantité de plastique est jetée jusqu'à ce que la bonne couleur et le nouveau grade soient atteints. Dans ce cas, la perte induite par le changement concerne non seulement une perte de temps disponible à la production, mais également une perte de matière.

Dans la littérature, ces problèmes sont appelés problèmes d'ordonnancement avec temps de changement dépendant de la séquence et sont caractérisés par la présence d'un s_{ij} [Plip 98], représentant le temps de changement nécessaire pour passer de la tâche *i* à la tâche *j* sur la même machine, il englobe le temps nécessaire pour le réglage de machines, le temps nécessaire pour passer à un nouveau grade de matière jusqu'à atteindre une qualité de plastique sans défauts. Ce temps peut se varier en fonction de la qualité du film plastique et sa couleur, il dépend aussi du personnel qualifié pour le réglage des machines, et le garnissage des conteneurs en matière additive. La valeur de s_{ij} est inclue dans les valeurs enregistrées au tableau.4.3 ci-dessus.

Le tableau.4.4 représente les valeurs des coûts de transitions en euros par kilogramme lors du passage d'une commande à une autre (pour notre étude on traite l'enchaînement de six commandes).

Tableau.4.4 Coût associé aux transitions entre commandes en Euros.

De /à	COMMANDE 1	COMMANDE 2	COMMANDE 3	COMMANDE 4	COMMANDE 5	COMMANDE 6
COMMANDE 1	////	1680	1553	1670	1400	1011
COMMANDE 2	1680	////	1162	1243	1231	1332
COMMANDE 3	1553	1162	////	2800	2598	1702
COMMANDE 4	1670	1243	2800	////	2520	1345
COMMANDE 5	1400	1231	2598	2520	////	2480
COMMANDE 6	1011	1332	1702	1345	2480	////

Les machines extrudeuses modernes fabriquent des bobines d'une même commande en gardant le même poids et le même métrage, donc l'ensemble de bobines qui forme une commande a le même poids, et le même temps de production.

La description ci-dessus du procédé de fabrication est une version simplifiée. Plusieurs suppositions ont été effectuées pour le fonctionnement du système, en vue de le rendre simple et clair. Ces suppositions comprennent :

1. La dynamique du flux de plastique, est simplifiée pour quantité matière entrante = quantité films sortant.

2. Le refroidissement et le découpage du plastique en bobines ne sont pas pris en considération.

3. Le modèle se compose d'une seule ligne de production, il a comme entrée l'un des silos, comme sortie la machine extrudeuse.

4. Les machines (silos, conteneurs et extrudeuse) fonctionnent en parallèle.

5. Les perturbations liées aux arrêts brusques des machines sont ignorées.

6. Les interventions périodiques de maintenance des machines ne seront pas prises en compte.

7. les contraintes en amont et en aval de l'atelier liées à l'approvisionnement en matière première et le stock des commandes ne sont pas pris en considération.

4. Algorithmes de résolution

De manière générale, les Algorithmes Génétiques ont montré leur puissance pour obtenir de bons résultats pour des problèmes complexes. Partant de ce constat, nous les avons adaptés pour les utiliser à notre problème d'ordonnancement d'atelier d'extrusion. A partir des chapitres que nous avons vu précédemment, nous nous proposons dans cette sous-section de mettre en évidence l'apport des Algorithmes Génétiques (AG) pour la résolution de notre problème d'ordonnancement.

Nous présentons d'abord la technique de codage que nous avons utilisé ainsi que les différentes méthodes propres aux Algorithmes Génétiques. Enfin, nous présentons les résultats inhérents au problème que nous avons traité.

4.1. Codage du problème en chromosome

La première étape dans les Algorithmes Génétiques est de transformer le problème de termes réels en un problème de termes biologiques. La tâche ici est de trouver une représentation de l'ordonnancement à travers un chromosome.

Comme il était mentionné au paragraphe 3 qui défini la problématique, l'ordonnancement est basé sur la production de 6 commandes de plastique, chaque commande est d'une qualité et d'un poids différents. Les tableaux 4.3 et 4.4, définissent la qualité des commandes et le temps de production pour chacune d'entre elles en détail.

Le procédé de production de toutes les commandes s'effectue sur une seule ligne de production qui contient 3 types de machines : le Silo, le Conteneur et l'Extrudeuse.

En appliquant l'ordonnancement pour la fabrication de l'une de six commandes, il y'a un Silo parmi les 3 Silos (Silo 1, Silo 2, Silo 3) à choisir comme le début de la ligne de production, ces derniers Silos sont couplés aux conteneurs (conteneur1, conteneur2, conteneur3) respectivement. Cela signifie que la production de l'une des commandes sera effectuée entre 3 choix : (Silo 1, Conteneur 1), (Silo 2, Conteneur 2), ou (Silo 3, Conteneur 3).

Selon l'ordre de passage des commandes sur la ligne de production, le coût de transitions est calculé et enregistré dans le Tableau.4.4. En outre le *makespan* qui définit le temps total d'exécution des tâches sera calculé sur chaque séquence d'ordonnancement, ces deux facteurs seront détaillés lors de l'implémentation de notre programme.

Notre approche a été choisie pour coder l'ordonnancement en chromosome. Nous avons 6 commandes (1, 2, 3, 4, 5 et 6), et 3 possibilités pour choisir la machine d'entrée (l'un des 3 silos). Chaque commande peut choisir l'un des 3 silos comme machine de démarrage, cela crée un espace de solutions qui peut atteindre 6 ! = 720 solutions possibles.

La convention utilisée pour crypter un chromosome est indiquée sur la Figure.4.2. Notons que cette figure représente un exemple d'une solution possible de l'ordonnancement, ce chromosome représente le codage non binaire de 6 commandes (codés de 1 à 6 respectivement), chaque commande est couplée à une machine d'entrée (l'un des silo 1, 2 ou 3), ses silos sont codés en binaire aussi sur la droite des commandes. Les machines couplées sont étiquetées dans le chromosome comme suit :

Silo 1, Conteneur 1 : ***Code 1***

Silo 2, Conteneur 1 : ***Code 2***

Silo 3, Conteneur 3 : ***Code 3***

De la même manière, on a étiqueté les commandes dans le même chromosome :

Commande 1 : ***Code 1;*** Commande 2 : ***Code 2***

Commande 3 : ***Code 3;*** Commande 4 : ***Code 4***

Commande 5 : ***Code 5;*** Commande 6 : ***Code 6***

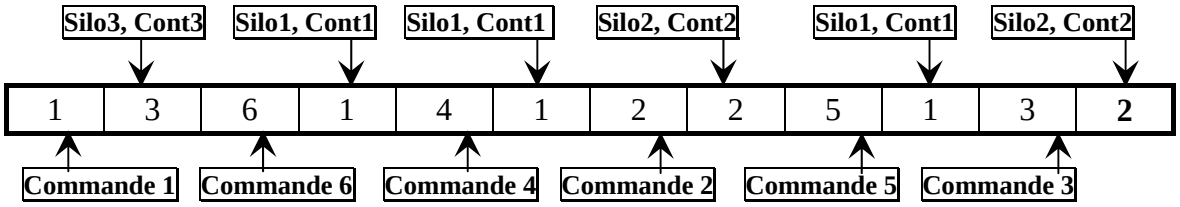

Figure.4.2 Représentation d'une solution possible d'Ordonnancement.

La séquence 13, 61, 41, 22, 51, 32 représente un ordonnancement possible à notre problème, on le schématise sur le diagramme de Gantt suivant :

Machine	Commandes 'Cm'/Temps 'Minutes							
Silo 1	*Cm4*	*Cm5*	*Cm6*					
Silo 2	*Cm2*	*Cm3*						
Silo 3	*Cm1*							
Conteneur1		*Cm4*	*Cm5*	*Cm6*				
Conteneur2		*Cm2*	*Cm3*					
Conteneur3		*Cm1*						
Extrudeuse			*Cm2*	*Cm1*	*Cm4*	*Cm3*	*Cm5*	*Cm6*

Figure.4.3. Ordonnancement représenté par le codage 13, 61, 41, 22, 51, 32

4.2. Population et Chromosomes

La population initiale qui constitue l'ensemble des chromosomes est générée d'une façon aléatoire. La population initiale contient des solutions possibles du problème. Par défaut une valeur de 24 chromosomes a été choisie. Cette valeur peut être changée dans le fichier de ''paramètres_initiaux.mat'' dans le programme de simulation qui a été développé.

4.3. *Makespan* et *coût* de transitions

Afin de déterminer l'ordonnancement optimal, le *makespan* et le coût total (des transitions entre les six commandes) produits par un ordonnancement donné doivent être minimisés. La Figure.4.4 montre une représentation d'un *makespan* sous diagramme de Gantt pour un exemple d'ordonnancement donné. On suppose que le codage de cet ordonnancement est 31 12 22 (on rappelle que cet exemple est donné pour 3 commandes). On remarque que la fabrication de la commande 3 nécessite un temps de 370 minutes, l'extrudeuse commence à fabriquer la commande 1 qui suit, d'une façon enchaînée rentre la commande 2 en production. Le *makespan* de cet ordonnancement est de 545 minutes. Notons que le *makespan* représente le temps total nécessaire pour la fabrication des 3 commandes, y inclut le temps de passage entre commandes (de transitions). Les coûts associés aux transitions sont calculés au tableau.4.4.

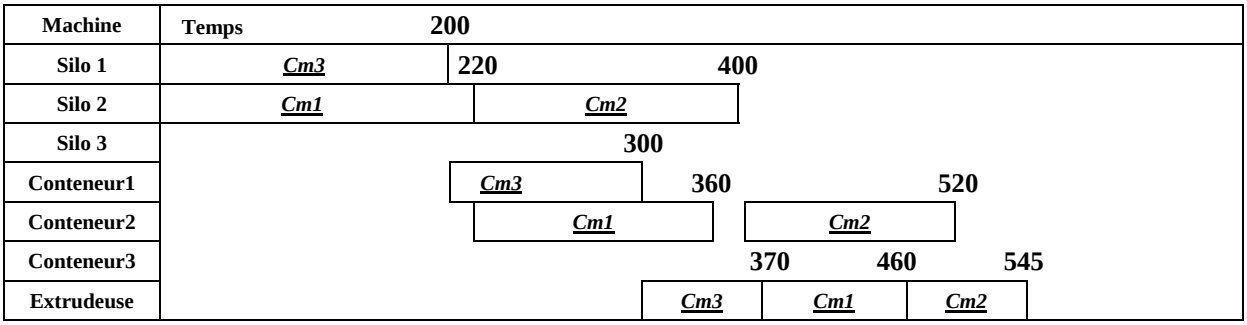

Figure.4.4. Le *Makespan* pour l'Ordonnancement représenté par le codage 31, 12, 22

4.3.2 Fonction multi-objective et fonction *Fitness*

Pour optimiser notre ordonnancement, des objectifs multiples et concurrents doivent être satisfaits (voir chapitre III). La résolution avec les Algorithmes Génétiques consiste à définir une *fonction multi-objective f(x)* à optimiser, pour notre cas cette fonction combine les objectifs coût et temps (*makespan*), elle traduit l'objectif à atteindre pour optimiser notre ordonnancement. On considère l'ordonnancement qui a la plus petite valeur *f(x)* comme optimale.

Cette fonction est définie comme suit :

$$f(x) = \alpha.C + \beta.M \tag{4.1}$$

D'où :

- α *:* poids de pondération pour C.
- C *: coût de transition pour un ordonnancement donné.*
- β *: poids de pondération pour M.*
- M *makespan pour un ordonnancement donné.*

Les poids α et β sont utilisés pour exprimer l'importance relative du coût et du *makespan*. Par défaut Ces facteurs ont pour valeur de 0.1 et 36 respectivement.

La fonction *fitness* ‘’d’adaptation’’ *F(x)*, mesure la puissance de chaque chromosome à s’adapter, pour notre problème cette fonction décrit l’aptitude d’un ordonnancement (chromosome), à minimiser l’ensemble coût et *makespan*. La plus grande valeur de F(x) correspond à la meilleure solution. La relation entre la fonction multi-objective et la fonction *fitness* est donnée par :

$$F(x)=\left(\frac{1}{f(x)}\right).100 \tag{4.2}$$

Remarquons que la valeur maximale de F(x) résulte du de la valeur minimale de *f(x).*

4.4 Méthode de sélection

L’une des méthodes de sélection utilisée dans notre étude est la méthode de sélection par roulette, *‘’roulette wheal selection’’*, la clef de cette méthode est la fonction *fitness*. Cette méthode sélectionne les chromosomes de grande *fitness*. Pour notre cas l’ensemble d’ordonnancements avec une grande valeur de *fitness* a plus de chance d’être sélectionné. On récapitule cette méthode dans les 6 étapes suivantes :

- ***Etape 1 :*** *Pour chaque ordonnancement de la population (population de taille 12 X 24), calculer les fonctions objective et fitness (on aura 24 valeurs pour chacune des deux fonctions dans chaque population).*
- ***Etape 2 :*** *Déterminer la valeur moyenne de la fonction F(x) pour chaque population.*

$$F(x)_{moy}=\frac{\sum F(x)}{taille\,de\,la\,population=24} \tag{4.3}$$

- ***Etape 3 :*** *Déterminer le nombre prévu de copies de chaque ordonnancement dans la population intermédiaire (bassin de reproduction ou mating pool). Un ordonnancement avec une grande fitness a beaucoup plus de copies dans la population intermédiaire.*

$$Nombre\_de\_Copies(x) = \frac{F(x)}{F(x)_{moy}} \tag{4.4}$$

- ***Etape 4 :*** *Déterminer la probabilité de sélection de chaque ordonnancement.*

$$\Pr obabilité\_de\_sélection(x) = \frac{Nombre\_de\_Copies(x)}{taille\,de\,la\,population} \tag{4.5}$$

- ***Etape 5 :*** *Déterminez la probabilité cumulée pour chaque ordonnancement par l'addition de la probabilité cumulée de l'ordonnancement précédent avec la probabilité de sélection de l'ordonnancement actuel. Cette probabilité a une valeur entre 0 et 1.*

$$\begin{aligned}\Pr obabilité\_cumulée(x) = \Pr obabilité\_de\_sélection(x) \\ + \Pr obabilité\_cumulée(x-1)\end{aligned} \tag{4.6}$$

- ***Etape 6 :*** *Choisir un nombre aléatoire entre 0 et 1. L'ordonnancement qui a une probabilité cumulée supérieure à ce nombre aléatoire doit être sélectionné dans la population intermédiaire (mating pool). Au total 24 nombres aléatoires vont être choisis pour placer 24 ordonnancements dans la population intermédiaire.*

4.5. Croisement et mutation

4.5.1. Algorithmes de croisement

Une fois la population de la génération intermédiaire (*maiting pool*) a été remplie, les ordonnancements (Chromosomes) sont aléatoirement répartis en couples. La Figure.4.5 montre un exemple de croisement entre deux ordonnancements pères qui forment un seul couple. Nous avons utilisé l'opérateur de croisement en deux points car il est généralement considéré comme plus efficace [Haup 04], [Sart 02].

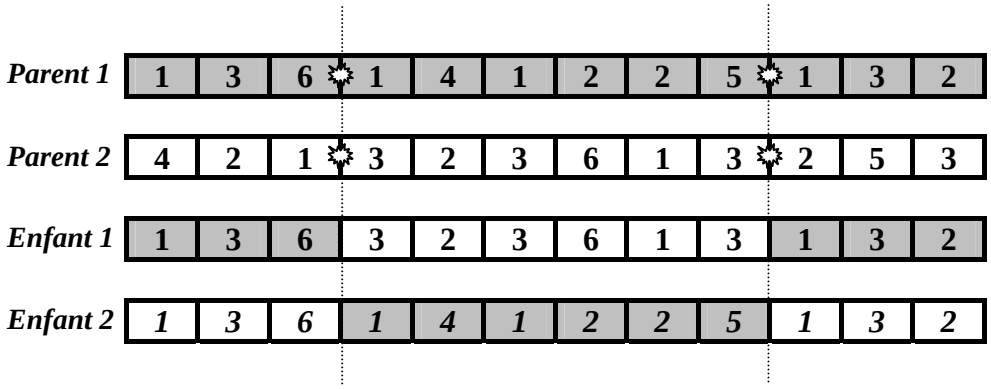

Figure.4.5. Exemple de croisement entre deux Ordonnancements pères

12 couples d'ordonnancements parents sont alors copiés et recombinés de façon à former des descendants (ordonnancements enfants) possédant des caractéristiques

issues de leurs parents. On forme ainsi une nouvelle population de 24 ordonnancements.

4.5.2. Algorithmes de mutation

Quand un ordonnancement enfant s'est crée à la population t, il est important de permettre à une petite probabilité de mutation de se produire. La Figure.4.6 montre le mécanisme de mutation sur un ordonnancement enfant de la dernière génération crée t, Cela revient à sélectionner aléatoirement une commande dans l'ordonnancement et changer sa machine de démarrage. Dans l'exemple ci-dessous, la commande 6 à été sélectionnée aléatoirement pour être fabriquée sur le Silo 1, Conteneur 1, au lieu de Silo 3, Conteneur 3.

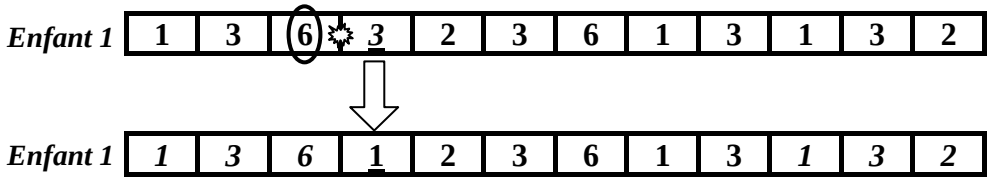

Figure.4.6. Exemple de mutation dans un ordonnancement enfant.

Ce mécanisme de mutation aide l'Algorithme Génétique à trouver rapidement l'ordonnancement optimal, en assurant une recherche aussi bien globale que locale.

4.6. Régénération

Une fois toutes les étapes de l'AG sont finis pour la première génération, on passe à la nouvelle génération en reprenant la même procédure à seule différence que la dernière population de la première génération devient automatiquement la première population de la deuxième génération (voir chapitre III). Le nombre de générations est fixé à 66 générations. Lors de l'exécution de notre programme de calculs, on ne peut pas poursuivre la dynamique de l'évolution de la population de taille 24 X 12 à travers 66 générations (66 itérations), par contre, il y a différentes statistiques à observer. Les mesures statistiques sont importantes pour ajuster les paramètres et mesurer aussi la performance de notre programme. On donne plus de détails dans le paragraphe qui suit en commentant les résultats et les statistiques observés dans notre programme de calcul.

5. Résultats et Commentaires

5.1. Description du programme de calcul

Le programme de placement permettant l'optimisation de l'ordonnancement dans l'atelier d'extrusion plastique à été développé sous Matlab 7. On dispose dans cette version d'un nouveau *tool box* d'Algorithmes Génétiques, on a exploité les subroutines des opérateurs génétiques (création de populations, sélection, croisement, et mutation) pour résoudre notre problème. La fonction principale de

notre programme ‘’Algo_Gen.mat’’ fait appel à d’autres sous fonctions de calcul, la Figure.4.7 représente l’organigramme du programme, pour plus de détail voir le listing du programme en annexe.

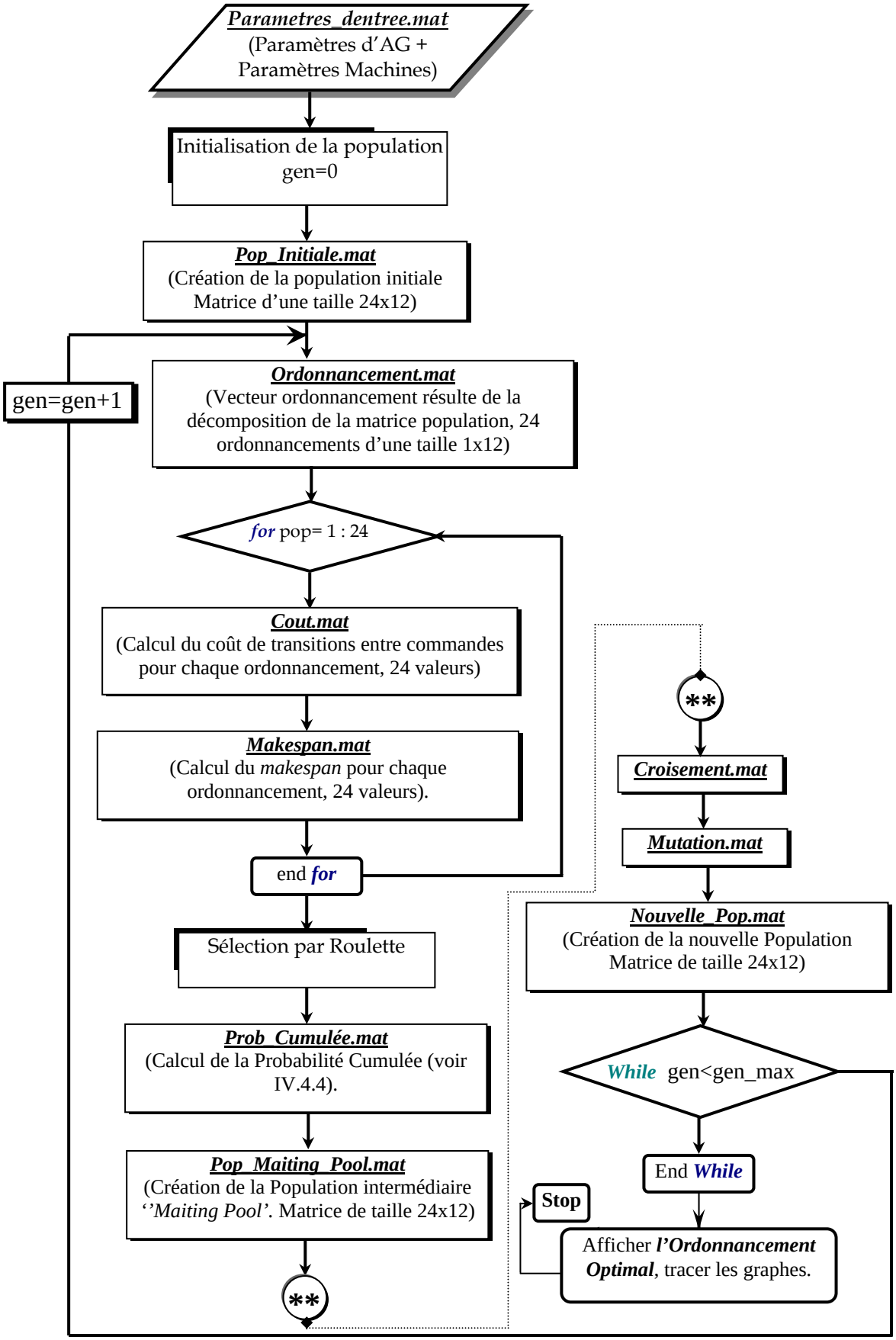

Figure.4.7 Organigramme d'optimisation en Ordonnancement par Algorithmes Génétiques

5.2. Résultats obtenus par le programme de calcul

Lors des recherches exhaustives de la solution optimale pour notre problème, il existe plusieurs ordonnancements au cours des différentes générations, en prenant le temps et le coût comme critères. Nous pouvons donc dire qu'il est indispensable de respecter certains compromis entre le coût et le *Makespan*, respecter aussi l'évolution naturelle de la fonction *Fitness*.

Nous allons effectuer la première exécution, le programme affiche le résultat d'ordonnancement optimal suivant :

Ordonnancement =												
	3	**1**	**2**	**2**	**6**	**2**	**4**	**1**	**5**	**3**	**1**	**2**

La question qui se pose fréquemment : le résultat d'ordonnancement obtenu représente-il une solution optimale globale ? Pour répondre à cette question, on commente les graphes obtenus.

La Figure.4.8 représente l'évolution de ''*fitness*'' en fonction du nombre de générations, une évolution croissante interprète l'amélioration de la nature d'ordonnancement (Chromosomes) au cours des générations, jusqu'à la génération 32, on remarque une convergence vers une valeur maximale de 3.34×10^{-3}. Cela signifie que la puissance d'ordonnancements (chromosomes) est augmentée jusqu'à la meilleure valeur qui correspond à l'ordonnancement optimal.

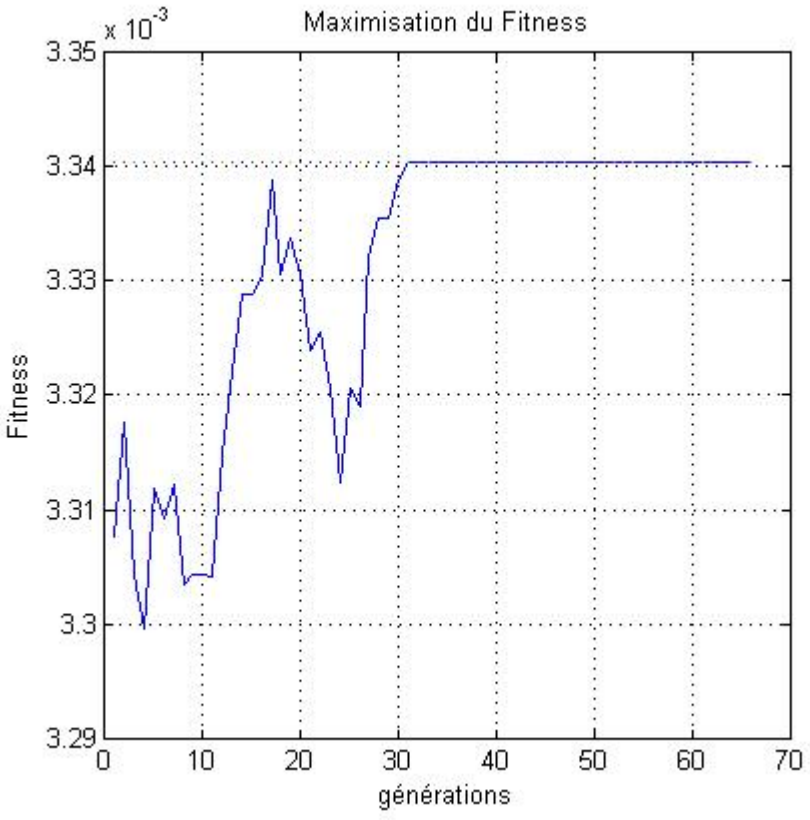

Figure.4.8. Maximisation de la fonction *Fitness*

La Figure.4.9 représente la convergence de la fonction coût. Le coût diminue en fonction du nombre de générations jusqu'à une valeur de 7765 Euros. A partir de la génération 12 notre programme donne le meilleur ordonnancement, cet ordonnancement nous propose la possibilité de baisser le coût de transitions entre les différentes commandes.

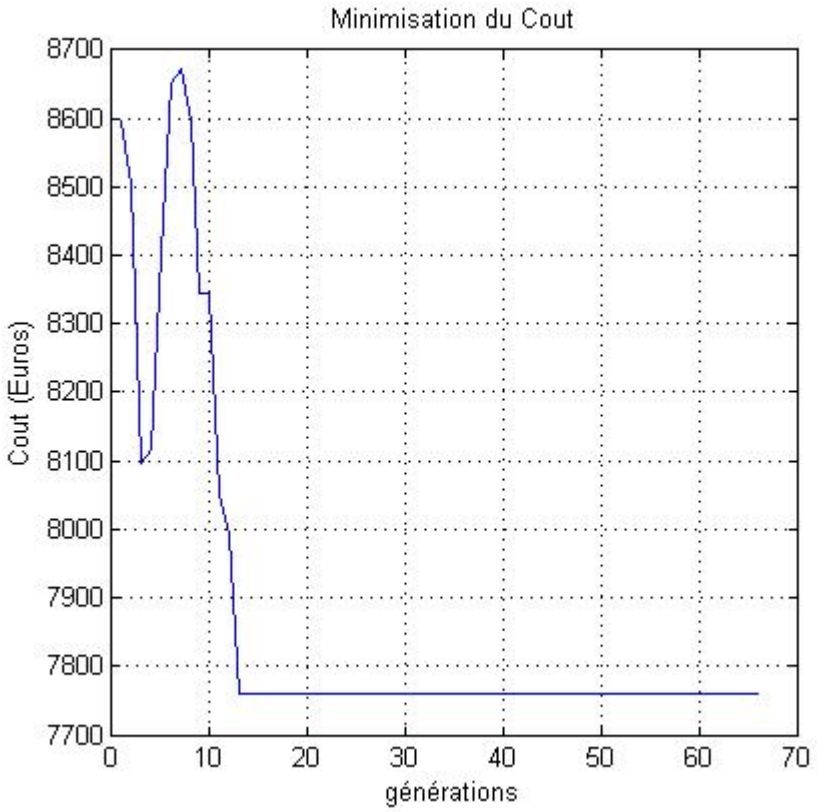

Figure.4.9. Minimisation de la fonction coût.

Il est intéressant de remarquer le compromis qui existe entre le coût et le *makespan*. La diminution du coût est affectée par une diminution du *Makespan*, sur la Figure.4.10, le meilleur ordonnancement trouvé à partir de la génération 32 fait diminuer progressivement la valeur du *Makespan* à une valeur de 810 minutes. Cet ordonnancement optimal permet donc de gagner plusieurs minutes sur la date de fin de toutes les tâches.

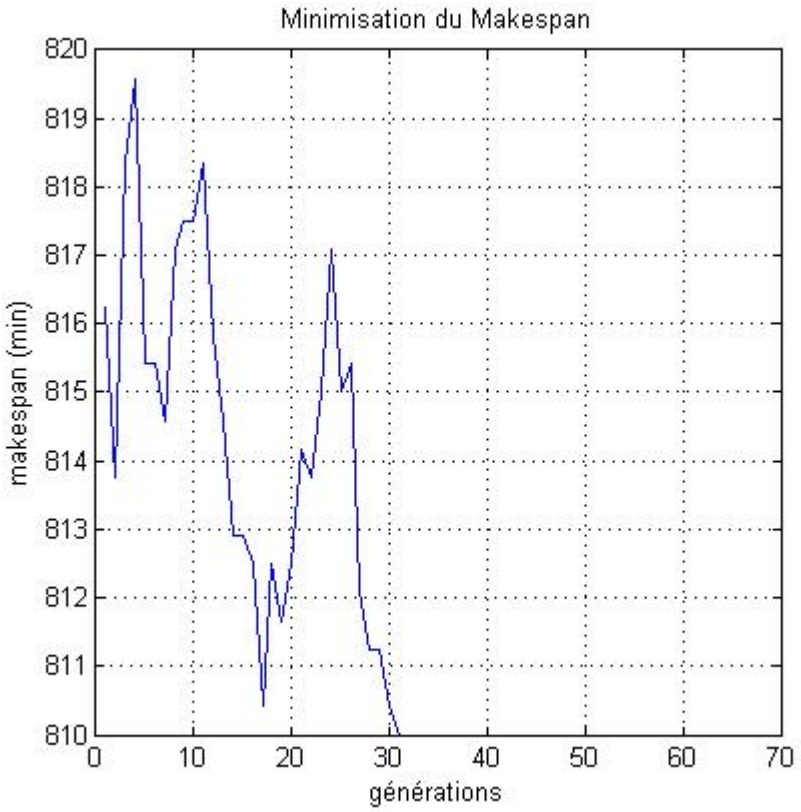

Figure.4.10. Minimisation de la fonction temps.

On conclut à partir des graphes 8, 9 et 10 qu'une très bonne nature d'un ordonnancement (Chromosome) a la forme :

3	**1**	**2**	**2**	**6**	**2**	**4**	**1**	**5**	**3**	**1**	**2**

Ce dernier, caractérisé par une très forte puissance ''*Fitness*'' de l'ordre de 3.3405×10^{-3}, baisse les charges de transitions entre les 6 commandes jusqu'à une valeur de 7759 euros, économise en parallèle le *makespan* nécessaire pour achever les différentes tâches de production à une valeur de 810 minutes

La représentation du *Makespan* (t =810 minutes) est facilitée en visualisant l'ordonnancement optimal sur le diagramme de Gantt (voir Figure.4.10), qui permet en outre de voir rapidement les moments d'occupation et de non occupation des différentes machines.

L'ordonnancement optimal conduit à placer :

- La commande 3 en première position sur le Silo 1, Conteneur 1, (t_{31} = 390 minutes).

- La commande 2 en première position sur le Silo 2, Conteneur 2, (t_{22} = 475 minutes).
- La commande 6 en deuxième position sur le Silo 2, Conteneur 2, (t_{62} = 555 minutes).
- La commande 4 en deuxième position sur le Silo 1, Conteneur 1, (t_{41} = 630 minutes).
- La commande 5 en première position sur le Silo 3, Conteneur 3, (t_{53} = 720 minutes),
- La commande 1 en troisième position sur le Silo 2, Conteneur 2, (t_{12} = 810 minutes).

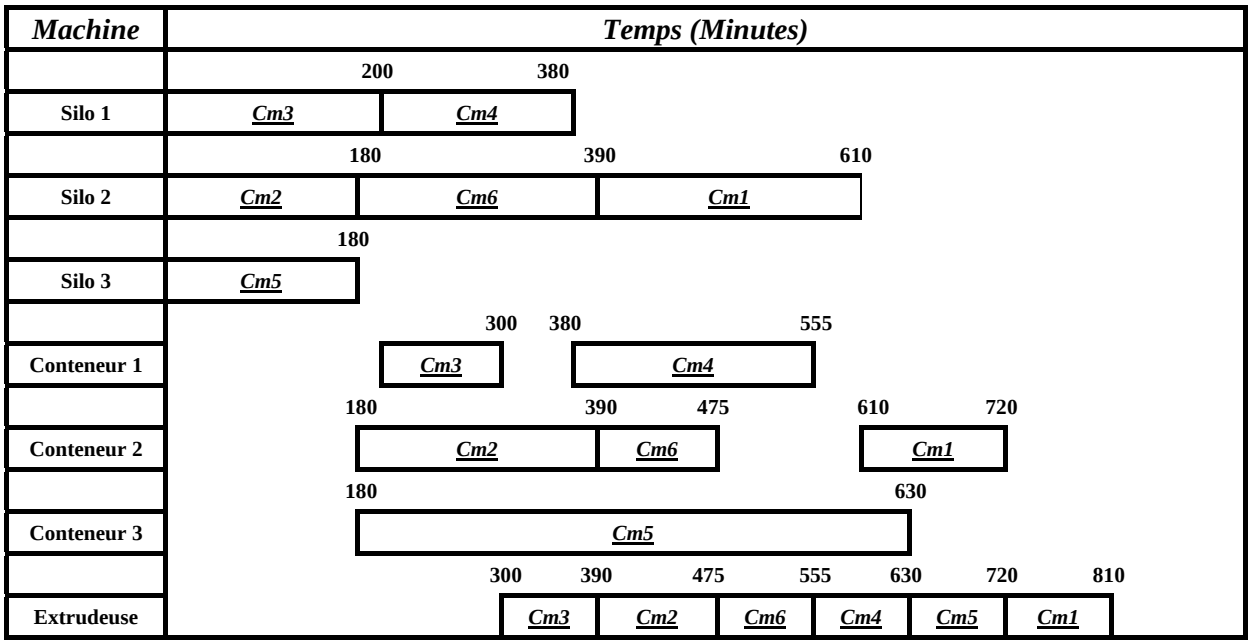

Figure.4.11. Ordonnancement optimal.

Le tableau.4.5 fait une simple comparaison entre les valeurs de la population originale et finale :

Tableau.4.5 Comparaison de la Population Originale et Finale.

Population	*Fitness*	Coût de transitions	*Makespan*
Originale	0.0033366	8107	820
Finale	0.0033405	7759	810

Après avoir exécuter le programme de calcul deux fichiers texte sont crées, ils représentent la première population de la première génération ‘‘pop_ première.txt’’, et la population finale de la dernière génération ‘‘pop_finale.txt’’ respectivement. Pour mieux visualiser ces fichiers on les a importés dans une feuille de calcul Excel (voir tableaux 4.6 et 4.7). On peut mieux remarquer la différence entre les valeurs des deux populations, la taille de chaque population est de 24. Les 12 premières

colonnes du tableau.4.6 représentent les ordonnancements crées d'une façon stochastique, les 3 dernières colonnes représentent respectivement le *Makespan*, Coût et *Fitness* engendrés par chaque ordonnancement proposé. Le mécanisme de l'évolution génétique conduit à une population finale optimale caractérisée par des valeurs constantes, l'ordonnancement trouvé est une solution optimale, il engendre également un *Makespan*, un Coût et une *Fitness* optimaux (voir tableau.4.7).

Tableau.4.6 Population Initiale.

Ordonnancements (Chromosomes)												***Makespan***	**coût**	***Fitness***
6	3	1	1	4	1	3	3	2	1	5	1	820	7874	0.0032995
5	1	4	1	6	1	1	2	3	1	2	1	810	7591	0.0033423
3	2	2	1	4	1	1	1	6	1	5	3	820	7566	0.0033029
3	1	2	2	1	2	5	2	4	3	6	1	810	8107	0.0033366
4	3	6	1	5	1	2	2	3	2	1	2	795	7771	0.0034017
3	2	6	1	5	2	4	1	2	1	1	3	820	9625	0.0032806
4	1	2	1	5	1	3	2	1	2	6	3	785	7636	0.0034455
6	1	4	2	3	2	5	2	1	1	2	2	835	9823	0.0032214
6	2	5	1	1	1	3	2	4	1	2	1	820	9476	0.0032822
5	1	1	3	3	3	2	3	6	1	4	1	810	6792	0.0033513
6	2	5	1	3	3	2	2	1	2	4	1	820	9590	0.0032809
4	2	6	1	2	2	3	3	5	2	1	2	795	7837	0.0034009
2	3	5	3	3	1	1	1	4	2	6	1	850	8397	0.0031807
6	2	3	2	1	3	5	3	2	2	4	3	820	7129	0.0033077
6	1	4	1	5	2	3	1	1	1	2	2	835	9696	0.0032227
3	1	2	2	6	2	5	1	4	1	1	1	810	9164	0.0033249
3	3	2	3	6	2	4	3	5	2	1	2	810	7759	0.0033405
6	3	5	1	3	2	1	2	4	2	2	1	820	9544	0.0032814
6	1	5	2	3	1	1	1	4	1	2	1	835	9544	0.0032243
5	3	6	1	2	1	1	1	4	2	3	3	820	9962	0.0032769
3	1	6	2	2	1	4	3	5	2	1	2	810	8197	0.0033356
6	1	4	2	2	2	1	1	3	1	5	1	835	8419	0.0032360
4	2	6	2	5	2	3	1	2	2	1	3	795	9265	0.0033845
3	1	2	1	1	1	4	2	5	1	6	2	810	9512	0.0033210

Tableau.4.7 Population Finale.

Ordonnancements (Chromosomes)												***Makespan***	**coût**	***Fitness***
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405
3	1	2	2	6	2	4	1	5	3	1	2	810	7759	0.0033405

6. Conclusion

La procédure d'optimisation basée sur les Algorithmes Génétiques, que nous avons proposé dans ce travail est très efficace pour la génération d'ordre de lancement des différentes commandes.

Notre travail s'inclue dans un processus global d'aide à la décision, où le décideur peut, en faisant varier quelques paramètres (en particulier sur les commandes des clients : temps et coût de fabrication), rechercher la meilleure solution. Cette solution correspond à un ordonnancement optimal qui permet au décideur de diminuer le temps lié à la production (*Makespan*) et le coût lié aux transitions entre différentes commandes.

Notre programme établit donc un ordre optimal de passage de six commandes sur 3 entrées possibles (Silos), un *Makespan* optimal et un coût de transitions optimal. Les avantages de notre algorithme sont :

1. L'algorithme converge dans un temps raisonnable tout en respectant les lois de l'évolution génétique, (sélection, croisement, mutation….etc.).

2. La solution produite par l'algorithme correspond à une solution optimale caractérisée par un ordre de passage sur des machines entrées.

3. L'algorithme donne possibilité à résoudre des problèmes en extrusion plastique de grande taille (Nombres de commandes élevées, coût et temps au choix de décideur).

Conclusion et Perspectives

Conclusion

Le travail exposé dans ce mémoire s'intéresse à l'étude d'une problématique industrielle concernant l'ordonnancement dans une entreprise d'extrusion plastique. Dans cette problématique, nous avons justifié le choix qui nous a amené à étudier ce type de système de production.

Après avoir présenter les différents problèmes d'ordonnancement dans la gestion de production et un certain nombre de méthodes de résolution, nous avons choisi d'appliquer les Algorithmes Génétiques à notre atelier.

Basés sur un processus d'évolution naturelle, les Algorithmes Génétiques appartiennent aux techniques d'optimisation se basant sur l'exploration aléatoire de l'espace des solutions, ils cherchent l'optimum à partir d'une population de points et non à partir d'un seul point. L'algorithme de base des AG se décompose en 4 parties : la sélection, le croisement, la mutation et la reproduction. Ils se rapprochent d'avantage de la formulation de problème d'ordonnancement, chaque chromosome correspond à une solution d'ordonnancement possible, le meilleur chromosome obtenu après plusieurs générations correspond à la meilleure solution d'ordonnancement. Nous avons montré que les Algorithmes Génétiques appliqués à l'ordonnancement sont efficaces car ils trouvent de meilleurs résultats et atteignent l'optimum, nous avons calculé cette solution représentée par l'ordonnancement optimal, cela donne avantage aux responsables de la production dans la perspective d'améliorer les choix des différents paramètres liés à la production du plastique en vue de baisser les charges.

Nous avons vu que l'ordonnancement obtenu par notre programme de calcul établit un ordre de passage optimal de six commandes sur 3 entrées possibles (Silos), le résultat de l'ordonnancement donne un *Makespan* optimal et un coût de transitions optimal aussi. Les paramètres machines peuvent être changés en augmentant ou en diminuant la taille du problème, suivant le type de commandes on peut changer par exemple les temps liés aux différentes tâches, les coûts de fabrications liés au marché international du polyéthylène.

Perspectives

Pour le moment, nous avons implémenté notre système de production sur trois entrées, une seule sortie. Il serait intéressant de distribuer notre système sur plusieurs machines distinctes en entrée et en sortie afin de profiter de la puissance de calculs de chaque machine mais aussi, de mieux gérer les conflits qui peuvent apparaître dans la gestion des ressources.

Dans notre travail, nous avons simplifié le système de production, en réalité, des machines peuvent tomber en panne et les perturbations ainsi engendrées doivent être absorbées. Ainsi, à tout niveau et à tout moment, des perturbations risquent de venir gêner le bon déroulement du plan de production prévu, chaque perturbation influe sur l'ordonnancement, il est donc nécessaire de remettre en cause un certain nombre

de décisions. La question est de savoir quelles décisions à remettre en cause. Jusqu'à où la perturbation va-t-elle se propager. Il peut donc être utile de réfléchir à une méthode permettant de modéliser ces perturbations, les implémenter dans le même programme, finir ensuite par réordonnancer en temps réel tout l'atelier.

Enfin, une troisième perspective intéressante serait la prise en compte des tâches de maintenance, l'introduction de ces tâches est indispensable par contre elle peut pénaliser fortement la production, en créant des perturbations, et des retards d'exécution des tâches. Il est encore utile de les modéliser pour réordonnancer en temps réel le système de production.

Références Bibliographie

Références Bibliographiques

[Arti 97] A. Artiba, S.E. Elmaghraby, ***The planning and scheduling of production systèms, methodologies and applications,*** London-Weinhein-New York – Tokyo – Melbourne – Madras, CHAPMAN & HALL, 1997, p268.

[Bapt 00] P.Baptise, ***scheduling of identical machines***, Compiène, Octobre 2000

[Bruc 99] P. Brucker, ***Complex scheduling problems***, Universität osnabrück. D-49069, Germany, June 1999, p41.

[Bruc & Krav 99] P. Brucker, S. Kravchenko, ***Premption can make parallel machine scheduling problems hard***, University of Osnabrück, 1999, p5.

[Carl & Chré 88] J. Carlier, P. Chrétiènne, ***Problèmes d'ordonnancement : modélisation, complexité, algorithmes***, Paris : Masson, 1988, p326.

[Caux 95] C. Caux, H. Pierreval, & M.C. Portmann, M. C, ***Les algorithmes génétiques et leur application aux problèmes d'ordonnancement***. Revue d'Automatique de Productique et d'Informatique Industrielle, 1995, 29, 409–443.

[Esqu & Lope 99] Patrick Esquirol, Pierre Lopez, ***L'ordonnancement***, Paris Economica, 1999. p 141.

[Giar 03] Vincent Giard, ***Gestion de la production et des Flux***, Paris Economica, 2003, p 1229.

[Giar 88] Vincent Giard, ***Gestion de production***, Paris Economica, 1988

[Gold 89] D.Goldberg***, Genetic algorithms in search optimization and machine learning***. Addison-Wesley, 1989.

[Goth 04] Groupe GOTHA, ***Modèle et algorithmes en ordonnancement***, Ellipses, 2004, p 227.

[Goth 93] Groupe GOTHA, ***Les problèmes d'ordonnancement***, Recherche Opérationnelle, Vol. 27, $n^0$1, 1993.

[Haup 04] R.L.Haupt, S.E.Haupt, ***Practical Genetic Algorithms, Second Edition,*** Wiley Interscience, 2004, p 251

[Hent & Guin, 97] Hentous, H. Guinet, ***A new heuristic to minimize completion time for the three machines Job-Shop problems***, International conference on industrial engineering and production management, tome 2, 1997, LYON, p 600-609.

[Hent 95] H. Hentous, ***Flow Shop hybride sous contraintes***, Rapport de DUA de recherche opérationnelle, Université de Joseph Fourier, 1995. p 37.

[Hent 99] H. Hentous, ***Contribution au pilotage des systèmes de production de type Job-Shop***. Thèse de doctorat, INSALYON, 1999, 149p.

[Hola 75] J.H. Holland***, Adaptation in natural and artificial systems***. Ann Arbor, MI: The University of Michigan Press. 1975.

[Houc 04] C.R.Houck, A.Joines, G.Kay, ***Genetic Algorithm for Function Optimization: A Matlab Implementation***, DMI-9322834, 2004, p 14.

[Jave 97] G. Javel, ***Organisation et gestion de la production***, Paris, Masson 1997, p 401.

[Lakh 98] C.Lakhmi, N.Jain, M.Martin, ***Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications,*** CRC Press, 1998, p 297

[Lope 01] Lopez, P. Roubellat, ***Ordonnancement de la production***. Paris Hermès, 2001, 432p

[Math 04] The Mathworks, ***Genetic Algorithm and Direct Search Toolbox for Matlab,*** User's Guide, Version 1, 2004, p 210

[Mela 99] M. Melanie, ***An Introduction to Genetic Algorithms,*** MIT Press Fifth printing, 1999, p 143

[Marc et al 99.a] N.Marco, J.A.Desideri & S.Lanteri, ***Multi-Objective optimization in CFD by Genetic Algorithms***, Rapport de recherche N 3686, INRIA, avril 1999

[Marc et al 99.b] N.Marco, J.A.Desideri, ***Numerical solution of optimization cases by Genetic Algorithms***, rapport de recherche N3622, INRIA, février 1999

[Mich & Dani 97] F. Michel, L. Daniel, ***L'ordonnancement en pratique : Outils techniques en usage dans l'industrie***, Deuxième Congrés international Franco- Québécois de génie industriel –ALBI 1997, p 15.

[O'Re 05] U.O'Reilly, T.Yu, R.Riolo, B.Worzel***, Genetic Programming Theory and Practice II***, Springer ,2005, p 317

[Oura 01] S. Ourari, ***Ordonnancement en temps réel en présence d'évènements aléatoires***. Thèse de Magister E.M.P., 2001. p 98.

[Park 03] B.J. Park, H.R. Choi, H.S. Kim, ***A hybrid genetic algorithm for the job shop scheduling problems***, Journal of Computers & Industrial Engineering , 2003, 45 (597–613).

[Penn 99] R.Penny, ***Numerical method using Matlab***, second edition, prentice hall, 1999

[Pine 95] Pinedo, M. S, ***Scheduling : Theory, algorithms and systems***, Englewood Cliff (N.J) : Prentice Hall, 1996, Mons (Belgique) p i-xx-iv.

[Plip 98] C. Flipo-Dhaenens, ***Optimisation d'un réseau de production et de distribution,*** Thèse de doctorat, INPG de Grenoble. 1998, p 197.

[Rend 95] L. Rendoz, ***Algorithmes Génétiques et Réseaux de Neurones***, Hermès, 1995, p 237.

[Rinn 76] Rinnooy Kan, ***Machine sequencing problem: Classification, complexity and computation***, The Hague: Nijhoff, 1976, p 180.

[Sart 02] M. Sartor, J. Guillot, ***Conception optimal de systèmes mécaniques***, Cours de techniques numériques d'optimisation, INSA, Toulouse, 2002, p 92.

[Vach 00] J.P Vacher, ***Un système adaptatif par agents avec utilisation des algorithmes génétiques multi-objectifs : Application à l'ordonnancement d'atelier de type job-shop***, Thèse de doctorat, Université du Havre, 2000, p 341.

[Zegh 03] N. Zeghichi, ***Optimisation multicritères des conditions de coupes en fraisage par algorithmes génétiques et la méthode Nelder Mead***, Mémoire de Magister, Université de Biskra, 2003, p 94.

Annexes

- **Statistiques et Analyse Atelier**

1. Statistique Déchet Plastique

La figure.1 ci-dessous représente le déchet total de l'atelier enregistré entre la semaine 18 et la semaine 28, 2005. Remarquons que la quantité de déchet par rapport à la production total du plastique prend des valeurs considérables et celà pour chaque semaine. Notons que le déchet de transition qui est l'objet de notre étude représente entre 70 et 80% de la quantité de déchet total.

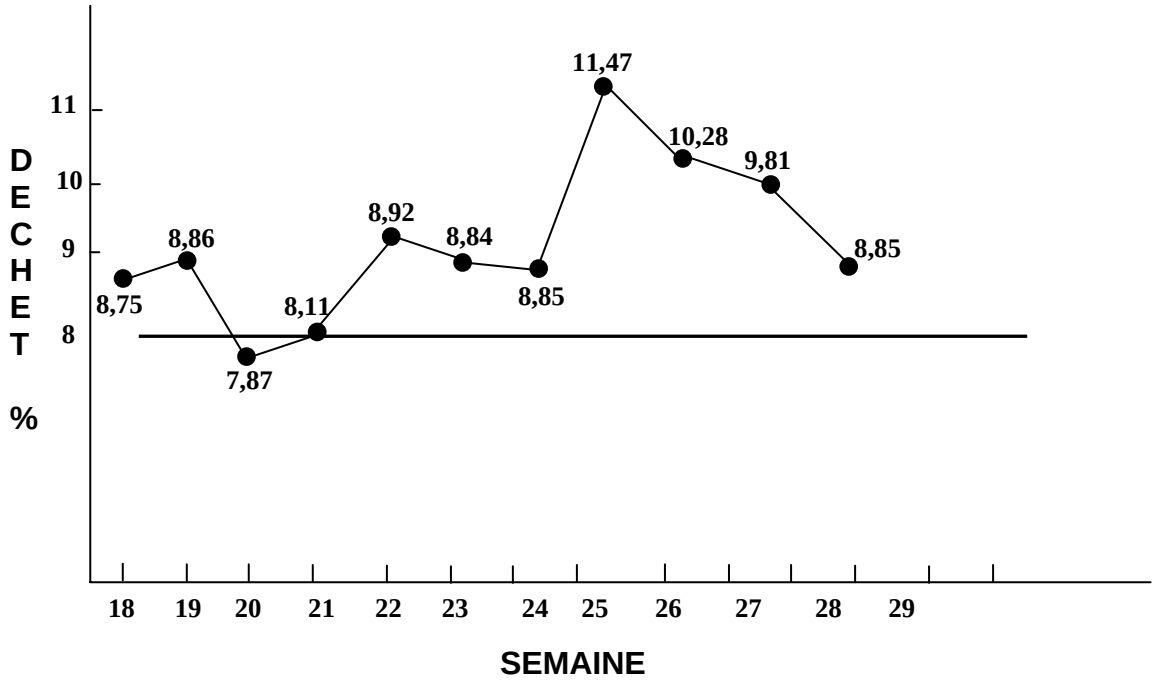

Figure.1. Déchet Total Atelier

1. Analyse de consommation de la Matière Première

Le Tableau.1 enregistre la consommation de la matière brute (matière première entrante), et la production nette (commandes ou matière sortante sous forme de bobines) au cours de la semaine 28 2005, la différence entre les deux représente la perte de la production, ce déchet qui est considérable contient comme on l'a indiqué au paragraphe précédent entre 70 et 80% de plastique transitoire, ce plastique créé suite aux différents changements de commandes pénalise fortement la production.

Tableau.1 Analyse Consommation Matière Première, Semaine 28, 2005

Machines	Relevé Plastique Brut (Kg)	Production Nette (Kg)	Déchet en (Kg)	% Déchet	Changements
New 4	**91 070**	**86 531**	**4 539**	**5,25**	**4**
M29	**32 736**	**29 900**	**2 836**	**9,48**	**7**
BRAMPTON	**36 928**	**29 898**	**7 030**	**23,51**	**5**
New 3	**60 398**	**53 743**	**6 665**	**12,38**	**7**
M5	**51 848**	**49 503**	**2 345**	**4,74**	**11**
M22	**45 759**	**43 266**	**2 493**	**5,76**	**5**
M91	**45 968**	**43 167**	**2 801**	**6,49**	**7**
Alpine	**27 375**	**23 537**	**3 838**	**16,31**	**5**

- **Exemple du mécanisme d'évolution de l'AG lors de la première génération :**

Afin de mieux comprendre le mécanisme d'évolution de l'ensemble d'ordonnancements constituants une population, nous proposons des résultats du programme lors de la première génération ($1^{ère}$ itération).

- Le programme commence par générer aléatoirement la population initiale de 24 ordonnancements (solutions possibles) et qui a la forme :

pop =

6	3	1	1	4	1	3	3	2	1	5	1
5	1	4	1	6	1	1	2	3	1	2	1
3	2	2	1	4	1	1	1	6	1	5	3
3	1	2	2	1	2	5	2	4	3	6	1
4	3	6	1	5	1	2	2	3	2	1	2
3	2	6	1	5	2	4	1	2	1	1	3
4	1	2	1	5	1	3	2	1	2	6	3
6	1	4	2	3	2	5	2	1	1	2	2
6	2	5	1	1	1	3	2	4	1	2	1
5	1	1	3	3	3	2	3	6	1	4	1
6	2	5	1	3	3	2	2	1	2	4	1
4	2	6	1	2	2	3	3	5	2	1	2
2	3	5	3	3	1	1	1	4	2	6	1
6	2	3	2	1	3	5	3	2	2	4	3
6	1	4	1	5	2	3	1	1	1	2	2
3	1	2	2	6	2	5	1	4	1	1	1
3	3	2	3	6	2	4	3	5	2	1	2
6	3	5	1	3	2	1	2	4	2	2	1
6	1	5	2	3	1	1	1	4	1	2	1
5	3	6	1	2	1	1	1	4	2	3	3
3	1	6	2	2	1	4	3	5	2	1	2
6	1	4	2	2	2	1	1	3	1	5	1
4	2	6	2	5	2	3	1	2	2	1	3
3	1	2	1	1	1	4	2	5	1	6	2

- On décompose la population initiale en vecteurs d'ordonnancement de taille 1×24, 24 ordonnancements sont décomposés, le premier ordonnancement a la forme :

ordonnancement =

6	3	1	1	4	1	3	3	2	1	5	1

- Le programme calcul le coût de transition entre les six commandes pour chaque ordonnancement, 24 valeurs sont enregistrées, on donne l'exemple du coût associé au premier ordonnancement :

cout_ord =

7874

- Le programme calcul ensuite le *Makespan* qui correspond à chaque ordonnancement, 24 valeurs sont enregistrées, on donne l'exemple du *Makespan* associé au premier ordonnancement :

tps_max =

820

- Pour chaque ordonnancement, le programme calcul la valeur de *Fitness* qui caractérise la puissance de chaque ordonnancement à s'adapter avec les objectifs imposés (on aura 24 valeurs), le *Fitness* du premier ordonnancement a la valeur :

f_fitness =

1.0e+004 * 3.0307

- Le programme calcul la probabilité cumulée de chaque ordonnancement (24 valeurs), cette probabilité a une valeur entre 0 et 1.

prob_cum =

0.0416 0.0837 0.1253 0.1673 0.2102 0.2515 0.2949 0.3355 0.3768 0.4190
0.4604 0.5032 0.5433 0.5849 0.6255 0.6674 0.7095 0.7508 0.7915 0.8327 0.8748
0.9155 0.9582 1.0000

- On choisi ensuite un nombre aléatoire entre 0 et 1, l'ordonnancement qui a une probabilité cumulée supérieure à ce nombre aléatoire doit être sélectionné dans la population intermédiaire (*Mating Pool*), au total 24 nombres aléatoires vont être choisis pour placer 24 ordonnancement dans la population intermédiaire suivante :

mating_pool =

6	3	5	1	3	2	1	2	4	2	2	1
3	3	2	3	6	2	4	3	5	2	1	2
3	2	2	1	4	1	1	1	6	1	5	3
6	2	5	1	3	3	2	2	1	2	4	1
6	2	5	1	3	3	2	2	1	2	4	1
6	2	5	1	1	1	3	2	4	1	2	1
3	1	2	2	1	2	5	2	4	3	6	1
3	3	2	3	6	2	4	3	5	2	1	2
3	3	2	3	6	2	4	3	5	2	1	2
6	3	5	1	3	2	1	2	4	2	2	1
4	2	6	1	2	2	3	3	5	2	1	2
6	2	3	2	1	3	5	3	2	2	4	3
3	2	2	1	4	1	1	1	6	1	5	3
6	2	5	1	3	3	2	2	1	2	4	1
6	3	5	1	3	2	1	2	4	2	2	1
6	1	4	2	2	2	1	1	3	1	5	1
4	1	2	1	5	1	3	2	1	2	6	3

4	1	2	1	5	1	3	2	1	2	6	3
3	1	6	2	2	1	4	3	5	2	1	2
3	2	6	1	5	2	4	1	2	1	1	3
5	3	6	1	2	1	1	1	4	2	3	3
6	1	4	2	2	2	1	1	3	1	5	1
3	2	6	1	5	2	4	1	2	1	1	3
3	2	6	1	5	2	4	1	2	1	1	3

- Une fois la population intermédiaire à été remplie, les ordonnancements sont aléatoirement répartis en couples, 12 couples sont recombinés de façon à former 24 ordonnancements enfants par croisement, on permet ensuite à une probabilité de l'ordre de 0.01 à sélectionner aléatoirement une commande dans l'ordonnancement enfant et changer sa machine de démarrage, ce mécanisme qu'on appel Mutation aide l'algorithme à trouver rapidement l'optimum global. une fois toutes ces étapes sont finis on aura une nouvelle population :

new_pop =

3	1	2	1	1	1	4	2	5	1	6	2
3	2	2	3	6	2	4	3	5	3	1	2
5	1	2	1	4	1	1	1	6	1	3	3
6	2	5	1	3	3	2	2	1	3	4	1
6	2	5	1	3	3	2	2	1	2	4	1
6	2	2	1	1	1	3	2	4	1	5	2
3	1	2	2	1	2	4	3	5	2	6	1
3	3	2	3	6	2	4	3	5	2	1	2
3	3	2	3	6	2	4	3	5	2	1	2
6	3	5	2	3	2	1	2	4	2	2	1
4	2	6	1	3	2	2	2	5	2	1	2
6	2	3	2	1	3	5	3	2	1	4	1
3	1	2	1	1	1	4	2	5	1	6	2
3	2	2	1	4	3	1	1	6	1	5	2
6	2	5	1	3	3	2	2	1	2	4	1
6	2	5	1	3	3	2	2	1	2	4	1
6	2	5	1	1	2	3	2	4	1	2	1
3	1	2	2	6	2	5	2	4	3	1	1
3	3	2	2	6	2	4	3	5	2	1	2
3	3	2	3	6	2	4	3	5	2	1	2
3	3	5	1	6	2	1	2	4	2	2	1
4	2	6	1	2	2	5	1	3	2	1	2
6	2	3	2	2	2	5	3	1	2	4	3
3	2	2	1	6	2	1	1	4	3	5	3

- Le programme passe enfin à la nouvelle génération en prenant la dernière population comme une population initiale jusqu'à atteindre la génération 66 qui est un critère d'arrêt pour notre cas.